Distributed Sparse Linear Regression

Gonzalo Mateos, *Student Member, IEEE*, Juan Andrés Bazerque, *Student Member, IEEE*, and Georgios B. Giannakis, *Fellow, IEEE*

***Abstract*—The Lasso is a popular technique for joint estimation and continuous variable selection, especially well-suited for sparse and possibly under-determined linear regression problems. This paper develops algorithms to estimate the regression coefficients via Lasso when the training data are distributed across different agents, and their communication to a central processing unit is prohibited for e.g., communication cost or privacy reasons. A motivating application is explored in the context of wireless communications, whereby sensing cognitive radios collaborate to estimate the radio-frequency power spectrum density. Attaining different tradeoffs between complexity and convergence speed, three novel algorithms are obtained after reformulating the Lasso into a separable form, which is iteratively minimized using the alternating-direction method of multipliers so as to gain the desired degree of parallelization. Interestingly, the per agent estimate updates are given by simple soft-thresholding operations, and inter-agent communication overhead remains at affordable level. Without exchanging elements from the different training sets, the local estimates *consent* to the global Lasso solution, i.e., the fit that would be obtained if the entire data set were centrally available. Numerical experiments with both simulated and real data demonstrate the merits of the proposed distributed schemes, corroborating their convergence and global optimality. The ideas in this paper can be easily extended for the purpose of fitting related models in a distributed fashion, including the adaptive Lasso, elastic net, fused Lasso and nonnegative garrote.**

***Index Terms*—Distributed linear regression, Lasso, parallel optimization, sparse estimation.**

I. Introduction

CONSIDER the classical setup for linear regression, in which an input vector $\mathbf{x} := [x_1, \ldots, x_p]^T \in \mathbb{R}^p$ is given, and the goal is to predict the real-valued scalar response y, where ^{T} stands for matrix transposition. A linear approximation to the regression function $E[y|\mathbf{x}]$ is adopted to this end, namely $f(\mathbf{x}) = \beta_0 + \mathbf{x}^T\boldsymbol{\beta}$, where $\boldsymbol{\beta} := [\beta_1, \ldots, \beta_p]^T \in \mathbb{R}^p$ is the vector of model coefficients, and the intercept is β_0. Given a training data set $\{y_n, \mathbf{x}_n\}_{n=1}^N$, the model parameters $\{\beta_0, \boldsymbol{\beta}\}$ are to be estimated according to a suitable criterion. The long standing and most popular criterion is least-squares (LS), which i) often times yields unsatisfactory prediction accuracy and ii) fails to provide a parsimonious model estimate whereby only the most relevant predictor variables are selected; see, e.g., [18]. Parsimony is a particularly attractive feature for interpretation purposes, especially in high-dimensional problems, where p is large.

The least-absolute shrinkage and selection operator [32], abbreviated as *Lasso*, is a regularization technique capable of performing both estimation and variable selection. It combines the features of ridge regression and subset selection, the two popular techniques traditionally employed to improve the LS estimates by separately dealing with the aforementioned limitations i) and ii). Upon defining $\mathbf{y} := [y_1 \ldots y_N]^T \in \mathbb{R}^N$ and the regression matrix $\mathbf{X} := [\mathbf{x}_1 \ldots \mathbf{x}_N]^T \in \mathbb{R}^{N\times p}$, the Lasso estimator is the minimizer of the following nonsmooth convex optimization problem

$$\hat{\boldsymbol{\beta}}_{\text{Lasso}} = \arg\min_{\{\beta_0,\boldsymbol{\beta}\}} \frac{1}{2}\,||\mathbf{y} - \mathbf{1}_N\beta_0 - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1 \qquad (1)$$

where $\mathbf{1}_N$ denotes the $N \times 1$ vector of all ones, and $||\boldsymbol{\beta}||_1 := \sum_{i=1}^p |[\boldsymbol{\beta}]_i|$ is the *sparsity-encouraging* ℓ_1-norm of vector $\boldsymbol{\beta}$. The nonnegative parameter λ controls the amount of sparsity (number of nonzero entries in $\hat{\boldsymbol{\beta}}_{\text{Lasso}}$), and is typically chosen via model selection techniques such as cross-validation (CV); see, e.g., [18]. Problem (1) is also known as *basis pursuit denoising*, a term coined by [7] in the context of finding the best sparse signal expansion using an overcomplete basis.

Lasso is equivalent to a quadratic programming (QP) problem; hence, an iterative procedure is required to determine $\hat{\boldsymbol{\beta}}_{\text{Lasso}}$ for a given value of λ. While standard QP solvers can be certainly invoked to this end, an increasing amount of effort has been put recently into developing fast algorithms that capitalize on the unique properties of the Lasso. The LARS algorithm [10] is an efficient scheme for computing the entire path of solutions (corresponding to all values of λ). Coordinate descent algorithms have been shown competitive, even outperforming LARS when p is large, as demonstrated in [13]; see also [36], and the references therein. Other approaches based on variable decoupling have been proposed by [17] and [35]. Since $||\boldsymbol{\beta}||_1$ is nondifferentiable, iterative subgradient methods are also applicable despite their generally slow convergence rate; see [30] for a survey.

In linear regression problems, the training set $\{y_n, \mathbf{x}_n\}_{n=1}^N = \{\mathbf{y}, \mathbf{X}\}$ is typically assumed to be centrally available, so that it can be jointly processed to obtain $\hat{\boldsymbol{\beta}}_{\text{Lasso}}$ in (1). However, collecting all data in a central location or fusion center (FC) may be prohibitive in certain applications. *Distributed* linear regression problems commonly arise with wireless sensor networks (WSNs), where data are inherently scattered across a large geographical area [1], [24]. As sensors are battery operated, transferring all data to an FC (possibly located far away) may be infeasible due to power constraints imposed on the individual nodes. In other cases, such as the Internet or collaborative interlaboratory studies, agents providing private data for

Manuscript received January 29, 2010; accepted June 20, 2010. Date of publication July 01, 2010; date of current version September 15, 2010. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Huaiyu Dai. Work in this paper was supported by the NSF Grants CCF-0830480 and ECCS-0824007. Part of the paper was presented at the International Conference on Acoustics, Speech and Signal Processing, Dallas, TX, March 15-19, 2010.

The authors are with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: mate0058@umn.edu; bazer002@umn.edu; georgios@umn.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.2010.2055862

the purpose of fitting, e.g., a sparse model, may not be willing to share their training data but only the learning results [8], [12].

In lieu of a central controller, it is the agents themselves that are responsible for processing their locally available training sets. The so-termed *in-network* processing introduces additional algorithmic challenges, as the information is not centrally available and can not be "flooded" for massive dissemination. Decentralized linear regression is based on successive refinements of local model parameter estimates maintained at individual agents. In a nutshell, each iteration of this broad class of fully *distributed* algorithms comprises i) a communication step where agents exchange messages with their neighbors and ii) an update step where each agent uses this information to refine its local estimate. Absence of hierarchy and the purely decentralized nature of in-network processing dictate that the collection of local (per agent) estimates should eventually *consent* to the global solution, namely, the parameter estimates that would be obtained if the entire data were centrally available.

