Decentralized Sparse Multitask RLS Over Networks

Xuanyu Cao and K. J. Ray Liu, *Fellow, IEEE*

***Abstract*—Distributed adaptive signal processing has attracted much attention in the recent decade owing to its effectiveness in many decentralized real-time applications in networked systems. Because many natural signals are highly sparse with most entries equal to zero, several decentralized sparse adaptive algorithms have been proposed recently. Most of them is focused on the single task estimation problems, in which all nodes receive data associated with the same unknown vector and collaborate to estimate it. However, many applications are inherently multitask oriented and each node has its own unknown vector different from others. The related multitask estimation problem benefits from collaborations among the nodes as neighbor nodes usually share analogous properties and thus similar unknown vectors. In this paper, we study the distributed sparse multitask recursive least squares (RLS) problem over networks. We first propose a decentralized online alternating direction method of multipliers algorithm for the formulated RLS problem. The algorithm is simplified for easy implementation with closed-form computations in each iteration and low storage requirements. Convergence analysis of the algorithm is presented. Moreover, to further reduce the complexity, we propose a decentralized online subgradient method with low computational overhead. We theoretically establish its mean square stability by providing upper bounds for the mean square deviation and the excess mean square error. A related distributed online proximal gradient method is presented and extension to clustered multitask networks is also provided. The effectiveness of the proposed algorithms is corroborated by numerical simulations and an accuracy-complexity tradeoff between the proposed algorithms is highlighted.**

***Index Terms*—Distributed estimation, decentralized optimization, adaptive signal processing.**

I. Introduction

IN THE last decade, distributed adaptive signal processing has emerged as a vital topic because of the vast applications in need of decentralized real-time data processing over networked systems. For multi-agent networks, distributed adaptive algorithms only rely on local information exchange, i.e., information exchange among neighbor nodes, to estimate the unknowns. This trait endows distributed adaptive algorithms with low communication overhead, robustness to node/link failures and scalability to large networks. In the literature, the centralized least mean squares (LMS) and recursive least squares (RLS) [1] have been extended to their decentralized counterparts [2]–[6] to deal with estimation problems over networks. Furthermore, many natural signals are inherently sparse with most entries equal to zero such as the image signals and audio signals. Sparsity of signals are particularly conspicuous in the era of big data: for many applications, redundant input features (e.g., a person's salary, education, height, gender, etc.) are collected to be fed into a learning system to predict a desired output (e.g., whether a person will resign his/her job). Most input features are unrelated to the output so that the weight vector between the input vector and the output is highly sparse. As such, several sparse adaptive algorithms have been proposed such as the sparse LMS in [7], [8], the sparse RLS in [9], the distributed sparse LMS in [10], [11], the distributed sparse total least-squares (TLS) for noisy input data in [12], the decentralized sparsity-aware adaptive learning based on projection techniques[13], and the distributed sparse RLS in [14].

Most of the decentralized sparse adaptive algorithms are focused on the single task estimation problem, in which all nodes receive data associated with the same unknown vector and collaborate to estimate it. On the contrary, many applications are inherently multitask-oriented, i.e., each node has its own unknown vector different from others'. For instance, in a sensor network, each node may want to estimate an unknown vector related to its specific location and thus different nodes have different unknown vectors to be estimated. In fact, several decentralized multitask adaptive algorithms have been proposed in the literature [15] including the multitask diffusion LMS in [16], asynchronous multitask LMS in [17], sparse multitask LMS in [18] and multitask LMS with linear equality constraints in [19]. In these works, networks are divided into clusters so that nodes within each cluster share the same task (weight vector) and connected clusters have similar tasks (usually embodied by a squared l_2 proximity regularization). An alternative family of multitask network models has been examined in [20]–[23]. In these works, the entire network has a set of tasks (unknown vectors to be estimated) and each node only receives signals related to a subset of these tasks. Nodes exchange information with their neighbors to infer the overlapping tasks cooperatively. This multitask model is named the node-specific parameter estimation (NSPE) problem. Specifically, incremental-based LMS and diffusion-based LMS are put forth in [22] and [21], respectively, for NSPE. Furthermore, in [20], incremental-based distributed RLS for NSPE has been studied, in which each node needs to estimate a local vector of local interest as well as a global vector shared by all nodes. Additionally, unsupervised multitask learning has been investigated in [24], [25], where each node is unaware of its cluster relations with its neighbors.

Manuscript received February 21, 2017; revised June 19, 2017; accepted September 3, 2017. Date of publication September 7, 2017; date of current version September 29, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Qingjiang Shi. *(Corresponding author: Xuanyu Cao.)*

X. Cao is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: x.cao@princeton.edu).

K. J. R. Liu is with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA (e-mail: kjrliu@umd.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2017.2750110

In such a blind case, without the prior knowledge of cluster affiliations, each node has to infer the clusters and learn the weight vectors simultaneously and adaptively. Moreover, applications of multitask adaptive signal processing, including the study of Parkinson's disease [26], power system state estimation [27], beamforming and localization [28], have been considered in the literature. A comprehensive review of NSPE and multitask networks is available in [23].

To the best of our knowledge, all the existing distributed adaptive algorithms for sparse multitask estimation problems are based on decentralized versions of LMS. The RLS based sparse multitask estimation problems have not aroused much attention. It is well known that the RLS possesses faster convergence speed than the LMS does in both centralized adaptive filters [1] and decentralized adaptive networks [29]. Hence, the RLS is more suitable for applications in need of fast and accurate tracking of the unknowns than the LMS, especially when the devices are capable of dealing with computations of moderately high complexity (which is the case as the computational capability of devices is increasing drastically). This motivates us to study the decentralized sparse multitask RLS problem over networks. The main contributions of this paper are summarized as follows.

- A global networked RLS minimization problem is formulated. In accordance with the multitask nature of the estimation problem, each node has its own weight vector. Since neighbor nodes often share analogous properties and thus similar weight vectors, we add regularization term to penalize deviations of neighbors' weight vectors. To enforce sparsity of the weight vectors, we further introduce l_1 regularization.
- A decentralized online alternating direction method of multipliers (ADMM) algorithm [27], [30]–[34] is proposed for the formulated sparse multitask RLS problem. The proposed ADMM algorithm is simplified so that each iteration consists of simple closed-form computations and each node only needs to store and update one $M \times M$ matrix and six M dimensional vectors, where M is the dimension of the weight vectors. Convergence analysis of the proposed ADMM algorithm is also provided.
- To overcome the relatively high computational cost of the proposed ADMM algorithm, we further present a decentralized online subgradient method, which enjoys lower computational complexity. The mean square stability of the proposed subgradient method is established by upper bounding its mean sqaure deviation and excess mean square error. A related distributed online proximal gradient method is also presented and its connections with the subgradient method is discussed. Extension to clustered multitask networks is also considered.
- Numerical simulations are conducted to corroborate the effectiveness of the proposed algorithms. Their advantages over the single task sparse RLS algorithm in [14] are highlighted. We also observe an accuracy-complexity tradeoff between the proposed algorithms.

The roadmap of the remaining part of this paper is as follows. In Section II, the sparse multitask RLS problem is formally formulated. In Section III, we propose and simplify a decentralized online ADMM algorithm for the formulated RLS problem. In Section IV, we propose a decentralized online subgradient method for the formulated problem in order to reduce computational complexity. In Section V, numerical simulations are conducted. In Section VI, we conclude this work.

II. The Statement of the Problem

We consider a network of N nodes and some edges between these nodes. We assume that the network is a simple graph, i.e., the network is undirected with no self-loop and there is at most one edge between any pair of nodes. Denote the set of neighbors of node n (those who are linked with node n by an edge) as Ω_n. The network can be either connected or disconnected (there does not necessarily exist a path connecting every pair of nodes). Time is divided into discrete slots denoted as $t = 1, 2, \ldots$. Each node n has an unknown (slowly) time-variant M dimensional weight vector $\widetilde{\mathbf{w}}_n(t) \in \mathbb{R}^M$ to be estimated. The formulated network is therefore a multitask learning network since different nodes have different weight vectors, as opposed to the traditional single task learning network [5], which is usually transformed into a consensus optimization problem framework [35]–[37]. Each node n has access to a sequence of private measurements $\{d_n(t), \mathbf{u}_n(t)\}_{t=1,2,\ldots}$, where $\mathbf{u}_n(t) \in \mathbb{R}^M$ is the input regressor at time t and $d_n(t) \in \mathbb{R}$ is the output observation at time t. The measurement data are private in the sense that node n has access only to its own measurement sequence but not others'. The data at node n are assumed to conform to a linear regression model with (slowly) time-variant weight vector $\widetilde{\mathbf{w}}_n(t)$:

$$d_n(t) = \mathbf{u}_n(t)^{\mathsf{T}} \widetilde{\mathbf{w}}_n(t) + e_n(t), \tag{1}$$

where $e_n(t)$ is the output measurement noise at time t. In multitask learning networks, the benefit of cooperation between nodes comes from the fact that neighboring nodes have *similar* weight vectors [16], where similarity is embodied by some specific distance measures. By incorporating terms promoting similarity between neighbors and enforcing cooperation in the network, an estimator may achieve potentially higher performance than its non-cooperative counterpart.

Moreover, many signals in practice are highly sparse, i.e., most entries in the signal are equal to zero, with examples encompassing image signals, audio signals, etc. The sparsity of signals is especially conspicuous in today's big data era because redundant data are collected as input features among which most are unrelated to the targeted output, leading to sparsity of the corresponding weight vectors. Furthermore, as per convention in adaptive algorithms [1], we assume that the weight vectors $\widetilde{\mathbf{w}}_n(t)$ vary with time very slowly. This suggests that past data are of great merit to estimate the current weight vector, which justifies the advantage of the RLS (studied in this paper) over the LMS (studied in most existing works on multitask estimation [16]–[18], [26]) in terms of convergence speed.

In all, we propose an RLS based estimator to track the unknown weight vectors $\{\widetilde{\mathbf{w}}_n(t)\}_{n=1,2,\ldots,N}$ while enforcing similarity between neighbors' weight vectors and sparsity of all weight vectors. The estimator at time T is the optimal solution of the following optimization problem:

$$\text{Minimize}_{\mathbf{w}_1,\ldots,\mathbf{w}_N} \sum_{n=1}^{N}\sum_{t=1}^{T}\lambda^{T-t}\left(d_n(t)-\mathbf{u}_n(t)^{\mathsf{T}}\mathbf{w}_n\right)^2 + \beta\sum_{n=1}^{N}\sum_{m\in\Omega_n}\|\mathbf{w}_n-\mathbf{w}_m\|_2^2+\gamma\sum_{n=1}^{N}\|\mathbf{w}_n\|_1, \quad (2)$$

where $0<\lambda<1, \beta>0, \gamma>0$ are the forgetting factor of the RLS algorithm, regularization coefficient for similarity between neighbors' weight vectors and regularization coefficient for sparsity, respectively. If $\beta=\infty$, then problem (2) enforces consensus of weight vectors across nodes, and thus degenerates to the sparse RLS problem in [14]. If the underlying network is disconnected, then (2) can be equivalently decomposed into parallel and independent subproblems, among which each subproblem is over one connected component of the original network and is still of the form (2). In this sense, one can just focus on solving (2) for connected networks only. In the following, as the disconnectedness does not hurt the algorithms proposed in this paper, to achieve a more unified analysis, we allow the network to be either connected or disconnected. Note that the measurement data $\{\mathbf{u}_n(t), d_n(t)\}$ arrive in a sequential manner, which necessitates an online (real time) algorithm to solve (2) due to the prohibitive computation and storage cost of offline methods. Further note that the private measurement data are distributed among network nodes. Thus, a distributed algorithm for (2) is imperative as centralized algorithms are vulnerable to link failures and can incur large communication costs, not to mention the privacy concerns of the private data. Therefore, we are aimed at finding *distributed online* algorithm for solving (2). In the following two sections, we propose two different distributed online algorithms with complementary merits in accuracy and computational complexity.

Remark 1: We remark that the multitask sparse RLS problem formulated in (2) is different from the multitask sparse learning problem in [38] for particle filtering, which explicitly enforces each particle to have the same sparsity pattern in the dictionary. To this end, the optimization problem of [38] penalizes the summation of some norm of vectors comprised of entries at the same position of each weight vector. This formulation couples every weight vector altogether and is not suitable for a decentralized solution which we pursue here. In contrast, our problem formulation (2) favors similar sparsity patterns of all weight vectors in a different way by using proximity regularization between neighbors so that the formulated problem remains amenable to a distributed treatment. In addition, we note that a related distributed online optimization problem with proximity constraints has been examined in [39] recently. However, the convergence of the saddle point algorithm proposed in [39] relies on several restrictive assumptions which do not hold in the generic adaptive signal processing problems and hence cannot be applied to (2) in this paper. For example, the action set is assumed to be bounded in [39] while the weight vectors in most adaptive signal processing scenarios can be any vectors in the entire Euclidean space. The cost functions in [39] are assumed to be Lipschitz continuous which are not satisfied by the quadratic cost functions widely used in signal processing.

III. The Decentralized Online ADMM

In this section, we propose an alternating direction method of multipliers (ADMM) based decentralized online algorithm for solving (2). It is further simplified so that its iteration consists of simple closed-form computations and each node only needs to store and update one $M\times M$ matrix and six M dimensional vectors. The convergence of the proposed ADMM algorithm is also analyzed. Before the development of the algorithm, we first present some rudimentary knowledge of ADMM in the following section.

A. Preliminaries of ADMM

ADMM is an optimization framework widely applied to various signal processing applications, including wireless communications [32], network resource allocation [31], power systems [33] and multi-agent coordination [34]. It enjoys fast convergence speed under mild technical conditions [40] and is especially suitable for the development of distributed algorithms [30], [41]. ADMM solves problems of the following form:

$$\text{Minimize}_{\mathbf{x},\mathbf{w}}\, f(\mathbf{x})+g(\mathbf{w}) \text{ s.t. } \mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{w}=\mathbf{c}, \quad (3)$$

where $\mathbf{A}\in\mathbb{R}^{p\times n}, \mathbf{B}\in\mathbb{R}^{p\times m}, \mathbf{c}\in\mathbb{R}^{p}$ are constants and $\mathbf{x}\in\mathbb{R}^{n}, \mathbf{w}\in\mathbb{R}^{m}$ are optimization variables. $f:\mathbb{R}^n\mapsto\mathbb{R}$ and $g:\mathbb{R}^m\mapsto\mathbb{R}$ are two convex functions. The augmented Lagrangian can be formed as:

$$\mathfrak{L}_\rho(\mathbf{x},\mathbf{w},\mathbf{y}) = f(\mathbf{x})+g(\mathbf{w})+\mathbf{y}^{\mathsf{T}}(\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{w}-\mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{w}-\mathbf{c}\|_2^2, \quad (4)$$

where $\mathbf{y}\in\mathbb{R}^p$ is the Lagrange multiplier and $\rho>0$ is some constant. The ADMM then iterates over the following three steps for $k\geq 0$ (the iteration index):

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}}\mathfrak{L}_\rho\left(\mathbf{x},\mathbf{w}^k,\mathbf{y}^k\right), \quad (5)$$

$$\mathbf{w}^{k+1} = \arg\min_{\mathbf{w}}\mathfrak{L}_\rho\left(\mathbf{x}^{k+1},\mathbf{w},\mathbf{y}^k\right), \quad (6)$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k+\rho\left(\mathbf{A}\mathbf{x}^{k+1}+\mathbf{B}\mathbf{w}^{k+1}-\mathbf{c}\right). \quad (7)$$

The ADMM is guaranteed to converge to the optimal point of (3) as long as f and g are convex [30], [41]. It is recently shown that global linear convergence can be ensured provided additional assumptions on problem (3) hold [40].

B. Development of Decentralized Online ADMM for (2)

To apply ADMM to (2), we first transform it to the form of (3). We introduce auxiliary variables $\mathbf{x}_n\in\mathbb{R}^M, n=1,\ldots,N$, and $\mathbf{v}_{n,i}\in\mathbb{R}^M, n=1,\ldots N, i=1,\ldots,|\Omega_n|$, where $|\cdot|$ denotes the cardinality of a set. Denote the index of the i-th neighbor of node n as $g(n,i)$. Thus, problem (2) can be equivalently transformed into the following problem:

$$\begin{aligned}&\text{Minimize}\\ &\sum_{n=1}^{N}\sum_{t=1}^{T}\lambda^{T-t}\left(d_n(t)-\mathbf{u}_n(t)^{\mathsf{T}}\mathbf{x}_n\right)^2\\ &+\beta\sum_{n=1}^{N}\left[|\Omega_n|\|\mathbf{x}_n\|_2^2-2\left(\sum_{i=1}^{|\Omega_n|}\mathbf{v}_{n,i}\right)^{\mathsf{T}}\mathbf{x}_n+\sum_{i=1}^{|\Omega_n|}\|\mathbf{v}_{n,i}\|_2^2\right]\\ &+\gamma\sum_{n=1}^{N}\|\mathbf{w}_n\|_1\\ &\text{s.t. } \mathbf{x}_n=\mathbf{w}_n, n=1,\dots,N,\\ &\qquad \mathbf{v}_{n,i}=\mathbf{w}_{g(n,i)}, n=1\dots,N, i=1\dots,|\Omega_n|, \end{aligned}\tag{8}$$

where the optimization variables are $\mathbf{w}_n, \mathbf{x}_n, \mathbf{v}_{n,i}, n=1,\dots,N, i=1,\dots,|\Omega_n|$. Note that optimization problem (8) is in the form of (3) (regarding $\mathbf{x}_n$'s and $\mathbf{v}_{n,i}$'s as the variable $\mathbf{x}$ in (3) and $\mathbf{w}_n$'s as the variable $\mathbf{z}$ in (3)). Thus, we can apply ADMM to problem (8). Introducing Lagrange multiplier $\mathbf{y}_n\in\mathbb{R}^M, \mathbf{z}_{n,i}\in\mathbb{R}^M$, we can form the augmented Lagrangian of (8) as follows:

$$\begin{aligned}&\mathfrak{L}_\rho(\{\mathbf{x}_n,\mathbf{v}_{n,i},\mathbf{w}_n,\mathbf{y}_n,\mathbf{z}_{n,i}\}_{n=1,\dots,N,i=1,\dots,|\Omega_n|})\\ &=\sum_{n=1}^{N}\sum_{t=1}^{T}\lambda^{T-t}\left(d_n(t)-\mathbf{u}_n(t)^{\mathsf{T}}\mathbf{x}_n\right)^2\\ &+\beta\sum_{n=1}^{N}\left[|\mathcal{N}_n|\|\mathbf{x}_n\|_2^2-2\left(\sum_{i=1}^{|\Omega_n|}\mathbf{v}_{n,i}\right)^{\mathsf{T}}\mathbf{x}_n+\sum_{i=1}^{|\Omega_n|}\|\mathbf{v}_{n,i}\|_2^2\right]\\ &+\gamma\sum_{n=1}^{N}\|\mathbf{w}_n\|_1+\sum_{n=1}^{N}\mathbf{y}_n^{\mathsf{T}}(\mathbf{x}_n-\mathbf{w}_n)\\ &+\sum_{n=1}^{N}\sum_{i=1}^{|\Omega_n|}\mathbf{z}_{n,i}^{\mathsf{T}}(\mathbf{v}_{n,i}-\mathbf{w}_{g_{n,i}})+\frac{\rho}{2}\sum_{n=1}^{N}\|\mathbf{x}_n-\mathbf{w}_n\|_2^2\\ &+\frac{\rho}{2}\sum_{n=1}^{N}\sum_{i=1}^{|\Omega_n|}\|\mathbf{v}_{n,i}-\mathbf{w}_{g(n,i)}\|_2^2\end{aligned}\tag{9}$$

In the following, for ease of notation, we use $\mathbf{x}$ to represent all the $\{\mathbf{x}_n\}$ and similarly for $\mathbf{v},\mathbf{w},\mathbf{y},\mathbf{z}$. We apply the ADMM updates (5), (6) and (7) to problem (8) as follows:

$$\left\{\mathbf{x}^{k+1},\mathbf{v}^{k+1}\right\}=\arg\min_{\mathbf{x},\mathbf{v}}\mathfrak{L}_\rho\left(\mathbf{x},\mathbf{v},\mathbf{w}^k,\mathbf{y}^k,\mathbf{z}^k\right),\tag{10}$$

$$\mathbf{w}^{k+1}=\arg\min_{\mathbf{w}}\mathfrak{L}_\rho\left(\mathbf{x}^{k+1},\mathbf{v}^{k+1},\mathbf{w},\mathbf{y}^k,\mathbf{z}^k\right),\tag{11}$$

$$\mathbf{y}_n^{k+1}=\mathbf{y}_n^k+\rho\left(\mathbf{x}_n^{k+1}-\mathbf{w}_n^{k+1}\right),\tag{12}$$

$$\mathbf{z}_{n,i}^{k+1}=\mathbf{z}_{n,i}^k+\rho\left(\mathbf{v}_{n,i}^{k+1}-\mathbf{w}_{g(n,i)}^{k+1}\right).\tag{13}$$

In the following, we detail how to implement the updates of the primal variables, i.e., (10) and (11), in a distributed and online fashion.

1) Updating $\mathbf{x}$ *and* $\mathbf{v}$*:* The update of $\mathbf{x}$ and $\mathbf{v}$ in (10) can be decomposed across nodes. For each node n, the subproblem is:

$$\left\{\mathbf{x}_n^{k+1},\left\{\mathbf{v}_{n,i}^{k+1}\right\}_{i=1,\dots,|\Omega_n|}\right\}=\arg\min_{\mathbf{x}_n,\{\mathbf{v}_{n,i}\}_{i=1,\dots,|\Omega_n|}}J_n^k(T),\tag{14}$$

in which the objective function $J_n^k(T)$ is defined as:

$$\begin{aligned}J_n^k(T)&=\sum_{t=1}^{T}\lambda^{T-t}\left(d_n(t)-\mathbf{u}_n(t)^{\mathsf{T}}\mathbf{x}_n\right)^2\\ &+\beta\left[|\Omega_n|\|\mathbf{x}_n\|_2^2-2\left(\sum_{i=1}^{|\Omega_n|}\mathbf{v}_{n,i}\right)^{\mathsf{T}}\mathbf{x}_n+\sum_{i=1}^{|\Omega_n|}\|\mathbf{v}_{n,i}\|_2^2\right]\\ &+\mathbf{y}_n^{k\mathsf{T}}\mathbf{x}_n+\sum_{i=1}^{|\Omega_n|}\mathbf{z}_{n,i}^{k\mathsf{T}}\mathbf{v}_{n,i}+\frac{\rho}{2}\left\|\mathbf{x}_n-\mathbf{w}_n^k\right\|_2^2\\ &+\frac{\rho}{2}\sum_{i=1}^{|\Omega_n|}\left\|\mathbf{v}_{n,i}-\mathbf{w}_{g(n,i)}^k\right\|_2^2.\end{aligned}\tag{15}$$

Define the data dependent input correlation matrix and input-output cross correlation vector of node n at time T to be:

$$\mathbf{R}_n(T)=\sum_{t=1}^{T}\lambda^{T-t}\mathbf{u}_n(t)\mathbf{u}_n(t)^{\mathsf{T}},\tag{16}$$

$$\mathbf{p}_n(T)=\sum_{t=1}^{T}\lambda^{T-t}d_n(t)\mathbf{u}_n(t).\tag{17}$$

Note that $J_n^k(T)$ is a convex quadratic function. Hence, the necessary and sufficient condition for optimality of problem (14) is that the gradient of $J_n^k(T)$ vanishes. The gradient of $J_n^k(T)$ with respect to $\mathbf{x}_n$ and $\mathbf{v}_{n,i}$ can be computed as follows:

$$\begin{aligned}\nabla_{\mathbf{x}_n}J_n^k(T)&=(2\mathbf{R}_n(T)+2\beta|\Omega_n|\mathbf{I}+\rho\mathbf{I})\mathbf{x}_n-2\beta\sum_{i=1}^{|\Omega_n|}\mathbf{v}_{n,i}\\ &\qquad-2\mathbf{p}_n(T)+\mathbf{y}_n^k-\rho\mathbf{w}_n^k,\end{aligned}\tag{18}$$

$$\nabla_{\mathbf{v}_{n,i}}J_n^k(T)=-2\beta\mathbf{x}_n+(2\beta+\rho)\mathbf{v}_{n,i}+\mathbf{z}_{n,i}^k-\rho\mathbf{w}_{g(n,i)}^k.\tag{19}$$

Letting the gradients with respect to $\mathbf{x}_n$ and $\mathbf{v}_{n,i}$ be zero, we rewrite the update in (14) as (20) shown at the bottom of the next page. To inverse the matrix in (20), we need to use the following matrix inversion lemma.

Lemma 1: For arbitrary matrices $\mathbf{A}\in\mathbb{R}^{m\times m},\mathbf{B}\in\mathbb{R}^{m\times n},\mathbf{C}\in\mathbb{R}^{n\times m},\mathbf{D}\in\mathbb{R}^{n\times n}$ such that all the matrix inversions at the R.H.S. of (21) exist, the (21) holds. (21) shown at the bottom of the next page.

Define a new matrix:

$$\mathbf{F}_n(T)=\left[2\mathbf{R}_n(T)+\left(\rho+\frac{2\beta\rho|\Omega_n|}{2\beta+\rho}\right)\mathbf{I}\right]^{-1}.\tag{22}$$

By invoking the matrix inversion lemma (21), we can solve for the update (20) in closed form:

$$\mathbf{x}_n^{k+1} = \mathbf{F}_n(T)\left(2\mathbf{p}_n(T) - \mathbf{y}_n^k + \rho\mathbf{w}_n^k\right) + \frac{2\beta}{2\beta+\rho}\mathbf{F}_n(T)\sum_{i=1}^{|\Omega_n|}\left(-\mathbf{z}_{n,i}^k + \rho\mathbf{w}_{g(n,i)}^k\right), \tag{23}$$

$$\mathbf{v}_{n,i}^{k+1} = \frac{2\beta}{2\beta+\rho}\mathbf{F}_n(T)\left(2\mathbf{p}_n(T) - \mathbf{y}_n^k + \rho\mathbf{w}_n^k\right) + \frac{1}{2\beta+\rho}\left(-\mathbf{z}_{n,i}^k + \rho\mathbf{w}_{g(n,i)}^k\right) + \left(\frac{2\beta}{2\beta+\rho}\right)^2\mathbf{F}_n(T)\sum_{j=1}^{|\Omega_n|}\left(-\mathbf{z}_{n,j}^k + \rho\mathbf{w}_{g(n,j)}^k\right). \tag{24}$$

2) Updating $\mathbf{w}$*:* We note that the update for $\mathbf{w}$ in (11) can be decomposed not only across nodes but also across entries of the vector $\mathbf{w}_n$. For each node n, the l-th entry of $\mathbf{w}_n$ can be updated as follows:

$$w_n^{k+1}(l) = \arg\min_{w_n(l)}\left\{\gamma|w_n(l)| - y_n^k(l)w_n(l) - \left(\sum_{g(m,i)=n} z_{m,i}^k(l)\right)w_n(l) + \frac{\rho}{2}\left[w_n(l) - x_n^{k+1}(l)\right]^2 + \frac{\rho}{2}\sum_{g(m,i)=n}\left[w_n(l) - v_{m,i}^{k+1}(l)\right]^2\right\} \tag{25}$$

$$= \arg\min_{w_n(l)}\left\{\gamma|w_n(l)| + \frac{\rho}{2}(1+|\Omega_n|)\left[w_n(l) - \frac{1}{\rho(1+|\Omega_n|)}\left(y_n^k(l) + \rho x_n^{k+1}(l) + \sum_{g(m,i)=n}\left(z_{m,i}^k(l) + v_{m,i}^{k+1}(l)\right)\right)\right]^2\right\} \tag{26}$$

$$= \mathcal{S}_{\frac{\gamma}{\rho(1+|\Omega_n|)}}\left(\frac{1}{\rho(1+|\Omega_n|)}\left(y_n^k(l) + \rho x_n^{k+1}(l) + \sum_{g(m,i)=n}\left(z_{m,i}^k(l) + v_{m,i}^{k+1}(l)\right)\right)\right), \tag{27}$$

where the soft-threshold function $\mathcal{S}$ is defined for $a \in \mathbb{R}, \kappa > 0$ as follows:

$$\mathcal{S}_\kappa(a) = \begin{cases} a - \kappa, & \text{if} a > \kappa, \\ 0, & \text{if} |a| \le \kappa, \\ a + \kappa, & \text{if } a < -\kappa. \end{cases} \tag{28}$$

In (27), we have made use of the following fact.