Tutorial treatments of related consensus-based distributed approaches can be found in [9] and [23]. Achieving consensus across agents was considered in applications as diverse as vehicle coordination [20], sample-averaging of distributed sensor observations [37], sensing for cognitive radio (CR) [2], [3], and distributed learning in WSNs [12]; see also [24]. A general distributed estimation framework was put forth in [29], which does not require the desired estimator to be expressible in closed form in terms of (weighted) sample averages. Several distributed estimation algorithms are rooted on iterative optimization methods, which capitalize upon the separable structure of the cost defining the desired estimator. The sample mean estimator was formulated in [26] as an optimization problem, and was solved in a distributed fashion using a primal dual approach; see, e.g., [5]. Similarly, the schemes in [25] and [27] are based in incremental (sub)gradient methods [4], [21]. Asynchronous variants for distributed (nonsmooth) optimization were proposed in [22] and [38], and generalized to cope with global convex constraints as well as subgradients affected by stochastic errors [28].

Building on the optimization framework in [29], this paper deals with consensus-based distributed algorithms for the Lasso. The approach entails reformulating (1) as a convex constrained optimization problem, whose structure lends itself naturally to distributed implementation. It is then possible to capitalize upon this favorable structure by resorting to the alternating-direction method of multipliers (AD-MoM), an iterative optimization method that can be traced back to the work by [16] (see also [15]), and which is specially well-suited for parallel processing [5]. This way decentralized recursions are derived to update each agent's local estimate, as well as a vector of dual prices through which agreement across all agents is effected. Three variants are developed which offer the flexibility to choose the most favorable tradeoff between computational complexity and convergence rate. This is possible by capitalizing on the closed-form solution that the Lasso admits when the problem is scalar or orthonormal [18, p. 93]. On a per iteration basis, agents only exchange their current local estimate with their neighbors, so that the training data efficiently percolate to all agents without compromising secrecy. Convergence of the proposed distributed algorithms to the global solution $\hat{\boldsymbol{\beta}}_{\text{Lasso}}$ is also investigated. Finally, a *distributed* CV procedure is developed to select the "best" λ in (1), in the sense of minimizing an estimate of the expected prediction error; see, e.g., [18]. The algorithm exploits "warm starts" to efficiently compute the Lasso path of solutions over a grid of values for λ [13].

The paper outline is as follows. In Section II, the problem of distributed linear regression based on the Lasso is formulated. It is further motivated through a spectrum sensing application for CR networks as investigated in, e.g., [3]. An equivalent reformulation of (2) based on consensus is put forth in Section III-A, which has a structure amenable to distributed implementation via the AD-MoM. This way two distributed algorithms for the Lasso are developed, which respectively entail i) (iteratively) solving a Lasso-type QP per agent (Section III-B), or ii) *cyclic* coordinate-wise local updates based on soft-thresholding operations (Section III-C). *Parallel* updates with soft-thresholding characterize the algorithm presented in Section IV, leading to faster convergence but requiring an off-line matrix inversion per agent. Section V deals with a distributed K-fold CV algorithm, for the purpose of selecting the tuning parameter λ via in-network processing. Numerical tests with both simulated and real data sets are presented in Section VI, which corroborate the convergence of the proposed distributed algorithms as well as their global optimality. Section VII includes a summarizing discussion.

II. Problem Statement With Spectrum Sensing as a Motivating Application

Consider J networked agents that are capable of performing some local computations, as well as exchanging messages among neighbors. An agent should be understood as an abstract entity, possibly representing a sensor node in a WSN, a router monitoring Internet traffic, a hospital or laboratory involved in e.g., a medical study; or a sensing CR from a next-generation mobile communications technology. The network is naturally modeled as an undirected graph $\mathcal{G}(\mathcal{J},\mathcal{E})$, where the set of vertices $\mathcal{J} := \{1,\ldots,J\}$ corresponds to the agents, and the edges in $\mathcal{E}$ represent pairs of agents that can communicate. Agent $j \in \mathcal{J}$ communicates with the single-hop agents in its neighborhood $\mathcal{N}_j$, and the size of the neighborhood is denoted by $|\mathcal{N}_j|$. Global connectivity information can be compactly captured in the symmetric adjacency matrix $\mathbf{E} \in \mathbb{R}^{J\times J}$, with entries $[\mathbf{E}]_{ij} = 1$ if $i \in \mathcal{N}_j$, and $[\mathbf{E}]_{ij} = 0$ otherwise. The graph Laplacian $\mathbf{L} \in \mathbb{R}^{J\times J}$ will be useful henceforth, where $\mathbf{L} := \mathbf{D} - \mathbf{E}$, and $\mathbf{D} := \text{diag}(|\mathcal{N}_1|,\ldots,|\mathcal{N}_J|)$. Graph $\mathcal{G}$ is assumed connected, i.e., there exists a (possibly multihop) path that joins any pair of agents in the network.

Each agent $j \in \mathcal{J}$ has available a local training vector $\mathbf{y}_j \in \mathbb{R}^{N_j}$ and matrix $\mathbf{X}_j \in \mathbb{R}^{N_j\times p}$. Agents collaborate to form the common Lasso estimator (1) in a distributed fashion, which can be rewritten as

$$\hat{\boldsymbol{\beta}}_{\text{Lasso}} = \arg\min_{\{\beta_0,\boldsymbol{\beta}\}} \frac{1}{2}\sum_{j=1}^{J} \|\mathbf{y}_j - \mathbf{1}_N\beta_0 - \mathbf{X}_j\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \quad (2)$$

where $\mathbf{y} := [\mathbf{y}_1^T \ldots \mathbf{y}_J^T]^T$ has size $N := \sum_{j=1}^{J} N_j$, and $\mathbf{X} := [\mathbf{X}_1^T \ldots \mathbf{X}_J^T]^T \in \mathbb{R}^{N\times p}$. Each summand in (2) involves the corresponding agent's data. Although the latter are distributed, the summands are coupled through the global decision variables $\{\beta_0,\ \boldsymbol{\beta}\}$. Each column $\mathbf{x}_i$ of the matrix $\mathbf{X}$ is assumed to be

centered, i.e., $\sum_{n=1}^{N}[\mathbf{x}_i]_n/N = 0$ for all $i = 1,\ldots,p$. Since each agent $j \in \mathcal{J}$ only has available $N_j < N$ entries of each column $\mathbf{x}_i$, removal of the per-agent sample average is not sufficient to remove the nonzero-mean of a column, and thus render it zero mean. However, this does not impose a practical limitation because it is possible to run, e.g., a consensus averaging algorithm once to center each nonzero-mean column of $\mathbf{X}$. After centering, the *globally optimal* intercept estimate is $\hat{\beta}_0 = \mathbf{1}_N^T\mathbf{y}/N$ [32], which can be computed only centrally. But observe that $\hat{\beta}_0 = \mathbf{1}_N^T\mathbf{y}/N = \sum_{j=1}^{J}\mathbf{1}_{N_j}^T\mathbf{y}_j/N = \sum_{j=1}^{J}\bar{y}_j/J$, where $\bar{y}_j := (J/N)\mathbf{1}_{N_j}^T\mathbf{y}_j$ is a local (weighted) average of the entries in $\mathbf{y}_j$. This shows that $\hat{\beta}_0$ can be obtained in a distributed fashion, by consenting on the average of the locally available $\{\bar{y}_j\}_{j=1}^{J}$. Henceforth, β_0 will be omitted for notational simplicity and without loss of generality, using the intercept compensated data $\mathbf{y}_j \leftarrow (\mathbf{y}_j - \mathbf{1}_N\hat{\beta}_0)$ in (2); see also Remark 1.