Lemma 2: For any $\lambda > 0, \rho > 0, v \in \mathbb{R}$, we have:

$$\mathcal{S}_{\frac{\lambda}{\rho}}(v) = \arg\min_x\left(\lambda|x| + \frac{\rho}{2}(x-v)^2\right). \tag{29}$$

Once we extend the definition of $\mathcal{S}$ to vectors in a entrywise way, we can write the update for $\mathbf{w}_n$ compactly as:

$$\mathbf{w}_n^{k+1} = \mathcal{S}_{\frac{\gamma}{\rho(1+|\Omega_n|)}}\left(\frac{1}{\rho(1+|\Omega_n|)}\left(\mathbf{y}_n^k + \rho\mathbf{x}_n^{k+1} + \sum_{g(m,i)=n}\left(\mathbf{z}_{m,i}^k + \mathbf{v}_{m,i}^{k+1}\right)\right)\right). \tag{30}$$

3) Online Algorithm with Varying T*:* So far, the derived ADMM algorithm is only suitable for one particular time shot T. Since it takes iterations for ADMM to converge to the optimal point, for each time T, we ought to run multiple rounds of ADMM iterations $k = 1, \ldots, K$ for some sufficiently large K. After the ADMM has converged for this particular time T, we update the data related quantities ($\mathbf{R}_n(T)$ and $\mathbf{p}_n(T)$) and move to the next time slot. However, since the underlying weight vectors are varying across time (i.e., the underlying linear system is non-stationary), it is meaningless to estimate the weight vectors very accurately for every time slot. Thus, in the following, we choose $K = 1$, i.e., only one iteration of ADMM update is executed in each time slot. This is inspired by many existing adaptive algorithms such as the LMS algorithm, where only one step of gradient descent is performed at each time slot [1]. As such, we replace k with $T-1$ in the previously derived updates (23), (24), (30) and get updates that are suitable for varying time T:

$$\mathbf{x}_n(T) = \mathbf{F}_n(T)\left(2\mathbf{p}_n(T) - \mathbf{y}_n(T-1) + \rho\mathbf{w}_n(T-1)\right) + \frac{2\beta}{2\beta+\rho}\mathbf{F}_n(T)\sum_{i=1}^{|\Omega_n|}\left(-\mathbf{z}_{n,i}(T-1) + \rho\mathbf{w}_{g(n,i)}(T-1)\right), \tag{31}$$

$$\left[\begin{array}{c:cccc} 2\mathbf{R}_n(T) + 2\beta|\Omega_n|\mathbf{I} + \rho\mathbf{I} & -2\beta\mathbf{I} & -2\beta\mathbf{I} & \cdots & -2\beta\mathbf{I} \\ \hdashline -2\beta\mathbf{I} & (2\beta+\rho)\mathbf{I} & & & \\ -2\beta\mathbf{I} & & (2\beta+\rho)\mathbf{I} & & 0 \\ \vdots & & & \ddots & \\ -2\beta\mathbf{I} & & 0 & & (2\beta+\rho)\mathbf{I} \end{array}\right]\left[\begin{array}{c} \mathbf{x}_n^{k+1} \\ \hdashline \mathbf{v}_{n,1}^{k+1} \\ \mathbf{v}_{n,2}^{k+1} \\ \vdots \\ \mathbf{v}_{n,|\Omega_n|}^{k+1} \end{array}\right] = \left[\begin{array}{c} 2\mathbf{p}_n(T) - \mathbf{y}_n^k + \rho\mathbf{w}_n^k \\ \hdashline -\mathbf{z}_{n,1}^k + \rho\mathbf{w}_{g(n,1)}^k \\ -\mathbf{z}_{n,2}^k + \rho\mathbf{w}_{g(n,2)}^k \\ \vdots \\ -\mathbf{z}_{n,|\Omega_n|}^k + \rho\mathbf{w}_{g(n,|\Omega_n|)}^k \end{array}\right] \tag{20}$$

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1} & -\left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1}\mathbf{B}\mathbf{D}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}\left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{C}\left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1}\mathbf{B}\mathbf{D}^{-1} \end{bmatrix}. \tag{21}$$

$$\mathbf{v}_{n,i}(T) = \frac{2\beta}{2\beta+\rho}\mathbf{F}_n(T)(2\mathbf{p}_n(T) - \mathbf{y}_n(T-1) + \rho\mathbf{w}_n(T-1)) + \frac{1}{2\beta+\rho}\left(-\mathbf{z}_{n,i}(T-1) + \rho\mathbf{w}_{g(n,i)}(T-1)\right) + \left(\frac{2\beta}{2\beta+\rho}\right)^2 \mathbf{F}_n(T)\sum_{j=1}^{|\Omega_n|}\left(-\mathbf{z}_{n,j}(T-1) + \rho\mathbf{w}_{g(n,j)}(T-1)\right), \tag{32}$$

$$\mathbf{w}_n(T) = \mathcal{S}_{\frac{\gamma}{\rho(1+|\Omega_n|)}}\left(\frac{1}{\rho(1+|\Omega_n|)}\left(\mathbf{y}_n(T-1) + \rho\mathbf{x}_n(T) + \sum_{g(m,i)=n}(\mathbf{z}_{m,i}(T-1) + \mathbf{v}_{m,i}(T))\right)\right). \tag{33}$$

Moreover, the updates (12) and (13) for dual variables can be rewritten as:

$$\mathbf{y}_n(T) = \mathbf{y}_n(T-1) + \rho\left(\mathbf{x}_n(T) - \mathbf{w}_n(T)\right), \tag{34}$$

$$\mathbf{z}_{n,i}(T) = \mathbf{z}_{n,i}(T-1) + \rho\left(\mathbf{v}_{n,i}(T) - \mathbf{w}_{g(n,i)}(T)\right). \tag{35}$$

The correlation matrices and cross-correlation vectors can be updated as follows:

$$\mathbf{R}_n(T+1) = \lambda\mathbf{R}_n(T) + \mathbf{u}_n(T+1)\mathbf{u}_n(T+1)^{\mathsf{T}}, \tag{36}$$

$$\mathbf{p}_n(T+1) = \lambda\mathbf{p}_n(T) + d_n(T+1)\mathbf{u}_n(T+1). \tag{37}$$

And $\mathbf{F}_n(T)$ is computed according to (22).

Remark 2: The computation of $\mathbf{F}_n(T)$ in (22) necessitates inversion of an $M \times M$ matrix, which incurs a computational complexity of $\mathcal{O}(M^3)$ unless special structure is present. For the special case of $\lambda = 1$ (which is suitable for time-invariant weight vectors), this burden can be alleviated as follows. According to (22), (36) and the condition that $\lambda = 1$, we have:

$$\begin{aligned}\mathbf{F}_n(T) &= \left[2\left(\mathbf{R}_n(T-1) + \mathbf{u}_n(T)\mathbf{u}_n(T)^{\mathsf{T}}\right) + \left(\rho + \frac{2\beta\rho|\Omega_n|}{2\beta+\rho}\right)\mathbf{I}\right]^{-1} \\ &= \left[\mathbf{F}_n^{-1}(T-1) + 2\mathbf{u}_n(T)\mathbf{u}_n(T)^{\mathsf{T}}\right]^{-1} \\ &= \mathbf{F}_n(T-1) - \frac{\mathbf{F}_n(T-1)\mathbf{u}_n(T)\mathbf{u}_n(T)^{\mathsf{T}}\mathbf{F}_n(T-1)}{\frac{1}{2} + \mathbf{u}_n(T)^{\mathsf{T}}\mathbf{F}_n(T-1)\mathbf{u}_n(T)}.\end{aligned} \tag{38}$$

However, in the general case where $\lambda < 1$, the matrix inversion incurred by the computation of $\mathbf{F}_n(T)$ is inevitable, which is the most computationally intensive part of the proposed ADMM algorithm. In fact, when $\lambda < 1$, from (22), (36), we have:

$$\mathbf{F}_n(T) = \left[2\lambda\mathbf{R}_n(T-1) + \left(\rho + \frac{2\beta\rho|\Omega_n|}{2\beta+\rho}\right)\mathbf{I} + 2\mathbf{u}_n(T)\mathbf{u}_n(T)^{\mathsf{T}}\right]^{-1}. \tag{39}$$

As $\lambda < 1$, the sum of the first two terms inside the inversion of (39) is not equal to $\mathbf{F}_n(T-1)^{-1}$ (c.f. (22)). Therefore, the update of (38) does not hold any more. Note that the matrix inversion in (22) is different from the one involved in classical centralized RLS [1]. In classical RLS, we are interested in the inversion of the input data correlation matrix $\mathbf{R}_n(T)$ only, which can be computed recursively via a rank one update (c.f. (36)). In such a case, even when $\lambda < 1$, we can still apply matrix inversion lemma to avoid explicit computation of matrix inversion. In contrast, in the matrix inversion (22) that we encounter in this paper, a scalar multiple of $\mathbf{I}$ appears inside the matrix to be inverted so that the rank one update no longer holds for the matrix to be inverted. This prohibits application of the matrix inversion lemma unless in the special case of $\lambda = 1$.

4) Simplification of the ADMM Updates: So far, the ADMM updates involve primal variables $\{\mathbf{v}_{n,i}\}$ and dual variables $\{\mathbf{z}\}_{n,i}$. For each node n, $\{\mathbf{v}_{n,i}\}$ and $\{\mathbf{z}\}_{n,i}$ include $2|\Omega_n|$ M-dimensional vectors, which is costly to sustain in terms of communication and storage overhead, especially when the numbers of neighbors (degrees) are large. This motivates us to simplify the ADMM updates (31)–(37) so that the number of vectors at each node is independent of its degree. To this end, we first define the following auxiliary variables:

$$\underline{\mathbf{z}}_n(T) = \sum_{i=1}^{|\Omega_n|}\mathbf{z}_{n,i}(T), \tag{40}$$

$$\overline{\mathbf{z}}_n(T) = \sum_{g(m,i)=n}\mathbf{z}_{m,i}(T), \tag{41}$$

$$\underline{\mathbf{v}}_n(T) = \sum_{i=1}^{|\Omega_n|}\mathbf{v}_{n,i}(T), \tag{42}$$

$$\overline{\mathbf{v}}_n(T) = \sum_{g(m,i)=n}\mathbf{v}_{m,i}(T), \tag{43}$$

$$\overline{\mathbf{w}}_n(T) = \sum_{m\in\Omega_n}\mathbf{w}_m(T), \tag{44}$$

$$\boldsymbol{\eta}_n(T) = \mathbf{F}_n(T)(2\mathbf{p}_n(T) - \mathbf{y}_n(T-1) + \rho\mathbf{w}_n(T-1)), \tag{45}$$

$$\boldsymbol{\theta}_n(T) = \mathbf{F}_n(T)(-\underline{\mathbf{z}}_n(T-1) + \rho\overline{\mathbf{w}}_n(T-1)), \tag{46}$$

$$\overline{\boldsymbol{\eta}}_n(T) = \sum_{m\in\Omega_n}\boldsymbol{\eta}_m(T), \tag{47}$$

$$\overline{\boldsymbol{\theta}}_n(T) = \sum_{m\in\Omega_n}\boldsymbol{\theta}_m(T). \tag{48}$$

Thus, the update for $\mathbf{x}$ in (31) can be rewritten as:

$$\mathbf{x}_n(T) = \boldsymbol{\eta}_n(T) + \frac{2\beta}{2\beta+\rho}\boldsymbol{\theta}_n(T). \tag{49}$$

Using (32) yields the update for $\underline{\mathbf{v}}_n(T)$ and $\overline{\mathbf{v}}_n(T)$:

$$\underline{\mathbf{v}}_n(T) = \frac{2\beta|\Omega_n|}{2\beta+\rho}\boldsymbol{\eta}_n(T) + \left(\frac{2\beta}{2\beta+\rho}\right)^2|\Omega_n|\boldsymbol{\theta}_n(T) + \frac{1}{2\beta+\rho}(-\underline{\mathbf{z}}_n(T-1) + \rho\overline{\mathbf{w}}_n(T-1)). \tag{50}$$

$$\overline{\mathbf{v}}_n(T) = \frac{2\beta}{2\beta+\rho}\overline{\boldsymbol{\eta}}_n(T) + \left(\frac{2\beta}{2\beta+\rho}\right)^2\overline{\boldsymbol{\theta}}_n(T) + \frac{1}{2\beta+\rho}(-\overline{\mathbf{z}}_n(T-1) + \rho|\Omega_n|\mathbf{w}_n(T-1)). \tag{51}$$

The update for $\mathbf{w}_n(T)$ can be rewritten as:

$$\mathbf{w}_n(T) = \mathcal{S}_{\frac{\gamma}{\rho(1+|\Omega_n|)}}\left(\frac{1}{\rho(1+|\Omega_n|)}(\mathbf{y}_n(T-1) + \rho\mathbf{x}_n(T) + \overline{\mathbf{z}}_n(T-1) + \rho\overline{\mathbf{v}}_n(T))\right). \tag{52}$$

Similarly, from (35), we can spell out the updates for $\underline{\mathbf{z}}_n(T)$ and $\overline{\mathbf{z}}_n(T)$:

$$\underline{\mathbf{z}}_n(T) = \underline{\mathbf{z}}_n(T-1) + \rho(\underline{\mathbf{v}}_n(T) - \overline{\mathbf{w}}_n(T)), \tag{53}$$

$$\overline{\mathbf{z}}_n(T) = \overline{\mathbf{z}}_n(T-1) + \rho(\overline{\mathbf{v}}_n(T) - |\Omega_n|\mathbf{w}_n(T)). \tag{54}$$

Now, we are ready to formally present the proposed decentralized online ADMM algorithm for solving (2), which is summarized in Algorithm 1. Notice that the algorithm is completely distributed: each node only needs to communicate with its neighbors. It is also online (real-time): each node only needs to store and update one $M \times M$ matrix $\mathbf{R}_n(T)$ and six M dimensional vectors $\mathbf{p}_n(T), \mathbf{w}_n(T), \overline{\mathbf{w}}_n(T), \mathbf{y}_n(T), \underline{\mathbf{z}}_n(T), \overline{\mathbf{z}}_n(T)$. All other involved quantities in Algorithm 1 are intermediate and can be derived from these stored matrices and vectors. For each node n, the most computationally intensive part is the matrix inversion in (22), which needs to be done once in each time slot with computational complexity of $\mathcal{O}(M^3)$. The rest part of Algorithm 1 incurs computational complexity of $7M^2 + (38 + 3|\Omega_n|)M$ for each node n. Hence, in total, the computational complexity of each node n is $\mathcal{O}(M^3) + 7M^2 + (38 + 3|\Omega_n|)M = \mathcal{O}(M^3 + |\Omega_n|M)$. In most scenarios, M^2 is much larger than $|\Omega_n|$, i.e., the number of neighbors of node n. In such a case, the computational complexity of each node is $\mathcal{O}(M^3)$. Furthermore, the communication complexity of Algorithm 1 at each node is $3M$, as each node n needs to broadcast three M-dimensional vectors $\boldsymbol{\eta}_n(T), \boldsymbol{\theta}_n(T), \mathbf{w}_n(T)$ to its neighbors at each time T.

We note that the online versions of ADMM have been investigated in [42] for the centralized scenario and in [43], [44] for the decentralized scenario. The generic online algorithms proposed in these works, analogous to the classical static ADMM in [30], necessitate solving nonlinear optimization problems in each iteration and are unfavorable in many online applications due to the high computational overhead. In this paper, by exploiting the special structure of the multitask sparse RLS problem in (2) and reformulating it into an appropriate equivalent form (8), the proposed online distributed ADMM algorithm only requires direct closed-form computations in each iteration, which reduces the computational complexity. To further lower the computational overhead, we simplify the ADMM updates so that each node only needs to maintain and update one $M \times M$ matrix and six M-dimensional vectors regardless of its degree (number of neighbors).

Algorithm 1: The proposed decentralized online ADMM algorithm.

```
Inputs:
    Measurement data stream at each node $\{\mathbf{u}_n(t), d_n(t)\}$,
    $n = 1, 2, \ldots, N$, $t = 1, 2, \ldots$
Outputs:
    Estimates of the unknown weight vectors at each node
    $\{\mathbf{w}_n(T)\}$, $T = 1, 2, \ldots$
 1: Initialize $\mathbf{R}_n(0) = \mathbf{0}_{M\times M}$ and $\mathbf{p}_n(0) = \mathbf{w}_n(0) = \overline{\mathbf{w}}_n(0) = \mathbf{y}_n(0) = \underline{\mathbf{z}}_n(0) = \overline{\mathbf{z}}_n(0) = \mathbf{0}_M$. $T = 0$.
 2: Repeat:
 3:   $T \leftarrow T + 1$.
 4:   Each node $n$ updates its correlation matrix and cross-correlation vector once receiving the new data $\mathbf{u}_n(T)$, $d_n(T)$:
          $\mathbf{R}_n(T) = \lambda\mathbf{R}_n(T-1) + \mathbf{u}_n(T)\mathbf{u}_n(T)^{\mathsf{T}}$,   (55)
          $\mathbf{p}_n(T) = \lambda\mathbf{p}_n(T-1) + d_n(T)\mathbf{u}_n(T)$.   (56)
 5:   Each node $n$ computes $\mathbf{F}_n(T)$ according to (22).
 6:   Each node $n$ computes $\boldsymbol{\eta}_n(T)$ and $\boldsymbol{\theta}_n(T)$ according to (45) and (46) and then broadcasts its results to its neighbors.
 7:   Each node $n$ receives $\boldsymbol{\eta}_m(T)$ and $\boldsymbol{\theta}_m(T)$ from its neighbors $m \in \Omega_n$ and forms $\overline{\boldsymbol{\eta}}_n(T)$ and $\overline{\boldsymbol{\theta}}_n(T)$ based on (47) and (48).
 8:   Each node $n$ computes $\mathbf{x}_n(T), \underline{\mathbf{v}}_n(T), \overline{\mathbf{v}}_n(T)$ according to (49), (50) and (51), respectively.
 9:   Each node $n$ computes $\mathbf{w}_n(T)$ based on (52) and broadcasts the result to its neighbors.
10:   Each node $n$ receives $\mathbf{w}_m(T)$ from its neighbors $m \in \Omega_n$ and forms $\overline{\mathbf{w}}_n(T)$ according to (44).
11:   Each node $n$ updates $\mathbf{y}_n(T), \underline{\mathbf{z}}_n(T), \overline{\mathbf{z}}_n(T)$ according to (34), (53) and (54).
```

C. Convergence of the Algorithm 1

In this section, we analyze the convergence of Algorithm 1. To facilitate analysis, we make the following assumptions.

Assumption 1: The true weight vector $\widetilde{\mathbf{w}}_n$ is time-invariant, i.e., the linear regression data model is $d_n(t) = \mathbf{u}_n(t)^{\mathsf{T}}\widetilde{\mathbf{w}}_n + e_n(t)$.

Assumption 2: For each node n, the input process $\{\mathbf{u}_n(t)\}_{t=1,2,\ldots}$ is independent across time with time-invariant correlation matrix $\mathbf{R}_n = \mathbb{E}\left[\mathbf{u}_n(t)\mathbf{u}_n(t)^{\mathsf{T}}\right]$.

Assumption 3: For each node n, the noise process $\{e_n(t)\}_{t=1,2,\ldots}$ has zero mean, i.e., $\mathbb{E}[e_n(t)] = 0$ and is independent across time and independent from the input process $\{\mathbf{u}_n(t)\}$.

Note that all of these assumptions are standard when analyzing the performance of adaptive algorithms in the literature [1]. Recall that λ is assumed to be strictly smaller than 1 in the problem formulation in Section II. Later, this will be used to establish the convergence of $\mathbf{R}_n(T)$ and $\mathbf{p}_n(T)$ (c.f. (16) and (17)) by invoking the strong law of large numbers. For a positive definite matrix $\boldsymbol{\Theta}$ and a vector $\boldsymbol{\omega}$, we define $\|\boldsymbol{\omega}\|_{\boldsymbol{\Theta}} := \sqrt{\boldsymbol{\omega}^{\mathsf{T}}\boldsymbol{\Theta}\boldsymbol{\omega}}$. Define $\mathbf{w}(T) = \left[\mathbf{w}_1(T)^{\mathsf{T}}, \ldots, \mathbf{w}_N(T)^{\mathsf{T}}\right]^{\mathsf{T}}$. Then, we have the following result regarding the convergence of Algorithm 1.

Theorem 1: Suppose Assumptions 1–3 hold. Then, the output of Algorithm 1, i.e., $\mathbf{w}(T)$, converges to the optimal point of the following optimization problem almost surely:

$$Minimize_{\mathbf{w}} \frac{1}{1-\lambda}\sum_{n=1}^{N}\|\mathbf{w}_n - \widetilde{\mathbf{w}}_n\|^2_{\mathbf{R}_n} + \beta\sum_{n=1}^{N}\sum_{m\in\Omega_n}\|\mathbf{w}_m - \mathbf{w}_n\|_2^2 + \gamma\sum_{n=1}^{N}\|\mathbf{w}_n\|_1. \tag{57}$$

Proof: The proof is presented in Appendix A. ■

Algorithm 2: The proposed decentralized online subgradient algorithm.

```
Inputs:
    Measurement data stream at each node $\{\mathbf{u}_n(t), d_n(t)\}$,
    $n = 1, 2, \ldots, N, t = 1, 2, \ldots$
Outputs:
    Estimates of the unknown weight vectors at each node
    $\{\mathbf{w}_n(T)\}, T = 1, 2, \ldots$
1: Initialize $\mathbf{R}_n(0) = \mathbf{0}_{M\times M}$, $\mathbf{p}_n(0) = \mathbf{w}_n(0) = \overline{\mathbf{w}}_n(0) = \mathbf{0}_M, T = 0$.
2: Repeat:
3: $T \leftarrow T + 1$.
4: Each node $n$ updates its correlation matrix and cross-correlation vector once receiving the new data $\mathbf{u}_n(T)$, $d_n(T)$:
       $\mathbf{R}_n(T) = \lambda\mathbf{R}_n(T-1) + \mathbf{u}_n(T)\mathbf{u}_n(T)^{\mathsf{T}}$,   (58)
       $\mathbf{p}_n(T) = \lambda\mathbf{p}_n(T-1) + d_n(T)\mathbf{u}_n(T)$.   (59)
5: Each node $n$ updates $\mathbf{w}_n(T)$:
       $\mathbf{w}_n(T) = \mathbf{w}_n(T-1) - \alpha\Big[2\mathbf{R}_n(T)\mathbf{w}_n(T-1) - 2\mathbf{p}_n(T) + 4\beta(|\Omega_n|\mathbf{w}_n(T-1) - \overline{\mathbf{w}}_n(T-1)) + \gamma\,\mathrm{sgn}(\mathbf{w}_n(T-1))\Big]$.   (60)
6: Each node $n$ broadcasts its $\mathbf{w}_n(T)$ to its neighbors.
7: Each node $n$ receives $\mathbf{w}_m(T)$ from its neighbors $m \in \Omega_n$ and forms $\overline{\mathbf{w}}_n(T)$ as follows:
       $\overline{\mathbf{w}}_n(T) = \sum_{m\in\Omega_n}\mathbf{w}_m(T)$.   (61)
```

IV. The Decentralized Online Subgradient Method

The implementation of the proposed Algorithm 1 necessitates an inversion of an $M \times M$ matrix at each time and each node, which may not be suitable for nodes with low computational capability. In fact, a relatively high computational overhead is a general drawback of dual domain methods (e.g., ADMM) in optimization theory [36]. On the contrary, primal domain methods such as gradient descent method, though having relatively slow convergence speed, enjoys low computational complexity [45]. As such, in this section, we present a distributed online subgradient method for problem (2) to trade off convergence speed and accuracy for low computational complexity.

A. Development of the Decentralized Online Subgradient Method

Recall the optimization problem at time T, i.e., problem (2). Denote the objective function of (2) as $H_T(\mathbf{w})$. We derive the subdifferential (the set of subgradients [46]) of H_T at $\mathbf{w}$ in (62), shown at the bottom of this page, where the sign (set) function is defined as:

$$\mathrm{sgn}(x) = \begin{cases} 1, & \text{if } x > 0, \\ -1, & \text{if } x < 0, \\ [-1,1], & \text{if } x = 0. \end{cases} \tag{63}$$

The extension of the sgn function to vectors is entrywise. The subgradient method is to simply use the iteration $\mathbf{w}(T) = \mathbf{w}(T-1) - \alpha\mathbf{g}$, where $\mathbf{g} \in \partial H_T(\mathbf{w}(T-1))$ is any subgradient of H_T at $\mathbf{w}(T-1)$ and $\alpha > 0$ is the step size [46]. This naturally leads to the following decentralized online update:

$$\mathbf{w}_n(T) = \mathbf{w}_n(T-1) - \alpha\Big[2\mathbf{R}_n(T)\mathbf{w}_n(T-1) - 2\mathbf{p}_n(T) + 4\beta\sum_{m\in\Omega_n}(\mathbf{w}_n(T-1) - \mathbf{w}_m(T-1)) + \gamma\,\mathrm{sgn}(\mathbf{w}_n(T-1))\Big], \tag{64}$$

where $\mathrm{sgn}(0)$ is any number within the interval $[-1,1]$[1]. By introducing an auxiliary variable $\overline{\mathbf{w}}_n(T)$, we propose the decentralized online subgradient method for (2) in Algorithm 2. We observe that Algorithm 2 is completely decentralized as every node only communicates with its neighbors. It is also online since each node only needs to store and update one $M \times M$ matrix and three M dimensional vectors. More importantly, Algorithm 2 is free of any matrix inversion, which is a major computational burden of Algorithm 1. The computational complexity of each node n in each time slot is $4M^2 + (12 + |\Omega_n|)M = \mathcal{O}\left(M^2 + |\Omega_n|M\right)$. When M is larger than $|\Omega_n|$ (which is usually the case), the computational complexity becomes $\mathcal{O}\left(M^2\right)$.

[1]There is a standard abuse of notation for the sgn function: in (62) and (63), $\mathrm{sgn}(0)$ is defined to be the interval $[-1,1]$ while in (64), $\mathrm{sgn}(0)$ is defined to be any arbitrary number within $[-1,1]$. In the following, the latter definition will be used.

$$\partial H_T(\mathbf{w}) = \begin{bmatrix} 2\mathbf{R}_1(T)\mathbf{w}_1 - 2\mathbf{p}_1(T) + 2\beta\left(2|\Omega_1|\mathbf{w}_1 - 2\sum_{m\in\Omega_1}\mathbf{w}_m\right) + \gamma\,\mathrm{sgn}(\mathbf{w}_1) \\ \vdots \\ 2\mathbf{R}_N(T)\mathbf{w}_N - 2\mathbf{p}_N(T) + 2\beta\left(2|\Omega_N|\mathbf{w}_N - 2\sum_{m\in\Omega_N}\mathbf{w}_m\right) + \gamma\,\mathrm{sgn}(\mathbf{w}_N) \end{bmatrix}, \tag{62}$$

In addition, the communication complexity of each node n at each time is M as it only needs to broadcast one M-dimensional vector $\mathbf{w}_n(T)$ to its neighbors at each time T. From these complexity analyses, we can see that the computation and communication complexity of Algorithm 2 is smaller than those of Algorithm 1.