The objective of this paper is to develop and analyze in terms of convergence, a class of *distributed* algorithms for the Lasso based on in-network processing of the locally available training data. The described setup naturally suggests three characteristics that the algorithms should exhibit i) convergence to the global solution $\hat{\boldsymbol{\beta}}_{\text{Lasso}}$ in (2); ii) processing per agent should be kept as simple and efficient as possible; and iii) communications among agents should be confined to the single-hop neighbors, and avoid exchanges of elements from the different training sets. The following application example further motivates the distributed Lasso algorithms that are developed in the present paper.

A. Cooperative Spectrum Sensing for Cognitive Radio Networks

The proliferation of radio communication systems has led to scarce bandwidth resources and expensive licenses that limit access to emergent wireless applications. Extensive measurement campaigns have evidenced however, that the perceived under-utilization of the spectrum is caused by the current access policy whereby fixed frequency bands are assigned per application. This motivates the development of CRs capable of *sensing the spectrum* and accessing it opportunistically; see, e.g., the tutorial paper [19] and the references therein. A cooperative approach to the sensing task of CR networks can be devised, utilizing a basis expansion model for the spectrum [3]

$$\Phi_r(f) = \sum_{s=1}^{N_s} g_{sr}\Phi_s(f) = \sum_{s=1}^{N_s} g_{sr}\sum_{b=1}^{N_b}\beta_{bs}\psi_b(f) \qquad (3)$$

where the first equality models the power spectrum density (PSD) $\Phi_r(f)$ at frequency f, and at the location of the rth CR, as the superposition of the PSDs $\Phi_s(f)$ generated by N_s sources present. The coefficient g_{sr} represents the channel gain modeling the average propagation loss between the source s and the CR r, and is assumed to be a known function of their distance. The second equality in (3) introduces a basis expansion $\Phi_s(f) = \sum_{b=1}^{N_b}\beta_{bs}\psi_b(f)$, where each source PSD is expressed as a linear combination of rectangular pulses of unit height $\psi_b(f)$. Hence, the parameter β_{bs} represents how much power is emitted by source s in the frequency band spanned by the basis $\psi_b(f)$. In the cooperative scenario, N_r sensing CRs collect smoothed periodogram samples $\{y_{rk}\}_{r=1}^{N_r}$ of the received signal at frequencies $\{f_k\}_{k=1}^{N_f}$. They obtain noisy samples of the received PSD $y_{rk} = \Phi_r(f_k) + \eta_{rk}$, where the noise η_{rk} is modeled as a Gaussian random variable.

The sensing scheme capitalizes on two forms of sparsity. The first one emerges from the narrowband nature of source-PSDs relative to the broad swaths of usable spectrum; i.e., for each source s, $\Phi_s(f)$ spans a few frequency bands so that only a few β_{bs} are nonzero. A second form of sparsity emerges when the location of the sources—which are needed to specify g_{sr}—are unknown. In this case, a grid of *candidate* locations is considered [3], augmenting N_s, so that model (3) becomes linear in the parameters $\{\beta_{bs}\}_{b,s=1}^{N_b,N_s}$. Because the number of sources present is typically unknown, a best subset selection approach entails exponential complexity. All in all, locating the active radios boils down to a variable selection problem, which motivates well employment of the Lasso. Joint estimation of the $\{\beta_{bs}\}$ provides a characterization of the PSD not only across frequency but also in space, which enables identification of the (un)occupied frequency bands at arbitrary geographical locations, and thus facilitates spatial frequency reuse.

Since data $\{y_{rk}\}_{r,k=1}^{N_r,N_f}$ are collected by cooperating CRs at different locations, the estimation of $\{\beta_{bs}\}_{s,b=1}^{N_s,N_b}$ amounts to solving a distributed parameter estimation problem. This demands taking into account the network topology, and devising a protocol to share the data. Communicating the CR measurements to an FC presents scalability issues, because far away nodes require extra power to communicate to the central unit, and extra infrastructure that may not be affordable. Instead, fully decentralized in-network processing is preferred whereby communications are constrained to the single-hop neighborhood.

III. Distributed Lasso Estimation Using Quadratic Programming or Coordinate Descent

In this section, we introduce the distributed quadratic program (DQP-)Lasso algorithm, first going through the algorithmic construction steps and salient features of its operation. The approach includes two main building blocks: i) recast (2) into an equivalent separable form which facilitates distributed implementation; and ii) split the optimization problem into simpler subtasks executed locally at each agent. The algorithm is then simplified into the distributed coordinate descent (DCD-)Lasso, which involves local updates given in closed form.

A. A Consensus-Based Reformulation of the Lasso

To distribute the Lasso cost in (2), consider replacing the *global* variable $\boldsymbol{\beta}$ which couples the per-agent summands with *local* variables $\{\boldsymbol{\beta}_j\}_{j=1}^{J}$ representing candidate estimates of $\boldsymbol{\beta}$ per agent. It is now possible to reformulate (2) as the following convex *constrained* minimization problem:

$$\left\{\hat{\boldsymbol{\beta}}_j\right\}_{j=1}^{J} =: \arg\min_{\{\boldsymbol{\beta}_j\}} \frac{1}{2}\sum_{j=1}^{J}\left[\left\|\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\beta}_j\right\|_2^2 + \frac{2\lambda}{J}\|\boldsymbol{\beta}_j\|_1\right]$$
$$\text{s.t.} \quad \boldsymbol{\beta}_j = \boldsymbol{\beta}_{j'},\ j \in \mathcal{J},\ j' \in \mathcal{N}_j. \qquad (4)$$

The equality constraints directly effect local agreement across each agent's neighborhood. Since the communication graph $\mathcal{G}$ is further assumed *connected*, these constraints also ensure *global* consensus a fortiori, meaning that $\boldsymbol{\beta}_j = \boldsymbol{\beta}_{j'},\ \forall j, j' \in \mathcal{J}$. Indeed, let $P(j,j') : j, j_1, j_2, \ldots, j_n, j'$ be a path on $\mathcal{G}$ that joins an arbitrary pair of agents (j,j'). Because contiguous agents in

the path are neighbors by definition, the corresponding chain of equalities $\boldsymbol{\beta}_j = \boldsymbol{\beta}_{j_1} = \boldsymbol{\beta}_{j_2} = \cdots = \boldsymbol{\beta}_{j_n} = \boldsymbol{\beta}_{j'}$ effected by the constraints in (4) imply $\boldsymbol{\beta}_j = \boldsymbol{\beta}_{j'}$ as desired. Thus, the constraints can be eliminated by replacing all the $\{\boldsymbol{\beta}_j\}$ with a common $\boldsymbol{\beta}$, say, in which case the cost in (4) reduces to the one in (2). This simple argument establishes the following result.