B. Convergence Analysis of Algorithm 2

In this section, we analyze the convergence behavior of Algorithm 2. Specifically, we will establish the mean square stability of Algorithm 2 by upper bounding the mean square deviation (MSD) and the excess mean square error (EMSE) under Assumptions 1–3 and the assumption that the step size α is small enough. Denote the stacked vector of $\{\mathbf{w}_n(T)\}_{n=1}^N$ as $\mathbf{w}(T) \in \mathbb{R}^{NM}$ and the stacked vector of $\{\widetilde{\mathbf{w}}_n\}_{n=1}^N$ as $\widetilde{\mathbf{w}} \in \mathbb{R}^{NM}$. Define the block Laplacian matrix $\mathbf{L} \in \mathbb{R}^{NM\times NM}$:

$$\mathbf{L}_{ij} = \begin{cases} |\Omega_i|\mathbf{I}, \text{ if } i = j, \\ -\mathbf{I}, \text{ if } j \in \Omega_i, \\ \mathbf{0}, \text{ o.w.}, \end{cases} \tag{65}$$

where $\mathbf{L}_{ij}$ is the (i,j)-th $M \times M$ block of matrix $\mathbf{L}$. From basic spectral graph theory, we know that $\mathbf{L}$ is positive semidefinite. Define a block diagonal matrix $\mathbf{R} = \text{diag}(\mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_n)$. Denote the smallest eigenvalue of $\mathbf{R}$, the largest eigenvalue of $\mathbf{R}$ and the largest eigenvalue of $\mathbf{L}$ as μ_1, μ_2, μ_3, respectively. We have $\mu_2 \geq \mu_1 > 0$ (each $\mathbf{R}_n$ is positive definite) and $\mu_3 \geq 0$. Furthermore, we define two non-negative constants δ and Γ as:

$$\delta = \max\left\{\left|1 - \frac{2\alpha\mu_2}{1-\lambda} - 4\alpha\beta\mu_3\right|, \left|1 - \frac{2\alpha\mu_1}{1-\lambda}\right|\right\}, \tag{66}$$

$$\Gamma = 4\alpha\beta\mu_3\|\widetilde{\mathbf{w}}\|_2 + \alpha\gamma\sqrt{MN}. \tag{67}$$

Then, we have the following results regarding the mean square stability of Algorithm 2.

Theorem 2: Suppose Assumptions 1–3 hold. Further assume that the step size α satisfies:

$$\alpha < \frac{1}{\frac{\mu_2}{1-\lambda} + 2\beta\mu_3}. \tag{68}$$

Then the constant δ is strictly smaller than 1 and we have the following bounds for the steady state mean square deviation (MSD) and the steady state excess mean square error (EMSE):

$$MSD := \limsup_{T\to\infty} \mathbb{E}\left[\|\mathbf{w}(T) - \widetilde{\mathbf{w}}\|_2^2\right] \leq \frac{\Gamma^2}{(1-\delta)^2}, \tag{69}$$

$$\begin{aligned} &EMSE \\ &:= \limsup_{T\to\infty} \mathbb{E}\left[\sum_{n=1}^N \left(\mathbf{u}_n(T)^\mathsf{T}\mathbf{w}_n(T-1) - \mathbf{u}_n(T)^\mathsf{T}\widetilde{\mathbf{w}}_n\right)^2\right] \\ &\leq \left[\max_{n=1,\dots,N} \text{tr}(\mathbf{R}_n)\right] \frac{\Gamma^2}{(1-\delta)^2}. \end{aligned} \tag{70}$$

Proof: The proof is presented in Appendix B. ■

Remark 3: The assumption (68) requires the step size α to be small enough, which is a standard requirement for the convergence of both centralized and decentralized adaptive filters in general [1], [5]. In practice, the choice of α should be neither too small nor too large. If α is too large and (68) is violated, the performance of Algorithm 2 degrades or may even be unstable. If α is too small, Algorithm 2 still converges but the convergence speed may be very slow. The requirement (68) suggests that a reasonable value of α may depend on lots of factors, including the data/signal statistics, the network topology and the problem parameters.

C. The Decentralized Online Proximal Gradient Method

An optimization algorithm closely related to the subgradient method is the proximal gradient method [47], which relies on the proximal operator defined as:

$$\text{prox}_f(\mathbf{v}) = \arg\min_{\mathbf{x}} \left(f(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{v}\|_2^2\right), \tag{71}$$

in which $\mathbf{v} \in \mathbb{R}^k$ and $f : \mathbb{R}^k \mapsto \mathbb{R}$ is some convex function. Suppose $\phi : \mathbb{R}^k \mapsto \mathbb{R}$ and $\psi : \mathbb{R}^k \mapsto \mathbb{R}$ are two convex functions, among which ϕ is differentiable. Consider the following convex optimization problem:

$$\text{Minimize}_{\mathbf{x}}\ \phi(\mathbf{x}) + \psi(\mathbf{x}). \tag{72}$$

Then, the proximal gradient method for solving problem (72) is:

$$\mathbf{x}^{i+1} = \text{prox}_{\alpha\psi}\left(\mathbf{x}^i - \alpha\nabla\phi\left(\mathbf{x}^i\right)\right), \tag{73}$$

where i is the iteration index and $\alpha > 0$ is the step size. Under certain technical assumptions (e.g., Lipschitz continuity of $\nabla\phi$), convergence of the proximal gradient method is guaranteed [48]. We can apply the proximal gradient method to problem (2) by letting ϕ and ψ be the first two summation terms and the last summation term of (2), respectively. Specifically, the proximal gradient update at node n is:

$$\begin{aligned} \mathbf{w}_n(T) = \mathcal{S}_{\alpha\gamma}\big(&\mathbf{w}_n(T-1) - \alpha[2\mathbf{R}_n(T)\mathbf{w}_n(T-1) \\ &- 2\mathbf{p}_n(T) + 4\beta(|\Omega_n|\mathbf{w}_n(T-1) - \overline{\mathbf{w}}_n(T-1))]\big), \end{aligned} \tag{74}$$

where $\mathcal{S}$ is the soft thresholding function defined in (28) and $\overline{\mathbf{w}}_n(T) := \sum_{m\in\Omega_n} \mathbf{w}_m(T)$. The proximal gradient method specified in (74) is clearly online and distributed. It is indeed analogous to the subgradient method update in (60). The only difference is that, instead of using a sgn function in the descent direction in (60), the proximal gradient method applies a soft thresholding operator in (74). This similarity suggests that the two methods may have analogous empirical performance, which will be verified in Section V.

D. Extension to Clustered Multitask Networks

In this section, we extend Algorithm 2 to clustered multitask networks [16]. In clustered multitask networks, nodes belonging to the same cluster share the same weight vector and connected clusters have similar weight vectors (two clusters are connected if there is at least one edge linking a node from one cluster to a node in the other cluster). Specifically, all nodes are divided into Q non-overlapping clusters: $\{1, \dots, N\} = \cup_{q=1}^Q \mathcal{C}_q, \mathcal{C}_i \cap \mathcal{C}_j =$

$\emptyset, \forall i \neq j$. Denote the index of the cluster that node n belongs to as $h(n)$, i.e., $n \in \mathcal{C}_{h(n)}$. Then, the clustered multitask sparse RLS problem is:

$$\begin{aligned}\text{Minimize}_{\mathbf{w},\mathbf{w}^{\#}} & \sum_{n=1}^{N}\sum_{t=1}^{T} \lambda^{T-t}\left(d_n(t) - \mathbf{u}_n(t)^{\mathsf{T}}\mathbf{w}_n\right)^2 \\ & + \sum_{n=1}^{N} \sum_{m\in\Omega_n\setminus\mathcal{C}_{h(n)}} \beta_{nm}\|\mathbf{w}_m - \mathbf{w}_n\|_2^2 \\ & + \sum_{n=1}^{N}\gamma_n\|\mathbf{w}_n\|_1 \\ \text{s.t.} \quad & \mathbf{w}_n = \mathbf{w}_q^{\#}, \forall q = 1,\ldots,Q,\ n\in\mathcal{C}_q. \end{aligned} \tag{75}$$

The constraints of problem (75) enforce every node in a cluster $\mathcal{C}_q$ to share the same weight vector $\mathbf{w}_q^{\#}$. The second summation term in the objective function of (75) promotes proximity between the weight vectors of neighbors belonging to different clusters, which leads to similarity between the weight vectors of connected clusters. We note that in problem (75), the proximity regularization parameter $\beta_{nm} > 0$ is edge dependent and the sparsity regularization parameter $\gamma_n > 0$ is node dependent. This dependence can be exploited to capture the prior knowledge of the model, e.g., the extent of proximity between two connected clusters and the extent of sparsity of the weight vector of a cluster. To solve problem (75) in a distributed and online manner, we resort to the distributed subgradient method for consensus optimization [45] (as nodes within the same cluster needs to reach consensus) and propose a variant suitable for multitask online processing [16]. To this end, we first define a combination coefficient matrix $\mathbf{C} \in \mathbb{R}^{N\times N}$ with non-negative entries ($c_{nm} \geq 0, \forall n, m = 1,\ldots,N$) and the following properties:

$$\sum_{m=1}^{N} c_{nm} = 1, \forall n = 1,\ldots,N, \tag{76}$$

$$c_{nm} = 0, \forall m \notin n \cup \{\Omega_n \cap \mathcal{C}_{h(n)}\}, n = 1,\ldots N. \tag{77}$$

Denote the objective function of problem (75) as $F_T(\mathbf{w})$. Thus, the subdifferential of $F_T(\mathbf{w})$ with respect to $\mathbf{w}_n$ is:

$$\begin{aligned}\partial_{\mathbf{w}_n} F_T(\mathbf{w}) = {} & 2\mathbf{R}_n(T)\mathbf{w}_n - 2\mathbf{p}_n(T) \\ & + \sum_{m\in\Omega_n\setminus\mathcal{C}_{h(n)}} (\beta_{nm} + \beta_{mn})(2\mathbf{w}_n - 2\mathbf{w}_m) \\ & + \gamma_n \,\text{sgn}(\mathbf{w}_n). \end{aligned} \tag{78}$$

Then, the distributed online subgradient method for problem (75) at each node n is to perform the following update:

$$\mathbf{w}_n(T) = \sum_{m=1}^{N} c_{nm}\mathbf{w}_m(T-1) - \alpha\partial_{\mathbf{w}_n} F_T(\mathbf{w}(T-1)), \tag{79}$$

in which the linear combination (the first term) is only between neighbor nodes belonging to the same cluster. For simplification, we define:

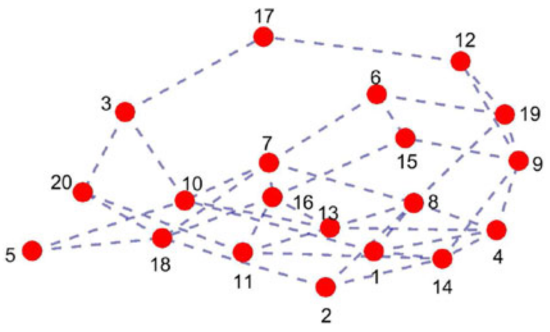


Fig. 1. The network topology.

$$\overline{\beta}_n := \sum_{m\in\Omega_n\setminus\mathcal{C}_{h(n)}} (\beta_{mn} + \beta_{nm}), \tag{80}$$

$$\overline{\mathbf{w}}_n(T) := \sum_{m\in\Omega_n\setminus\mathcal{C}_{h(n)}} (\beta_{mn} + \beta_{nm})\mathbf{w}_m(T). \tag{81}$$

Thus, the update of the distributed online subgradient method at each node n becomes:

$$\begin{aligned}\mathbf{w}_n(T) = {} & \sum_{m=1}^{N} c_{nm}\mathbf{w}_m(T-1) \\ & - \alpha[2\mathbf{R}_n(T)\mathbf{w}_n(T-1) - 2\mathbf{p}_n(T) \\ & + 2\overline{\beta}_n\mathbf{w}_n(T-1) - 2\overline{\mathbf{w}}_n(T-1) \\ & + \gamma_n\,\text{sgn}(\mathbf{w}_n(T-1))]. \end{aligned} \tag{82}$$

The matrix $\mathbf{R}_n(T)$ and the vector $\mathbf{p}_n(T)$ can also be updated according to (58) and (59) respectively when new data arrive.

V. Numerical Evaluation

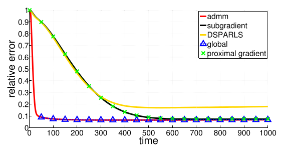
In this section, numerical simulations are conducted to verify the effectiveness of the proposed decentralized online ADMM algorithm (or ADMM algorithm in short), Algorithm 1, and the proposed decentralized online subgradient method (or subgradient method in short), Algorithm 2. The performance of the global offline optimization of problem (2) (global optimizor henceforth) is shown as a benchmark. The performance of the proposed distributed online proximal gradient method is also reported. For the sake of comparison, the performance of the distributed single task sparse RLS algorithm in [14] (DSPARLS henceforth) is presented as well to highlight the impact of multitask.

We consider a network with $N = 20$ nodes and 40 random edges so that the average node degree is 4. The network topology is illustrated in Fig. 1. The dimension of the input data is $M = 20$. Each entry of the input data sequence $\{\mathbf{u}_n(t)\}$ is generated according to the uniform distribution over the interval $[0, 1]$ independently. The noise sequence $\{e_n(t)\}$ is generated according to the uniform distribution on $[0, N_0]$ independently, where N_0 is a constant controlling the noisy level of the observations. To achieve sparsity, we let 18 entries (whose positions are randomly selected) of the true weight vectors $\widetilde{\mathbf{w}}_n(t)$ be zero. The two remaining entries $\widetilde{\mathbf{w}}_n^{\text{part}}(0) \in \mathbb{R}^2$ of the initial weight vectors are generated in a way that enforces similarity between neighbors. Specifically, we first generate N i.i.d. two dimensional random vectors $\{\boldsymbol{\phi}_n\}_{n=1,\ldots,N}$ uniformly distributed on $[0,1]^2$. Then, we solve the following optimization problem to obtain $\widetilde{\mathbf{w}}_n^{\text{part}}(0)$:

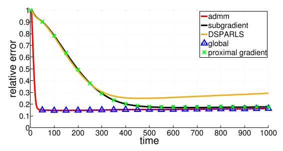
$$\begin{aligned}&\{\widetilde{\mathbf{w}}_n^{\text{part}}(0)\}_{n=1,\ldots,N}\\&= \arg \min_{\mathbf{w}_n \in \mathbb{R}^2, n=1,\ldots,N} \left\{ \sum_{n=1}^{N} \|\mathbf{w}_n - \boldsymbol{\phi}_n\|_2^2 \right.\\&\left. + \frac{1}{2}\sum_{n=1}^{N}\sum_{m\in\Omega_n} \|\mathbf{w}_n - \mathbf{w}_m\|_2^2 \right\}, \qquad (83)\end{aligned}$$

which promotes similarity between neighbors and can be easily solved as the objective function is a convex quadratic function. To capture the slowly time-variant trait of the weight vectors, the increment from $\widetilde{\mathbf{w}}_n(t)$ to $\widetilde{\mathbf{w}}_n(t+1)$, i.e., $\widetilde{\mathbf{w}}_n(t+1) - \widetilde{\mathbf{w}}_n(t)$ is generated by uniform distribution on $[-0.5N_1, 0.5N_1]$ independently across time and nodes, where N_1 is a constant controlling the varying rate of the weight vectors.

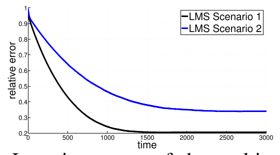
Now, we choose the regularization parameters and forgetting factor as $\beta = \gamma = 1, \lambda = 0.995$. We consider two scenarios in terms of the noise level N_0 and the varying rate of weight vectors N_1. In Scenario 1, $N_0 = 0.1, N_1 = 0.02$ while in Scenario 2, $N_0 = 0.3, N_1 = 0.05$. The latter scenario has noisier observations and weight vectors which vary faster. Thus, the weight vectors of Scenario 2 are more difficult to track than those of Scenario 1. In particular, we note that the noises and the signals are roughly of the same scale in Scenario 2. In such a scenario, the average value of the signal $\mathbf{u}_n(t)^\mathsf{T}\widetilde{\mathbf{w}}_n(t)$ is approximately equal to $2\times 0.5\times 0.5 = 0.5$ (note that only two entries of the true weight vector are nonzero) while the noise is drawn from the interval $[0, 0.3]$. In addition, the scale of the temporal variations of the true weight vectors is smaller than that of the true weight vectors themselves by only one order of magnitude, which can be regarded as a fast time-varying scenario. For the proposed ADMM algorithm, the proposed subgradient method, the proposed proximal gradient method, the DSPARLS algorithm in [14] and the global optimizer, we plot the relative errors (defined to be $\|\mathbf{w}(t) - \widetilde{\mathbf{w}}(t)\|_2/\|\widetilde{\mathbf{w}}(t)\|_2$, where $\mathbf{w}(t)$ and $\widetilde{\mathbf{w}}(t)$ are concatenations of $\mathbf{w}_n(t)$ and $\widetilde{\mathbf{w}}_n(t)$ of all nodes, respectively) as functions of time indices, i.e., the learning curves, under Scenario 1 (Fig. 2(a)) and Scenario 2 (Fig. 2(b)), respectively. Each learning curve is the average of 300 independent trials. Several interesting observations can be made from Fig. 2. First, the relative errors of both the proposed ADMM algorithm and the proposed subgradient method can converge to that of the global optimizor, i.e., the performance benchmark, as the observation data accumulate. On the contrary, the relative error of DSPARLS does not converge to that of the global optimizor. This highlights the effectiveness of the two proposed algorithms when tracking multitask weight vectors, which cannot be tracked well by existing method (DSPARLS in this case) for the single task situation. Second, comparisons between the learning curves of the proposed two algorithms indicate that the proposed ADMM algorithm needs much fewer observations, or equivalently much less time (about 100 time units), to track the weight vectors accurately than the proposed subgradient method does (about 600 time units). This is not surprising as dual domain methods generally converge faster than primal domain methods in the literature of optimization theory [36]. However, the advantage of the proposed ADMM algorithm in convergence speed comes at the cost of higher computational overhead per time unit than the proposed subgradient method. This accuracy-complexity tradeoff makes the proposed two algorithms appropriate for different applications depending on the computational capability of devices and needed tracking accuracy. Besides, for comparison purpose, we remark that hundreds of time units are typical amount of time needed for convergence of distributed adaptive algorithms for either single-task or multitask networks [2], [10]–[13], [16], [18], [29]. Third, as one expects, Scenario 1 has better tracking performance than Scenario 2: the ultimate relative error of the proposed algorithms in Scenario 1 is about 0.067 while that of Scenario 2 is about 0.17. So, higher noise level and faster varying speed of the weight vectors do result in lower tracking accuracy. We also observe that, in either scenario, the performance of the proximal gradient method is very similar to that of the subgradient method. This similarity in performance is reasonable as the algorithms of the two methods do not differ much. Lastly, in Fig. 2(c), we report the learning curves of the multitask diffusion LMS with l_1 regularization [16] in both Scenario 1 and Scenario 2. The value of α is increased to 0.01 to accelerate convergence (further increasing α will increase errors or even lead to instability). We remark that, in either scenario, the LMS method incurs larger tracking errors than the proposed RLS based Algorithms 1 and 2 and needs more time (about 1500 and 2000 for Scenarios 1 and 2, respectively) to converge. This highlights the convergence advantage of RLS over LMS, which comes at the expense of higher computational complexity [1], [29].



(a) Learning curve of Scenario 1



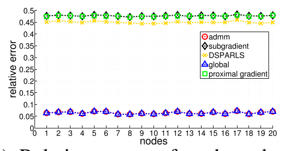
(b) Learning curve of Scenario 2



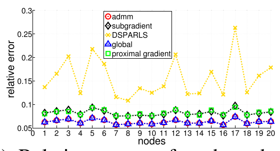
(c) Learning curve of the multitask LMS

Fig. 2. Learning curves of different algorithms in different scenarios.

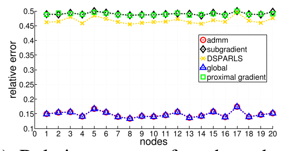
Next, we investigate the tracking performance of each individual node. To this end, in Fig. 3, we show the relative tracking errors of each node at time 200 and 500 under Scenario 1 and Scenario 2. Several remarks are in order. First, we note that in all four cases of Fig. 3, the red curve (the proposed ADMM algorithm) and the blue curve (the global optimizor) coincide precisely for every node. This further confirms the previous observation from Fig. 2 that the performance of the



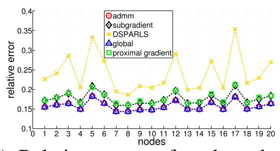
(a) Relative errors of each node at time 200 in Scenario 1



(b) Relative errors of each node at time 500 in Scenario 1

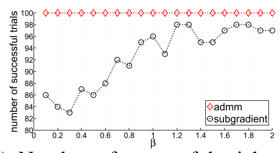


(c) Relative errors of each node at time 200 in Scenario 2

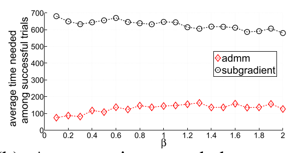


(d) Relative errors of each node at time 500 in Scenario 2

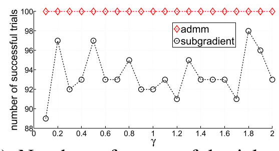
Fig. 3. Relative tracking errors of each node.



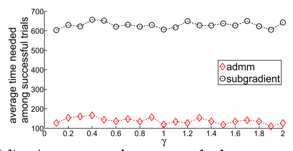
(a) Number of successful trials for different β



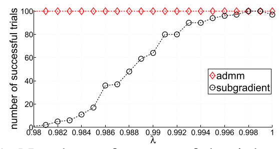
(b) Average time needed to reach success among successful trials for different β



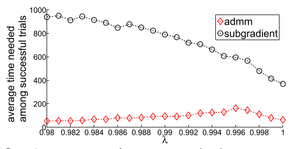
(c) Number of successful trials for different γ



(d) Average time needed to reach success among successful trials for different γ



(e) Number of successful trials for different λ



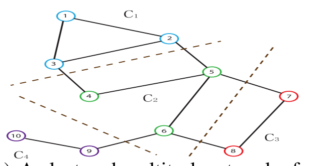
(f) Average time needed to reach success among successful trials for different λ

Fig. 4. Number of successful trials and the average time needed to reach success among successful trials.

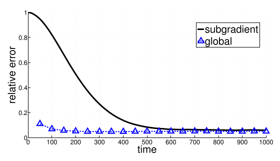
proposed ADMM can converge to that of the global optimizor quickly. Second, the proposed subgradient method, though performs poorly at time 200, has relative errors close to those of the global optimizor at time 500. This suggests that the proposed subgradient method eventually has performance close to the benchmark (the global optimizor). But this good performance necessitates longer time (or equivalently more data) compared to the proposed ADMM algorithm. Third, the performance of the single task learning algorithm DSPARLS never converge to that of the global optimizor. In particular, from Fig. 3(b) and (d), the performance of DSPARLS is worst at node 5 and node 17. Recall the network topology in Fig. 1 and we see that these two nodes are loosely connected to other nodes, i.e., their degree is low. Thus, the weight vectors at these two nodes can potentially deviate far from the weight vectors at the rest of the nodes and thus violate the single task assumption of DSPARLS the most among all nodes. This partially explains the poor performance of DSPARLS at nodes 5 and 17. Last, we observe that the performance of the proximal gradient method is very analogous to that of the subgradient method at every node in either scenario, confirming the similarity between the proximal gradient method and the subgradient method again.

Previous experiments indicate that the proposed ADMM algorithm possesses faster and more accurate tracking performance than the proposed subgradient method. Next, we conduct a more thorough performance comparison between the proposed two algorithms for different regularization parameters β, γ and different forgetting factors λ. We note that the global optimizor usually converges well before time 1000 and we denote the steady relative error of the global optimizor at time 1000 as $\breve{e}$. We say a simulation trial of an algorithm (either the proposed ADMM algorithm or the proposed subgradient method) is *successful* if, before time 1000, there exists a time window (i.e., interval) of length 20 over which the average relative error of the algorithm is lower than $1.1\breve{e}$. The basic parameter setup is $\gamma = \beta = 1, \lambda = 0.995$. In each of the subfigures in Fig. 4(a), (c), and (e), we vary one parameter while keep the remaining two parameters the same as the basic setup. For each parameter setup, we conduct 100 independent trials and plot the number of successful trials in Fig. 4(a), (c), and (e). We observe that (i) the proposed ADMM algorithm is always successful, i.e., it can always converge to the steady performance of the global optimizor; (ii) the proposed subgradient method is successful in most trials as long as the forgetting factor λ is sufficiently close to 1, which is the case in most applications (e.g., $\lambda = 0.995$) as the weight vectors are varying very slowly and a large λ is needed for tracking them. Moreover, we investigate the average time needed to reach success (defined to be the middle point of the first time window over which the average relative error is less than $1.1\breve{e}$) among successful trials. The results are shown in Fig. 4(b), (d), and (f). We remark that the proposed ADMM mostly needs no more than 150 time units to be successful, i.e., be close to the steady performance of the global optimizor, while it takes the proposed subgradient method a much longer time (around 600 time units) to be successful. This further confirms our previous assertion that the proposed ADMM algorithm possesses faster tracking performance than the proposed subgradient method.

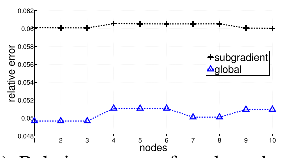
Next, we endeavor to validate the effectiveness of the proposed distributed online subgradient method for clustered multitask networks with edge dependent β and node dependent γ (Section IV-D). We consider a clustered multitask network with $N = 10$ nodes and $Q = 4$ clusters (Fig. 5(a)), the same as the one used in [16]. As before, we still set the data dimension to be $M = 20$, among which only two random positions correspond to non-zero entries. Initially, the two nonzero entries of the true weight vectors of the four clusters are set to be $\widetilde{\mathbf{w}}^{\#}_{1,\text{part}} = [1.53, -0.98]^{\mathsf{T}}, \widetilde{\mathbf{w}}^{\#}_{2,\text{part}} = [1.5, -1]^{\mathsf{T}}, \widetilde{\mathbf{w}}^{\#}_{3,\text{part}} = [1.48, -1.02]^{\mathsf{T}}, \widetilde{\mathbf{w}}^{\#}_{4,\text{part}} = [1.55, -1.04]^{\mathsf{T}}$. The evolution of the time-varying true weight vectors, the generation of input sequences and the generation of the noise sequences are similar to the scenario used for Fig. 2. We still choose $\lambda = 0.995$. The proximity regularization parameters β_{nm} and the sparsity



(a) A clustered multitask network of four clusters [16]



(b) The learning curve



(c) Relative errors of each node at time 1000

Fig. 5. Clustered multitask networks with edge dependent β and node dependent γ.

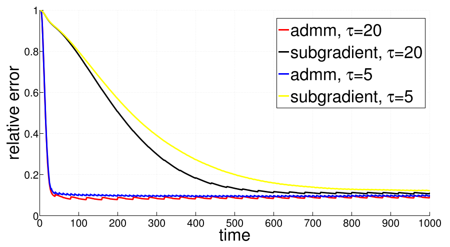


Fig. 6. The learning curves for dynamic networks with different evolution rates.

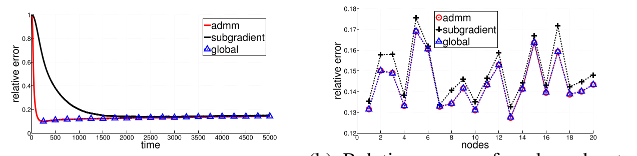
regularization parameters γ_n are randomly generated within the interval $[0.8, 1.2]$. In Fig. 5(b) and (c), we report the learning curve, i.e., the global relative error defined previously for Fig. 2, and the steady state relative errors at each individual node, respectively. The performance of the global optimizor is also shown as a centralized offline benchmark. We observe that, similar to the non-clustered multitask network (c.f. Figs. 2 and 3), the performance of the distributed online subgradient method will ultimately converge to that of the global optimizor and it needs around 600 time units to achieve this convergence. This indicates the robustness of the subgradient method to clustering of multitask networks.

In the following, we want to test the robustness of the proposed Algorithms 1 and 2 when the network topology varies across time. To this end, we randomly delete old nodes from the network and add new nodes to the network. Specifically, the initial network topology is given by Fig. 1. Then, after every τ time units, we either delete an existing node randomly or add a new node with four random links, as the average degree of the initial graph in Fig. 1 is four. The parameter τ controls how fast the network evolves. The learning curves (relative errors) of the proposed ADMM algorithm and the proposed subgradient method are shown in Fig. 6 for $\tau = 20$ and $\tau = 5$. Comparing the learning curves of dynamic network in Fig. 6 with those of static network in Fig. 2, we observe that the performance of the proposed ADMM algorithm is basically not affected by the evolution of networks in terms of both convergence time and the steady state errors. The performance of the proposed subgradient method is not influenced much either when $\tau = 20$. When the network evolves faster, i.e., $\tau = 5$, the convergence time of the subgradient method increases to around 800 (the convergence time for static network is about 600) and the steady state relative error also increases slightly. Overall, we can conclude that the proposed two algorithms are robust to the temporal variations of network topologies.