Proposition 1: *If $\mathcal{G}$ is a connected graph, then (2) and (4) are equivalent optimization problems, in the sense that $\hat{\boldsymbol{\beta}}_{\text{Lasso}} = \hat{\boldsymbol{\beta}}_j,\ \forall\ j \in \mathcal{J}$.*

B. Quadratic Programming Distributed Lasso

To tackle (4) in a distributed fashion, we will resort to the alternating-direction method of multipliers (AD-MoM) [15], [16]. To this end, consider adding to problem (4) the auxiliary local variables $\boldsymbol{\gamma} := \{\{\breve{\boldsymbol{\gamma}}_j^{j'}\}_{j' \in \mathcal{N}_j}, \{\bar{\boldsymbol{\gamma}}_j^{j'}\}_{j' \in \mathcal{N}_j}\}_{j \in \mathcal{J}}$, one pair per neighbor. Introducing these new variables (4) is rewritten as

$$\min_{\{\boldsymbol{\beta}_j\}, \boldsymbol{\gamma}} \ \frac{1}{2}\sum_{j=1}^{J}\left[\left\|\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\beta}_j\right\|_2^2 + \frac{2\lambda}{J}\|\boldsymbol{\beta}_j\|_1\right]$$
$$\text{s.t.} \quad \boldsymbol{\beta}_j = \breve{\boldsymbol{\gamma}}_j^{j'}, \boldsymbol{\beta}_{j'} = \bar{\boldsymbol{\gamma}}_j^{j'}, \breve{\boldsymbol{\gamma}}_j^{j'} = \bar{\boldsymbol{\gamma}}_j^{j'}, \quad j \in \mathcal{J},\ j' \in \mathcal{N}_j. \quad (5)$$

The equivalence of (2) and (5), as stated in the following corollary, is immediate because the latter only introduces the auxiliary variables in $\boldsymbol{\gamma}$ to yield an alternative representation of the constraint set in (4).

Corollary 1: *If $\mathcal{G}$ is a connected graph, then (2) and (5) are equivalent optimization problems, in the sense that $\hat{\boldsymbol{\beta}}_{\text{Lasso}} = \hat{\boldsymbol{\beta}}_j,\ \forall\ j \in \mathcal{J}$.*

Corollary 1 establishes that the optimal solutions of (5) correspond to $\hat{\boldsymbol{\beta}}_{\text{Lasso}}$ across all agents. Different from (2) however,(5) has a separable structure that facilitates distributed implementation. To capitalize on this favorable structure, associate Lagrange multipliers $\mathbf{v} := \{\{\breve{\mathbf{v}}_j^{j'}\}_{j' \in \mathcal{N}_j}, \{\bar{\mathbf{v}}_j^{j'}\}_{j' \in \mathcal{N}_j}\}_{j \in \mathcal{J}}$ with the constraints in (5), and form the quadratically augmented Lagrangian function

$$\mathcal{L}_a\left[\{\boldsymbol{\beta}_j\}, \boldsymbol{\gamma}, \mathbf{v}\right] = \frac{1}{2}\sum_{j=1}^{J}\left[\left\|\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\beta}_j\right\|_2^2 + \frac{2\lambda}{J}\|\boldsymbol{\beta}_j\|_1\right]$$
$$+\sum_{j=1}^{J}\sum_{j' \in \mathcal{N}_j}\left[(\breve{\mathbf{v}}_j^{j'})^T(\boldsymbol{\beta}_j - \breve{\boldsymbol{\gamma}}_j^{j'}) + (\bar{\mathbf{v}}_j^{j'})^T(\boldsymbol{\beta}_{j'} - \bar{\boldsymbol{\gamma}}_j^{j'})\right]$$
$$+\frac{c}{2}\sum_{j=1}^{J}\sum_{j' \in \mathcal{N}_j}\left[\|\boldsymbol{\beta}_j - \breve{\boldsymbol{\gamma}}_j^{j'}\|_2^2 + \|\boldsymbol{\beta}_{j'} - \bar{\boldsymbol{\gamma}}_j^{j'}\|_2^2\right]. \quad (6)$$

The constraints $\boldsymbol{\gamma} \in C_{\boldsymbol{\gamma}} := \{\boldsymbol{\gamma} : \breve{\boldsymbol{\gamma}}_j^{j'} = \bar{\boldsymbol{\gamma}}_j^{j'},\ j \in \mathcal{J},\ j' \in \mathcal{N}_j\}$ have not been dualized, and $c > 0$ is a preselected penalty coefficient. The extra quadratic terms in (6) render $\mathcal{L}_a$ *strictly convex* with respect to (w.r.t.) the variables $\{\boldsymbol{\beta}_j\}$ and $\boldsymbol{\gamma}$. This important property will be exploited later on. Other attractive features of the augmented Lagrangian for parallel optimization methods are discussed in [5, Sec. 3.4.4].

The AD-MoM entails an iterative process comprising three steps per iteration $k = 0, 1, 2, \ldots$; see, e.g., [5], [16, p. 253], and further details in the Appendix. The augmented Lagrangian is first minimized w.r.t. the collection $\{\boldsymbol{\beta}_j\}$, considering the auxiliary variables in $\boldsymbol{\gamma}$ and multipliers in $\mathbf{v}$ as fixed parameters. The resulting minimizers define the updates $\{\boldsymbol{\beta}_j(k)\}$ corresponding to iteration k. Note that a unique set of minimizers $\{\boldsymbol{\beta}_j(k)\}$ is guaranteed to exist, from the strict convexity of the augmented Lagrangian. Subsequently, $\mathcal{L}_a$ is minimized w.r.t. $\boldsymbol{\gamma} \in C_{\boldsymbol{\gamma}}$ while keeping all other variables fixed, to yield the updates $\boldsymbol{\gamma}(k)$. Finally, the Lagrange multipliers in $\mathbf{v}$ are updated via dual gradient ascent iterations [5], and the cycle is repeated for the $(k+1)st$ iteration. The aforementioned procedure amounts to a block-coordinate descent method with dual variable updates. At each step while minimizing the augmented Lagrangian, the variables not being updated are treated as fixed and are substituted with their most up to date values.