Finally, we apply the proposed ADMM and subgradient algorithms to distributed cooperative spectrum sensing [49]. Consider a cognitive radio network (CRN) with N secondary users (SUs). A link between two SUs indicate that they are close in physical locations and are willing to cooperate to sense the spectrum. Aftering spectrum sensing, SUs can exploit the spectrum temporarily unused by the primary users (PUs). There are M frequency bands in total, among which the m-th band is centered at frequency point $f_m \in \left[-\frac{1}{2}, \frac{1}{2}\right]$. Suppose the power spectrum of signals transmitted on the m-th band can be represented (or approximated) as scalar multiple of the signature spectrum $S_m(f) = \exp\left(-\frac{(f - f_m)^2}{2\sigma_m^2}\right)$. Define $\mathbf{s}(f) = [S_1(f), \ldots, S_M(f)]^\mathsf{T}$. Each PU uses at most one frequency band at each time and the number of active PUs is much smaller than M, i.e., the number of bands (otherwise, it is hard for SUs to get opportunities to use the spectrum and a CRN is not suitable here). At each time t, each SU n senses the received power spectrum at frequency point $\check{f}_{n,t}$ and observes an output $d_n(t)$ given as follows:

$$d_n(t) = \widetilde{\mathbf{w}}_n(t)^\mathsf{T} \mathbf{s}\left(\check{f}_{n,t}\right) + e_n(t), \tag{84}$$

where $\widetilde{\mathbf{w}}_n(t)$ specifies the received signal strengths at all bands by node n and $e_n(t)$ is the noise. The goal of SU n is to infer $\widetilde{\mathbf{w}}_n(t)$ so that she can exploit the temporarily unused frequency bands. $\widetilde{\mathbf{w}}_n(t)$ is determined by the transmitted signal strength at each band and the path losses from the signal sources (or PUs) to SU n. Thus, for different nodes, the unknown $\widetilde{\mathbf{w}}_n(t)$'s are generally different due to the difference in physical locations and path losses. Since the number of active PUs is much smaller than M, $\widetilde{\mathbf{w}}_n(t)$'s are sparse vectors. Besides, if SU n and SU n' are neighbors in the CRN, $\widetilde{\mathbf{w}}_n(t)$ and $\widetilde{\mathbf{w}}_{n'}(t)$ should be similar because the two SUs are close in physical locations and should have similar path losses. A reasonable way to enforce this proximity is to minimize the squared l_2 norm of the difference. Therefore, by defining $\mathbf{u}_n(t) = \mathbf{s}\left(\check{f}_{n,t}\right)$, we see that the signal model (84) is in the form of (1) and the distributed sparse multitask RLS problem in (2) can be used to estimate $\widetilde{\mathbf{w}}_n(t)$'s. The CRN topology is set to be the same as the network in Fig. 1. The number of frequency bands is $M = 20$ and there are two active PUs, i.e., two nonzero entries in each $\widetilde{\mathbf{w}}_n(t)$. The centers of the frequency bands, i.e., $f_m, m = 1, \ldots, M$, are uniformly distributed over the interval $\left[-\frac{1}{2}, \frac{1}{2}\right]$ while the sample frequency points $\check{f}_{n,t}$ are randomly generated within $\left[-\frac{1}{2}, \frac{1}{2}\right]$. Other details of the setup are the same as those specified previously for Fig. 2. The learning curves (global relative errors) and the steady state relative errors at each individual node are shown in Fig. 7(a) and (b), respectively. We also plot the performance of the global optimizor as a centralized offline benchmark. The convergence of ADMM is faster than that of the subgradient method at an expense of higher computational complexity.



(a) The learning curve

(b) Relative errors of each node at time 5000

Fig. 7. Distributed cooperative spectrum sensing.

VI. Conclusion

In this paper, we study the decentralized sparse multitask RLS problem. We first propose a decentralized online ADMM algorithm for the formulated RLS problem. We simplify the algorithm so that each node only needs to store and update one $M \times M$ matrix and six M dimensional vectors. Convergence of the proposed ADMM algorithm is also established. Moreover, to further reduce the computational complexity, we propose a decentralized online subgradient method. Mean square stablitiy of the subgradient method is guaranteed by deriving explicit upper bounds on its mean square deviation and excess mean square error. Compared with the ADMM algorithm, the subgradient method enjoys lower computational complexity at the expense of slower convergence speed. The proposed algorithms are corroborated by numerical experiments.

Appendix A
Proof of Theorem 1

From the definition of $\mathbf{R}_n(T)$ and $\mathbf{p}_n(T)$, we know that they are weighted sum of i.i.d. terms. According to the strong law of large numbers for weighted sums [9], [50], as $T \to \infty$, $\mathbf{R}_n(T)$ converges to $\lim_{T\to\infty} \mathbb{E}[\mathbf{R}_n(T)] = \frac{\mathbf{R}_n}{1-\lambda}$ almost surely. Similarly, $\mathbf{p}_n(T)$ converges to $\lim_{T\to\infty} \mathbb{E}[\mathbf{p}_n(T)] = \frac{\mathbf{R}_n \widetilde{\mathbf{w}}_n}{1-\lambda}$ almost surely. Algorithm 1 is to apply ADMM to the dynamic (time-varying) optimization problem (8), which converges almost surely to the following static optimization problem:

$$\begin{aligned}
&\text{Minimize}_{\mathbf{x},\mathbf{w},\mathbf{v}}\\
&\frac{1}{1-\lambda}\sum_{n=1}^{N}\left(\mathbf{x}_n^{\mathsf{T}}\mathbf{R}_n\mathbf{x}_n - 2\widetilde{\mathbf{w}}_n^{\mathsf{T}}\mathbf{R}_n\mathbf{x}_n\right)\\
&+\beta\sum_{n=1}^{N}\left[|\Omega_n|\|\mathbf{x}_n\|_2^2 - 2\left(\sum_{i=1}^{|\Omega_n|}\mathbf{v}_{n,i}\right)^{\mathsf{T}}\mathbf{x}_n + \sum_{i=1}^{|\Omega_n|}\|\mathbf{v}_{n,i}\|_2^2\right]\\
&+\gamma\sum_{n=1}^{N}\|\mathbf{w}_n\|_1\\
&\text{s.t.}\quad \mathbf{x}_n = \mathbf{w}_n, n = 1,\ldots,N,\\
&\qquad\ \mathbf{v}_{n,i} = \mathbf{w}_{g(n,i)}, n = 1\ldots,N, i = 1\ldots,|\Omega_n|. \qquad (85)
\end{aligned}$$

We note that problem (85) is in the form of (3). Hence, according to the convergence of ADMM for static convex optimization problems of the form (3), we know that the output of Algorithm 1, i.e., $\mathbf{w}(T)$, converges almost surely to the optimal point of (85). Eliminating the dummy variables $\mathbf{x}$ and $\mathbf{v}$, we can equivalently rewrite problem (85) as:

$$\begin{aligned}
\text{Minimize}_{\mathbf{w}}\ &\frac{1}{1-\lambda}\sum_{n=1}^{N}\|\mathbf{w}_n - \widetilde{\mathbf{w}}_n\|_{\mathbf{R}_n}^2\\
&+\beta\sum_{n=1}^{N}\sum_{m\in\Omega_n}\|\mathbf{w}_n - \mathbf{w}_m\|_2^2 + \gamma\sum_{n=1}^{N}\|\mathbf{w}_n\|_1.
\end{aligned} \tag{86}$$

Therefore, we conclude that $\mathbf{w}_n(T)$ converges to the optimal point of problem (86) almost surely.

Appendix B
Proof of Theorem 2

As demonstrated in Appendix A, for large T, we can approximate $\mathbf{R}_n(T)$ by $\frac{\mathbf{R}_n}{1-\lambda}$ and $\mathbf{p}_n(T)$ by $\frac{\mathbf{R}_n\widetilde{\mathbf{w}}_n}{1-\lambda}$. Define the error vector of node n at time T to be:

$$\mathbf{f}_n(T) = \mathbf{w}_n(T) - \widetilde{\mathbf{w}}_n. \tag{87}$$

Thus, substituting (87) into the definition of $\overline{\mathbf{w}}_n(T-1)$ yields:

$$\overline{\mathbf{w}}_n(T-1) = \sum_{m\in\Omega_n}\left(\widetilde{\mathbf{w}}_m + \mathbf{f}_m(T-1)\right). \tag{88}$$

Hence, using (60), (87) and the approximation of $\mathbf{R}_n(T)$, $\mathbf{p}_n(T)$, we can derive a recursive equation for the error vector:

$$\begin{aligned}
\mathbf{f}_n(T) &= \mathbf{w}_n(T-1) - \widetilde{\mathbf{w}}_n - \alpha\Bigg[2\mathbf{R}_n(T)\left(\widetilde{\mathbf{w}}_n + \mathbf{f}_n(T-1)\right)\\
&\quad - 2\mathbf{p}_n(T) + 4\beta\Bigg(|\Omega_n|\left(\widetilde{\mathbf{w}}_n + \mathbf{f}_n(T-1)\right)\\
&\quad - \sum_{m\in\Omega_n}\left(\widetilde{\mathbf{w}}_m + \mathbf{f}_m(T-1)\right)\Bigg) + \gamma\,\mathrm{sgn}(\mathbf{w}_n(T-1))\Bigg]
\end{aligned} \tag{89}$$

$$\begin{aligned}
&\approx \mathbf{f}_n(T-1) - \alpha\Bigg[\frac{2\mathbf{R}_n}{1-\lambda}\left(\widetilde{\mathbf{w}}_n + \mathbf{f}_n(T-1)\right) - \frac{2\mathbf{R}_n\widetilde{\mathbf{w}}_n}{1-\lambda}\\
&\quad + 4\beta\Bigg(|\Omega_n|\widetilde{\mathbf{w}}_n - \sum_{m\in\Omega_n}\widetilde{\mathbf{w}}_m + |\Omega_n|\mathbf{f}_n(T-1)\\
&\quad - \sum_{m\in\Omega_n}\mathbf{f}_m(T-1)\Bigg) + \gamma\,\mathrm{sgn}(\mathbf{w}_n(T-1))\Bigg]
\end{aligned} \tag{90}$$

$$\begin{aligned}
&= \left(\mathbf{I} - \frac{2\alpha\mathbf{R}_n}{1-\lambda}\right)\mathbf{f}_n(T-1)\\
&\quad - 4\alpha\beta\left(|\Omega_n|\mathbf{f}_n(T-1) - \sum_{m\in\Omega_n}\mathbf{f}_m(T-1)\right)\\
&\quad - 4\alpha\beta\left(|\Omega_n|\widetilde{\mathbf{w}}_n - \sum_{m\in\Omega_n}\widetilde{\mathbf{w}}_m\right) - \alpha\gamma\,\mathrm{sgn}(\mathbf{w}_n(T-1)).
\end{aligned} \tag{91}$$

Define $\mathbf{f}(T) = \left[\mathbf{f}_1(T)^\mathsf{T}, \ldots, \mathbf{f}_N(T)^\mathsf{T}\right]^\mathsf{T}$ and $\boldsymbol{\xi}(T) = -4\alpha\beta \mathbf{L}\widetilde{\mathbf{w}} - \alpha\gamma\,\mathrm{sgn}(\mathbf{w}(T-1))$. Stacking (91) for all nodes n, we obtain:

$$\mathbf{f}(T) = \left(\mathbf{I} - \frac{2\alpha}{1-\lambda}\mathbf{R} - 4\alpha\beta\mathbf{L}\right)\mathbf{f}(T-1) + \boldsymbol{\xi}(T). \quad (92)$$

Define $\boldsymbol{\Psi} = \mathbf{I} - \frac{2\alpha}{1-\lambda}\mathbf{R} - 4\alpha\beta\mathbf{L}$ and thus:

$$\mathbf{f}(T) = \boldsymbol{\Psi}\mathbf{f}(T-1) + \boldsymbol{\xi}(T). \quad (93)$$

From the definitions of $\mu_i, i = 1, 2, 3$, we know that $\mu_1\mathbf{I} \preceq \mathbf{R} \preceq \mu_2\mathbf{I}$ and $\mathbf{0} \preceq \mathbf{L} \preceq \mu_3\mathbf{I}$. Substituting these bounds into the expression of $\boldsymbol{\Psi}$, we get:

$$\left(1 - \frac{2\alpha\mu_2}{1-\lambda} - 4\alpha\beta\mu_3\right)\mathbf{I} \preceq \boldsymbol{\Psi} \preceq \left(1 - \frac{2\alpha\mu_1}{1-\lambda}\right)\mathbf{I}. \quad (94)$$

According to the assumption in (68), we know that $1 - \frac{2\alpha\mu_2}{1-\lambda} - 4\alpha\beta\mu_3 > -1$. Hence, the eigenvalues of $\boldsymbol{\Psi}$ satisfy:

$$-1 < 1 - \frac{2\alpha\mu_2}{1-\lambda} - 4\alpha\beta\mu_3 \le \lambda_i(\boldsymbol{\Psi}) \le 1 - \frac{2\alpha\mu_1}{1-\lambda} < 1, \quad \forall i = 1, \ldots, NM. \quad (95)$$

Therefore,

$$\begin{aligned}\|\boldsymbol{\Psi}\|_2 &= \rho(\boldsymbol{\Phi}) \\ &\le \max\left\{\left|1 - \frac{2\alpha\mu_2}{1-\lambda} - 4\alpha\beta\mu_3\right|, \left|1 - \frac{2\alpha\mu_1}{1-\lambda}\right|\right\} \\ &= \delta < 1, \end{aligned} \quad (96)$$

where $\|\boldsymbol{\Psi}\|_2$ and $\rho(\boldsymbol{\Phi}) = \max_{i=1,\ldots,NM}\{|\lambda_i(\boldsymbol{\Psi})|\}$ denote the maximum singular value and the spectral radius of $\boldsymbol{\Psi}$, respectively. Applying (93) recursively yeilds:

$$\mathbf{f}(T) = \boldsymbol{\Psi}^T\mathbf{f}(0) + \sum_{t=0}^{T-1}\boldsymbol{\Psi}^t\boldsymbol{\xi}(T-t), \quad (97)$$

in which the superscript T denotes time instead of transposition. Note that $\boldsymbol{\xi}(T)$ can be uniformly bounded as follows:

$$\begin{aligned}\|\boldsymbol{\xi}(T)\|_2 &\le 4\alpha\beta\|\mathbf{L}\widetilde{\mathbf{w}}\|_2 + \alpha\gamma\sqrt{MN} \\ &\le 4\alpha\beta\mu_3\|\widetilde{\mathbf{w}}\|_2 + \alpha\gamma\sqrt{MN} = \Gamma. \end{aligned} \quad (98)$$