Reformulating the Lasso problem as (5) renders the augmented Lagrangian in (6) highly decomposable. The separability comes in two flavors, both w.r.t. the variable groups $\{\boldsymbol{\beta}_j\}$ and $\boldsymbol{\gamma}$, as well as across agents $j \in \mathcal{J}$. This in turn leads to highly parallelized, simplified recursions corresponding to the aforementioned three steps. Specifically, it is shown in the Appendix that if the multipliers are initialized to zero, the distributed algorithm reduces to the following updates carried out locally at every agent

$$\mathbf{p}_j(k) = \mathbf{p}_j(k-1) + c\sum_{j' \in \mathcal{N}_j}\left[\boldsymbol{\beta}_j(k) - \boldsymbol{\beta}_{j'}(k)\right] \quad (7)$$

$$\boldsymbol{\beta}_j(k+1) = \arg\min_{\boldsymbol{\beta}_j}\left\{\frac{1}{2}\left\|\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\beta}_j\right\|_2^2 + \frac{\lambda}{J}\|\boldsymbol{\beta}_j\|_1 + \mathbf{p}_j^T(k)\boldsymbol{\beta}_j + c\sum_{j' \in \mathcal{N}_j}\left\|\boldsymbol{\beta}_j - \frac{\boldsymbol{\beta}_j(k) + \boldsymbol{\beta}_{j'}(k)}{2}\right\|_2^2\right\} \quad (8)$$

where $\mathbf{p}_j(k) := 2\sum_{j' \in \mathcal{N}_j}\breve{\mathbf{v}}_j^{j'}(k)$, and all initial values are set to zero.

Recursions (7) and (8) entail local updates, and comprise the distributed quadratic programming Lasso (DQP-Lasso) algorithm tabulated as Algorithm 1. The inherently redundant set of auxiliary variables and multipliers $\{\breve{\boldsymbol{\gamma}}_j^{j'}, \bar{\boldsymbol{\gamma}}_j^{j'}, \bar{\mathbf{v}}_j^{j'}\}$ have been eliminated. Each agent, say the jth, does not need to *separately* keep track of all its multipliers $\{\breve{\mathbf{v}}_j^{j'}(k)\}_{j \in \mathcal{N}_j}$, but only to update the (scaled) sum $\mathbf{p}_j(k)$. In the end, agent j has to store and update only two p-dimensional vectors, namely $\{\boldsymbol{\beta}_j(k), \mathbf{p}_j(k)\}$. A unique feature of this distributed setup is that agents communicate their updated local estimates $\{\boldsymbol{\beta}_j\}$ with their neighbors, in order to carry out the tasks (7) and (8).

Algorithm 1: DQP-Lasso

All agents $j \in \mathcal{J}$ initialize to zero $\{\boldsymbol{\beta}_j(0), \mathbf{p}_j(-1)\}$, and locally run

for $k = 0, 1, \ldots$ **do**

Transmit $\boldsymbol{\beta}_j(k)$ to neighbors in $\mathcal{N}_j$.

Update $\mathbf{p}_j(k)$ via (7).

Update $\boldsymbol{\beta}_j(k+1)$ by solving (8).

end for

The overall operation of the algorithm can be described as follows. During iteration $k+1$, agent j receives the local estimates $\{\boldsymbol{\beta}_{j'}(k)\}_{j' \in \mathcal{N}_j}$ from its neighbors and plugs them into

(7) to evaluate its dual price vector $\mathbf{p}_j(k)$. Subsequently, $\mathbf{p}_j(k)$ is jointly used along with $\{\boldsymbol{\beta}_{j'}(k)\}_{j'\in\mathcal{N}_j}$ and the local training data set $\mathbf{y}_j, \mathbf{X}_j$ to obtain $\boldsymbol{\beta}_j(k+1)$, after solving (8). The $(k+1)$st iteration is concluded after agent j broadcasts $\boldsymbol{\beta}_j(k+1)$ to its neighbors. The local optimization (8) is a QP. In particular, it can be recast as the Lasso in (2) with the substitutions $\boldsymbol{\beta}\longrightarrow\boldsymbol{\beta}_j$, $\lambda\longrightarrow\lambda/J$, and the augmented data

$$\mathbf{y}:=\begin{bmatrix}\mathbf{y}_j\\ \dfrac{c\sum_{j'\in\mathcal{N}_j}\left(\boldsymbol{\beta}_j(k)+\boldsymbol{\beta}_{j'}(k)\right)-\mathbf{p}_j(k)}{\sqrt{c|\mathcal{N}_j|}}\end{bmatrix},\quad \mathbf{X}:=\begin{bmatrix}\mathbf{X}_j\\ \sqrt{c|\mathcal{N}_j|}\mathbf{I}_p\end{bmatrix}.$$

As such, (8) can be iteratively solved using standard optimization routines for quadratic programming, second-order cone programming; or alternatively, using coordinate descent or subgradient algorithms. In summary, the DQP-Lasso entails a global outer iteration to attain consensus (in the index k), and also local inner iterations ran at every agent to minimize (8) and update $\boldsymbol{\beta}_j(k)$ for all $j\in\mathcal{J}$. Per iteration of the consensus loop, there is a communication step whereby agents exchange their updated local estimates across the neighborhood, followed by local dual price vector updates [cf. (7)].

As asserted in the following proposition, the DQP-Lasso algorithm generates a sequence of local iterates $\boldsymbol{\beta}_j(k)$ that converge to the desired Lasso estimate $\hat{\boldsymbol{\beta}}_{\text{Lasso}}$. A proof can be found in the Appendix.

Proposition 2: *Let $\mathcal{G}$ be a connected graph, and consider recursions (7) and (8) that comprise the DQP-Lasso algorithm. Then for any value of the penalty coefficient $c>0$, the iterates $\boldsymbol{\beta}_j(k)$ converge to the Lasso solution [cf. (2)] as $k\to\infty$, i.e.,*

$$\lim_{k\to\infty}\boldsymbol{\beta}_j(k)=\hat{\boldsymbol{\beta}}_{\text{Lasso}},\ \forall\, j\in\mathcal{J}. \tag{9}$$

Formally, if the number of parameters p is greater than the length of the data set N, a unique solution of (2) is not guaranteed for a general design matrix $\mathbf{X}$. Proposition 2 remains valid however, if the right-hand side of (9) is replaced by the set of minima; i.e., $\lim_{k\to\infty}\boldsymbol{\beta}_j(k)\in\arg\min_{\boldsymbol{\beta}}1/2\sum_{j=1}^{J}\|\mathbf{y}_j-\mathbf{X}_j\boldsymbol{\beta}\|_2^2+\lambda\|\boldsymbol{\beta}\|_1$.

From (9), all asymptotic (as N grows large) properties of centralized Lasso carry over to its distributed counterpart developed here. Those include not only the bias, but also weak support consistency as well as estimation consistency, which for the centralized Lasso have been studied in, e.g., [39] and [11]. Specifically for the bias, one can for instance borrow the weighted versions of the ℓ_1-norm in [39], with weights provided by the (distributed) LS estimates in order to ensure asymptotic unbiasedness.

C. Coordinate Descent Distributed Lasso

A simplified version of the DQP-Lasso algorithm is developed in this section, which efficiently tackles the local minimization (8) to update $\boldsymbol{\beta}_j(k)$. Reduced complexity is particularly desirable when the cost of the nodes is a restrictive constraint, since simpler (cheaper) processors can still accomplish the estimation goals. Battery operated nodes can also benefit in this case, because lower complexity and faster convergence rates translate into longer battery lifetime. The rationale behind the new algorithm hinges upon the fact that the solution of (8) does not need to be super accurate, since it is just an intermediate step in the consensus loop defined by the DQP-Lasso. This motivates stopping earlier the inner iteration which solves (8), even after a single minimization step, as detailed next.

In this direction consider running Algorithm 1 as it is given. As previously discussed, many choices are available when it comes to solving (8) iteratively. The coordinate descent algorithm, albeit not the most efficient choice for general optimization problems [4, p. 162], is well suited for the Lasso and related models. Coordinate descent schemes capitalize on the separability of the ℓ_1-norm penalty across the entries of $\boldsymbol{\beta}$, and the resulting simple soft-thresholding solution when β is a scalar. These algorithms also take advantage of the prior knowledge about the sparsity of the estimators sought. Driven by the assessments of its good performance, extensively corroborated via numerical experiments in [13], a coordinate descent algorithm is incorporated here in order to solve the Lasso-type of subproblem (8).

Let $l=0,1,\ldots,$ denote the inner iteration index for the coordinate descent algorithm used to solve (8). For the minimization at step k of the (outer) consensus iteration, the sequence of iterates $\boldsymbol{\beta}_j(k;l)$ are initialized as $\boldsymbol{\beta}_j(k;0):=\boldsymbol{\beta}_j(k)$. At each step l, the scalar coordinates $[\boldsymbol{\beta}_j]_i$ of vector $\boldsymbol{\beta}_j$ are updated cyclically, by solving for $i=1,2,\ldots,p$

$$[\boldsymbol{\beta}_j(k;l+1)]_i=\arg\min_{\beta}\left\{\frac{1}{2}\left\|\mathbf{y}_j^{(-i)}-\mathbf{x}_{ji}\beta\right\|_2^2+\frac{\lambda}{J}|\beta|+[\mathbf{p}_j(k)]_i\beta+c\sum_{j'\in\mathcal{N}_j}\left(\beta-\frac{[\boldsymbol{\beta}_j(k;0)]_i+[\boldsymbol{\beta}_{j'}(k;0)]_i}{2}\right)^2\right\} \tag{10}$$

$$\mathbf{y}_j^{(-i)}:=\mathbf{y}_j-\sum_{i'=1}^{i-1}\mathbf{x}_{ji'}[\boldsymbol{\beta}_j(k;l+1)]_{i'}-\sum_{i'=i+1}^{p}\mathbf{x}_{ji'}[\boldsymbol{\beta}_j(k;l)]_{i'} \tag{11}$$

where $\mathbf{x}_{ji}$ stands for the ith column of the design matrix $\mathbf{X}_j$. Vector $\mathbf{y}_j^{(-i)}$ corresponds to the partial residual error without considering the contribution of the predictor $\mathbf{x}_{ji}$. The usefulness of a coordinate descent approach stems from the fact that the coordinate updates (10) amount to scalar Lasso-type optimizations. Skipping details that can be found in, e.g., [13], the solutions are thus expressible in the closed form

$$[\boldsymbol{\beta}_j(k;l+1)]_i=\left(2c|\mathcal{N}_j|+\|\mathbf{x}_{ji}\|^2\right)^{-1}\times\mathcal{S}\left(\mathbf{x}_{ji}^T\mathbf{y}_j^{(-i)}+\left[c\sum_{j'\in\mathcal{N}_j}\left(\boldsymbol{\beta}_j(k;0)+\boldsymbol{\beta}_{j'}(k;0)\right)-\mathbf{p}_j(k)\right]_i,\frac{\lambda}{J}\right) \tag{12}$$

where $\mathcal{S}(z,\mu):=\text{sign}(z)(|z|-\mu)_+$ is the soft-thresholding operator and $(\cdot)_+:=\max(0,\cdot)$ denotes the projection onto the nonnegative reals. Separability of the nondifferentiable ℓ_1-norm term in (8) is sufficient to guarantee the convergence of (12) to the unique minimizer of (8), as $l\to\infty$ [34]. Hence, the update $\boldsymbol{\beta}_j(k+1):=\lim_{l\to\infty}\boldsymbol{\beta}_j(k;l)$ is well defined, and identical to the one in (8).

As remarked earlier, carrying out the iteration (12) until convergence is overly precise. In the relaxation pursued here, the iteration is instead ended after a single step; i.e., when $l=1$. In this case, the index l can be dropped and (12) simplifies to

$$[\boldsymbol{\beta}_j(k+1)]_i=(2c|\mathcal{N}_j|+\|\mathbf{x}_{ji}\|^2)^{-1}\times\mathcal{S}\left(\mathbf{x}_{ji}^T\mathbf{y}_j^{(-i)}+\left[c\sum_{j'\in\mathcal{N}_j}\left(\boldsymbol{\beta}_j(k)+\boldsymbol{\beta}_{j'}(k)\right)-\mathbf{p}_j(k)\right]_i,\frac{\lambda}{J}\right) \tag{13}$$

where $\mathbf{y}_j^{(-i)}$ is given by (11), and $i = 1, \ldots, p$. A novel algorithm to solve the Lasso in a distributed fashion is obtained after replacing (8) with the simple update rule (13). Per step of the coordinate-wise cycle, a soft-thresholding operation is performed, followed by a proportional shrinkage. The first operation is due to the Lasso penalty, while the second one is due to the quadratic terms in the augmented Lagrangian. The novel scheme is termed distributed coordinate descent Lasso (DCD-Lasso) algorithm, and is tabulated as Algorithm 2. Both the communication step and the dual price vector updates [cf. (7)] are identical for the DQP-Lasso and DCD-Lasso.

Algorithm 2: DCD-Lasso

All agents $j \in \mathcal{J}$ initialize to zero $\{\boldsymbol{\beta}_j(0), \mathbf{p}_j(-1)\}$, and locally run

for $k = 0, 1, \ldots$ **do**

  Transmit $\boldsymbol{\beta}_j(k)$ to neighbors in $\mathcal{N}_j$.