Taking norms on both sides of (97) gives:

$$\|\mathbf{f}(T)\|_2 \le \|\boldsymbol{\Psi}\|_2^T\|\mathbf{f}(0)\|_2 + \sum_{t=0}^{T-1}\|\boldsymbol{\Psi}\|_2^t\|\boldsymbol{\xi}(T-t)\|_2 \quad (99)$$

$$\le \delta^T\|\mathbf{f}(0)\|_2 + \sum_{t=0}^{T-1}\delta^t \cdot \Gamma \quad (100)$$

$$\le \delta^T\|\mathbf{f}(0)\|_2 + \frac{\Gamma}{1-\delta}. \quad (101)$$

Taking expected squares of the both sides of (101), we obtain:

$$\begin{aligned}&\mathbb{E}\left[\|\mathbf{f}(T)\|_2^2\right] \\ &\le \delta^{2T}\mathbb{E}\left[\|\mathbf{f}(0)\|_2^2\right] + \frac{2\delta^T\Gamma}{1-\delta}\mathbb{E}\left[\|\mathbf{f}(0)\|_2\right] + \frac{\Gamma^2}{(1-\delta)^2}. \end{aligned} \quad (102)$$

Note the first two terms of the R.H.S. of (102) converge to zero as T goes to infinity. Thus, the steady state mean square deviation (MSD) satisfies:

$$\mathrm{MSD} = \limsup_{T\to\infty}\mathbb{E}\left[\|\mathbf{f}(T)\|_2^2\right] \le \frac{\Gamma^2}{(1-\delta)^2}. \quad (103)$$

Furthermore, we derive:

$$\mathbb{E}\left[\sum_{n=1}^N\left(\mathbf{u}_n(t)^\mathsf{T}\mathbf{f}_n(t-1)\right)^2\right] \quad (104)$$

$$\le \mathbb{E}\left[\sum_{n=1}^N\|\mathbf{u}_n(t)\|_2^2\|\mathbf{f}_n(t-1)\|_2^2\right] \quad (105)$$

$$= \sum_{n=1}^N\mathbb{E}\left[\|\mathbf{u}_n(t)\|_2^2\right]\mathbb{E}\left[\|\mathbf{f}_n(t-1)\|_2^2\right] \quad (106)$$

$$\le \left[\max_{n=1,\ldots,N}\mathrm{tr}(\mathbf{R}_n)\right]\mathbb{E}\left[\|\mathbf{f}(t-1)\|_2^2\right], \quad (107)$$

where (106) is due to the fact that $\mathbf{u}_n(t)$ and $\mathbf{f}_n(t-1)$ are independent. Taking limits on both sides of (107) and making use of (103), we get the following bound for the steady state excess mean square error (EMSE):

$$\begin{aligned}\mathrm{EMSE} &= \limsup_{t\to\infty}\mathbb{E}\left[\sum_{n=1}^N\left(\mathbf{u}_n(t)^\mathsf{T}\mathbf{f}_n(t-1)\right)^2\right] \\ &\le \left[\max_{n=1,\ldots,N}\mathrm{tr}(\mathbf{R}_n)\right]\frac{\Gamma^2}{(1-\delta)^2}. \end{aligned} \quad (108)$$

REFERENCES

[1] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

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

[3] F. S. Cattivelli and A. H. Sayed, "Diffusion lms strategies for distributed estimation," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1035–1048, Mar. 2010.

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

[5] A. H. Sayed, "Adaptive networks," *Proc. IEEE*, vol. 102, no. 4, pp. 460–497, Apr. 2014.

[6] C. Jiang, Y. Chen, and K. J. R. Liu, "Distributed adaptive networks: A graphical evolutionary game-theoretic view," *IEEE Trans. Signal Process.*, vol. 61, no. 22, pp. 5675–5688, Nov. 2013.

[7] G. Su, J. Jin, Y. Gu, and J. Wang, "Performance analysis of l_0 norm constraint least mean square algorithm," *IEEE Trans. Signal Process.*, vol. 60, no. 5, pp. 2223–2235, May 2012.

[8] J. Jin, Y. Gu, and S. Mei, "A stochastic gradient approach on compressive sensing signal reconstruction based on adaptive filtering framework," *IEEE J. Sel. Top. Signal Process.*, vol. 4, no. 2, pp. 409–420, 2010.

[9] B. Babadi, N. Kalouptsidis, and V. Tarokh, "SPARLS: The Sparse RLS algorithm," *IEEE Trans. Signal Process.*, vol. 58, no. 8, pp. 4013–4025, Aug. 2010.

[10] P. Di Lorenzo and A. H. Sayed, "Sparse distributed learning based on diffusion adaptation," *IEEE Trans. Signal Process.*, vol. 61, no. 6, pp. 1419–1433, Mar. 2013.

[11] Y. Liu, C. Li, and Z. Zhang, "Diffusion sparse least-mean squares over networks," *IEEE Trans. Signal Process.*, vol. 60, no. 8, pp. 4480–4485, Aug. 2012.

[12] S. Huang and C. Li, "Distributed sparse total least-squares over networks," *IEEE Trans. Signal Process.*, vol. 63, no. 11, pp. 2986–2998, Jun. 2015.

[13] S. Chouvardas, K. Slavakis, Y. Kopsinis, and S. Theodoridis, "A sparsity promoting adaptive algorithm for distributed learning," *IEEE Trans. Signal Process.*, vol. 60, no. 10, pp. 5412–5425, Oct. 2012.

[14] Z. Liu, Y. Liu, and C. Li, "Distributed sparse recursive least-squares over networks," *IEEE Trans. Signal Process.*, vol. 62, no. 6, pp. 1386–1395, Mar. 2014.

[15] R. Nassif, "Distributed adaptive estimation over multitask networks," Ph.D. thesis, Université Côte d'Azur, Provence-Alpes-Côte d'Azur, France, 2016.

[16] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Trans. Signal Process.*, vol. 62, no. 16, pp. 4129–4144, Aug. 2014.

[17] R. Nassif, C. Richard, A. Ferrari, and A. H. Sayed, "Multitask diffusion adaptation over asynchronous networks," *IEEE Trans. Signal Process.*, vol. 64, no. 11, pp. 2835–2850, Aug. 2014.

[18] R. Nassif, C. Richard, A. Ferrari, and A. H. Sayed, "Proximal multitask learning over networks with sparsity-inducing coregularization," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6329–6344, Dec. 2016.

[19] R. Nassif, C. Richard, A. Ferrari, and A. H. Sayed, "Diffusion LMS for multitask problems with local linear equality constraints," *IEEE Trans. Signal Process.*, vol. 65, no. 19, pp. 4979–4993, Oct. 2017.

[20] J. Plata-Chaves, N. Bogdanovic, and K. Berberidis, "Distributed incremental-based RLS for node-specific parameter estimation over adaptive networks," in *Proc. IEEE 21st Eur. Signal Process. Conf.*, 2013, pp. 1–5.

[21] J. Plata-Chaves, N. Bogdanović, and K. Berberidis, "Distributed diffusion-based LMS for node-specific adaptive parameter estimation," *IEEE Trans. Signal Process.*, vol. 63, no. 13, pp. 3448–3460, Jul. 2015.

[22] N. Bogdanovic, J. Plata-Chaves, and K. Berberidis, "Distributed incremental-based LMS for node-specific adaptive parameter estimation," *IEEE Trans. Signal Process.*, vol. 62, no. 20, pp. 5382–5397, Oct. 2014.

[23] J. Plata-Chaves, A. Bertrand, M. Moonen, S. Theodoridis, and A. M. Zoubir, "Heterogeneous and multitask wireless sensor networks—Algorithms, applications, and challenges," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 3, pp. 450–465, Apr. 2017.

[24] J. Chen, C. Richard, and A. H. Sayed, "Diffusion LMS over multitask networks," *IEEE Trans. Signal Process.*, vol. 63, no. 11, pp. 2733–2748, Jun. 2015.

[25] X. Zhao and A. H. Sayed, "Distributed clustering and learning over networks," *IEEE Trans. Signal Process.*, vol. 63, no. 13, pp. 3285–3300, Jul. 2015.

[26] S. Monajemi, K. Eftaxias, S. Sanei, and S.-H. Ong, "An informed multitask diffusion adaptation approach to study tremor in Parkinson's Disease," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 7, pp. 1306–1314, Oct. 2016.

[27] V. Kekatos and G. B. Giannakis, "Distributed robust power system state estimation," *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 1617–1626, May 2013.

[28] A. Hassani, J. Plata-Chaves, M. H. Bahari, M. Moonen, and A. Bertrand, "Multi-task wireless sensor network for joint distributed node-specific signal enhancement, lcmv beamforming and doa estimation," *IEEE J. Sel Topics Signal Process.*, vol. 11, no. 3, pp. 518–533, Apr. 2017.

[29] G. Mateos, I. D. Schizas, and G. B. Giannakis, "Distributed recursive least-squares for consensus-based in-network adaptive estimation," *IEEE Trans. Signal Process.*, vol. 57, no. 11, pp. 4583–4588, Nov. 2009.

[30] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learni.*, vol. 3, no. 1, pp. 1–122, 2011.

[31] W.-C. Liao, M. Hong, H. Farmanbar, X. Li, Z.-Q. Luo, and H. Zhang, "Min flow rate maximization for software defined radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1282–1294, Jun. 2014.

[32] C. Shen, T.-H. Chang, K.-Y. Wang, Z. Qiu, and C.-Y. Chi, "Distributed robust multicell coordinated beamforming with imperfect CSI: An ADMM approach," *IEEE Trans. Signal Process.*, vol. 60, no. 6, pp. 2988–3003, Jun. 2012.

[33] J. Zhang, S. Nabavi, A. Chakrabortty, and Y. Xin, "Admm optimization strategies for wide-area oscillation monitoring in power systems under asynchronous communication delays," *IEEE Trans. Smart Grid*, vol. 7, no. 4, pp. 2123–2133, Jul. 2016.

[34] T.-H. Chang, "A proximal dual consensus admm method for multi-agent constrained optimization," *IEEE Trans. Signal Process.*, vol. 64, no. 14, pp. 3719–3734, Jul. 2016.

[35] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Trans. Signal Process.*, vol. 62, no. 7, pp. 1750–1761, Apr. 2014.

[36] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "Dlm: Decentralized linearized alternating direction method of multipliers," *IEEE Trans. Signal Process.*, vol. 63, no. 15, pp. 4051–4064, Aug. 2015.

[37] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM J. Optim.*, vol. 25, no. 2, pp. 944–966, 2015.

[38] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via multi-task sparse learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2042–2049.

[39] A. Koppel, B. Sadler, and A. Ribeiro, "Proximity without consensus in online multi-agent optimization," *IEEE Trans. Signal Process.*, vol. 65, no. 12, pp. 3062–3077, Jun. 2017.

[40] W. Deng and W. Yin, "On the global and linear convergence of the generalized alternating direction method of multipliers," *J. Sci. Comput.*, vol. 66, no. 3, pp. 889–916, 2016.

[41] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, vol. 23. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.

[42] H. Wang and A. Banerjee, "Online alternating direction method," in *Proc. Int. Conf. Mach. Learn.*, 2012, pp. 1119–1126.

[43] H.-F. Xu, Q. Ling, and A. Ribeiro, "Online learning over a decentralized network through ADMM," *J. Oper. Res. Soc. China*, vol. 3, no. 4, pp. 537–562, 2015.

[44] Q. Ling and A. Ribeiro, "Decentralized dynamic optimization through the alternating direction method of multipliers," *IEEE Trans. Signal Process.*, vol. 62, no. 5, pp. 1185–1197, Mar. 2014.

[45] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Automat. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

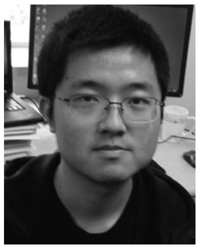
[46] S. Boyd and A. Mutapcic, "Subgradient methods," Stanford Univ., Stanford, CA, USA, Lecture Notes EE364b, vol. 2007, 2006.

[47] N. Parikh *et al.*, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, 2014.

[48] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. New York, NY, USA: Springer, 2011, pp. 185–212.

[49] T. Yucek and H. Arslan, "A survey of spectrum sensing algorithms for cognitive radio applications," *IEEE Commun. Surv. Tuts.*, vol. 11, no. 1, pp. 116–130, Jan.–Mar. 2009.

[50] Y. Chow and T. Lai, "Limiting behavior of weighted sums of independent random variables," *Ann. Probab.*, vol. 1, pp. 810–824, 1973.



Xuanyu Cao received the Bachelor's degree from Shanghai Jiao Tong University, Shanghai, China, in 2013, and the Ph.D. degree from the University of Maryland, College Park, MD, USA, in 2017, both in electrical engineering. He is currently a Postdoctoral Research Associate in the Electrical Engineering Department, Princeton University, Princeton, NJ, USA. His research lies in the intersection (or union) of optimization, game theory, statistical signal processing, probabilistic analysis, and their applications in networked multi-agent systems.

K. J. Ray Liu (F'03) was named a Distinguished Scholar-Teacher of University of Maryland, College Park, MD, USA, in 2007, where he is Christine Kim Eminent Professor of Information Technology. He leads the Maryland Signals and Information Group conducting research encompassing broad areas of information and communications technology with recent focus on smart radios for smart life.

He received the 2016 IEEE Leon K. Kirchmayer Technical Field Award on graduate teaching and mentoring, the IEEE Signal Processing Society 2014 Society Award, and the IEEE Signal Processing Society 2009 Technical Achievement Award. Recognized by Thomson Reuters as a Highly Cited Researcher, he is a Fellow of AAAS. He is a member of IEEE Board of Director as Division IX Director. He was the President of IEEE Signal Processing Society, where he has served as Vice President Publications and Board of Governor. He has also served as the Editor-in-Chief of IEEE SIGNAL PROCESSING MAGAZINE.