  Update $\mathbf{p}_j(k)$ via (7).

  for $i = 1, \ldots, p$ **do**

    Update $[\boldsymbol{\beta}_j(k+1)]_i$ as in (13).

  end for

end for

A convergence proof for DCD-Lasso will not be provided here, however it is intuitive from the dynamics of the resulting system. Indeed the recursion defined by (13) comprises a linear update followed by a soft-thresholding operator. The linear part can be combined with the dual price updates (7) and put in matrix form. It can be seen that if the parameter c is large enough the eigenvalues of the transition matrix have modulus less than one, which implies a contraction of the error.

The following proposition, to be proved in the Appendix, asserts that upon convergence, DCD-Lasso achieves optimality.

Proposition 3: *Let $\mathcal{G}$ be a connected graph and consider the recursions (7) and (13) that comprise the DCD-Lasso algorithm. Then,*

$$\lim_{k\to\infty} \boldsymbol{\beta}_j(k) = \lim_{k\to\infty} \boldsymbol{\beta}_{j'}(k),\ \forall j, j' \in \mathcal{J} \tag{14}$$

and

$$\lim_{k\to\infty} \boldsymbol{\beta}_j(k) = \hat{\boldsymbol{\beta}}_{\text{Lasso}},\ \forall j \in \mathcal{J}. \tag{15}$$

In words, the local iterates reach consensus asymptotically, and their common limit point corresponds to the solution of (2). As discussed after Proposition 2, for the under-determined case ($N < p$), the local estimates are guaranteed to consent to a minimizer of (2) (from the set of possibly multiple minimizers).

IV. Distributed Lasso

Based on the framework presented in Section III, it is possible to derive yet another improved variant to the DQP-Lasso algorithm. This new solver circumvents the need of an iterative procedure to tackle the per-agent optimizations (8), and in turn yields local estimate updates in closed form. The key is to recognize that the Lasso coefficients are obtained via soft-thresholding not only in the case of a single predictor, but also when the design matrix is orthonormal [32]. Different from DCD-Lasso, the algorithm developed in this section allows to update all coordinates of $\boldsymbol{\beta}_j$ in parallel, potentially leading to faster convergence. Moreover, no relaxation is required to develop the new algorithm that is an instance of the AD-MoM solver. As a result, convergence of all local iterates towards the Lasso can be established along the lines of Proposition 2.

Going back to the equivalent Lasso problem (5), consider an additional group of auxiliary local variables $\{\boldsymbol{\gamma}_j\}_{j=1}^J$, one per agent. Through them, the goal is to split the cost in (5) so that the squared error loss depends on the $\{\boldsymbol{\gamma}_j\}$, while the ℓ_1-norm penalty is a function of the variables $\{\boldsymbol{\beta}_j\}$. This way, the optimizations w.r.t. $\{\boldsymbol{\beta}_j\}$ will be shown to boil down to a Lasso in the orthonormal design case, i.e., problem (1) where $\mathbf{X}^T\mathbf{X} = \mathbf{I}_p$. Solutions are hence given in closed form, implementing soft-thresholding operations; see, e.g., [18, p. 69]. A related scheme was reported for centralized ℓ_1-norm regularized problems in [17]. There, algorithms are developed for image denoising and for compressive sampling-based reconstructions arising in Magnetic Resonance Imaging (MRI), and without connections to the provably convergent AD-MoM pursued here. Upon introducing appropriate constraints $\boldsymbol{\beta}_j = \boldsymbol{\gamma}_j$ that guarantee the equivalence of the formulations, (5) can be recast as

$$\begin{aligned} \min_{\{\boldsymbol{\beta}_j\},\boldsymbol{\gamma}} \quad & \frac{1}{2}\sum_{j=1}^{J}\left[\left\|\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\gamma}_j\right\|_2^2 + \frac{2\lambda}{J}\|\boldsymbol{\beta}_j\|_1\right] \\ \text{s.t.} \quad & \boldsymbol{\beta}_j = \boldsymbol{\gamma}_j, \quad j \in \mathcal{J} \\ & \boldsymbol{\beta}_j = \breve{\boldsymbol{\gamma}}_j^{j'}, \boldsymbol{\beta}_{j'} = \bar{\boldsymbol{\gamma}}_j^{j'}, \breve{\boldsymbol{\gamma}}_j^{j'} = \bar{\boldsymbol{\gamma}}_j^{j'},\ j \in \mathcal{J},\ j' \in \mathcal{N}_j. \end{aligned} \tag{16}$$

Associating additional Lagrange multipliers $\{\mathbf{v}_j\}_{j\in\mathcal{J}}$ to the new constraints present in (16), while redefining the sets $\boldsymbol{\gamma} := \{\boldsymbol{\gamma}_j, \{\breve{\boldsymbol{\gamma}}_j^{j'}\}_{j'\in\mathcal{N}_j}, \{\bar{\boldsymbol{\gamma}}_j^{j'}\}_{j'\in\mathcal{N}_j}\}_{j\in\mathcal{J}}$ and $\mathbf{v} := \{\mathbf{v}_j, \{\breve{\mathbf{v}}_j^{j'}\}_{j'\in\mathcal{N}_j}, \{\bar{\mathbf{v}}_j^{j'}\}_{j'\in\mathcal{N}_j}\}_{j\in\mathcal{J}}$ for notational convenience, the augmented Lagrangian becomes

$$\begin{aligned} \mathcal{L}_a\left[\{\boldsymbol{\beta}_j\}, \boldsymbol{\gamma}, \mathbf{v}\right] = & \frac{1}{2}\sum_{j=1}^{J}\left[\left\|\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\gamma}_j\right\|_2^2 + \frac{2\lambda}{J}\|\boldsymbol{\beta}_j\|_1\right] \\ & + \sum_{j=1}^{J}\mathbf{v}_j^T(\boldsymbol{\beta}_j - \boldsymbol{\gamma}_j) + \frac{c}{2}\sum_{j=1}^{J}\|\boldsymbol{\beta}_j - \boldsymbol{\gamma}_j\|_2^2 \\ & + \sum_{j=1}^{J}\sum_{j'\in\mathcal{N}_j}\left[(\breve{\mathbf{v}}_j^{j'})^T(\boldsymbol{\beta}_j - \breve{\boldsymbol{\gamma}}_j^{j'}) + (\bar{\mathbf{v}}_j^{j'})^T(\boldsymbol{\beta}_{j'} - \bar{\boldsymbol{\gamma}}_j^{j'})\right] \\ & + \frac{c}{2}\sum_{j=1}^{J}\sum_{j'\in\mathcal{N}_j}\left[\|\boldsymbol{\beta}_j - \breve{\boldsymbol{\gamma}}_j^{j'}\|_2^2 + \|\boldsymbol{\beta}_{j'} - \bar{\boldsymbol{\gamma}}_j^{j'}\|_2^2\right]. \end{aligned} \tag{17}$$

Exactly as in Section III, the constraints $\boldsymbol{\gamma} \in C_{\boldsymbol{\gamma}}$ have not been dualized. In order to tackle (16) in a distributed fashion, the AD-MoM is utilized. As a result, the iterates $\{\boldsymbol{\beta}_j(k+1)\}$ are the minimizers of the augmented Lagrangian w.r.t. $\{\boldsymbol{\beta}_j\}$. For the sake of such minimization, the auxiliary variables in $\boldsymbol{\gamma}$ and Lagrange multipliers in $\mathbf{v}$ are treated as fixed parameters, taking on

their respective values at iteration k. From the separable structure in $\mathcal{L}_a$, the pertinent minimization problem decouples into J sub-problems

$$\boldsymbol{\beta}_j(k+1)=\arg\min_{\boldsymbol{\beta}_j}\Bigg\{\frac{\lambda}{J}\|\boldsymbol{\beta}_j\|_1+\mathbf{v}_j^T(k)\boldsymbol{\beta}_j+\frac{c}{2}\|\boldsymbol{\beta}_j-\boldsymbol{\gamma}_j(k)\|_2^2 + \sum_{j'\in\mathcal{N}_j}\left[\breve{\mathbf{v}}_j^{j'}(k)+\bar{\mathbf{v}}_{j'}^{j}(k)\right]^T\boldsymbol{\beta}_j + \frac{c}{2}\sum_{j'\in\mathcal{N}_j}\left[\|\boldsymbol{\beta}_j-\breve{\boldsymbol{\gamma}}_j^{j'}(k)\|_2^2+\|\boldsymbol{\beta}_j-\bar{\boldsymbol{\gamma}}_{j'}^{j}(k)\|_2^2\right]\Bigg\} \tag{18}$$

that can be cast as the Lasso in the orthonormal design case. Due to the variable splitting procedure that led to (16) and the block-coordinate descent nature of the AD-MoM, the "undesirable" coupling term $\left\|\mathbf{y}_j-\mathbf{X}_j\boldsymbol{\gamma}_j\right\|_2^2$ is not present in (18). For these reasons, it is possible to obtain $\boldsymbol{\beta}_j(k+1)$ in closed form as detailed next [cf. (21)].

Deferring the detailed derivations to the Appendix, the AD-MoM solver leads to the following recursions to be run locally at every agent:

$$\mathbf{p}_j(k)=\mathbf{p}_j(k-1)+c\sum_{j'\in\mathcal{N}_j}[\boldsymbol{\beta}_j(k)-\boldsymbol{\beta}_{j'}(k)] \tag{19}$$

$$\mathbf{v}_j(k)=\mathbf{v}_j(k-1)+c[\boldsymbol{\beta}_j(k)-\boldsymbol{\gamma}_j(k)] \tag{20}$$

$$\boldsymbol{\beta}_j(k+1)=[c(2|\mathcal{N}_j|+1)]^{-1}\mathcal{S}\left(c\boldsymbol{\gamma}_j(k)-\mathbf{p}_j(k)-\mathbf{v}_j(k) + c\sum_{j'\in\mathcal{N}_j}\left[\boldsymbol{\beta}_j(k)+\boldsymbol{\beta}_{j'}(k)\right],\frac{\lambda}{J}\right) \tag{21}$$

$$\boldsymbol{\gamma}_j(k+1)=\left[c\mathbf{I}_p+\mathbf{X}_j^T\mathbf{X}_j\right]^{-1}\left(\mathbf{X}_j^T\mathbf{y}_j+c\boldsymbol{\beta}_j(k+1)+\mathbf{u}_j(k)\right). \tag{22}$$

A slight abuse of notation has been introduced in (21), where the vector-valued function $\mathcal{S}:\mathbb{R}^p\times\mathbb{R}\rightarrow\mathbb{R}^p$ performs the coordinate-wise soft thresholding operation defined in Section III, i.e., its ith coordinate is given by $[\mathcal{S}(\mathbf{z},\mu)]_i:=\text{sign}([\mathbf{z}]_i)\left(|[\mathbf{z}]_i|-\mu\right)_+$. Recursions (19)–(22) comprise the novel D-Lasso algorithm, tabulated as Algorithm 3.

Algorithm 3: D-Lasso

All agents $j\in\mathcal{J}$ initialize to zero $\{\boldsymbol{\beta}_j(0),\boldsymbol{\gamma}_j(0),\mathbf{p}_j(-1),\mathbf{v}_j(-1)\}$, and locally run

for $k=0,1,\ldots$ **do**

Transmit $\boldsymbol{\beta}_j(k)$ to neighbors in $\mathcal{N}_j$.

Update $\mathbf{p}_j(k)$ via (19).

Update $\mathbf{v}_j(k)$ via (20).

Update $\boldsymbol{\beta}_j(k+1)$ using (21).

Update $\boldsymbol{\gamma}_j(k+1)$ using (22).

end for

The algorithm entails the following steps. During iteration $k+1$, agent j receives the local estimates $\{\boldsymbol{\beta}_{j'}(k)\}_{j'\in\mathcal{N}_j}$ and plugs them into (19) to evaluate the dual price vector $\mathbf{p}_j(k)$. The new multiplier $\mathbf{v}_j(k)$ is then obtained using the locally available vectors $\{\boldsymbol{\gamma}_j(k),\boldsymbol{\beta}_j(k)\}$. Subsequently, vectors $\{\mathbf{p}_j(k),\mathbf{v}_j(k)\}$ are jointly used along with $\{\boldsymbol{\beta}_{j'}(k)\}_{j'\in\mathcal{N}_j}$ to obtain $\boldsymbol{\beta}_j(k+1)$ via the soft thresholding/proportional shrinkage operation in (21). Different from DCD-Lasso, all coordinates of $\boldsymbol{\beta}_j$ can be updated in parallel avoiding the need of a cycle. Finally, the updated $\boldsymbol{\gamma}_j(k+1)$ is obtained from (22), and requires the previously updated quantities along with the local response vector $\mathbf{y}_j$ and predictor matrix $\mathbf{X}_j$. The $(k+1)$st iteration is concluded after agent j broadcasts $\boldsymbol{\beta}_j(k+1)$ to its neighbors. A few remarks are now in order.

Remark 1 (Accounting for the Intercept): If the intercept β_0 is not removed from (2), one can augment the local parameter vectors $\boldsymbol{\beta}_j$ as well as the matrices $\mathbf{X}_j$ to form $[\beta_{0j}\boldsymbol{\beta}_j^T]^T$ and $[\mathbf{1}_{N_j}\mathbf{X}_j]$, respectively. Then, the D-Lasso algorithm carries over if $\mathcal{S}(\cdot,\mu)$ in (21) is replaced by $\tilde{\mathcal{S}}$, defined by $[\tilde{\mathcal{S}}(\mathbf{z},\mu)]_1:=[\mathbf{z}]_1$ and $[\tilde{\mathcal{S}}(\mathbf{z},\mu)]_i:=\text{sign}([\mathbf{z}]_i)\left(|[\mathbf{z}]_i|-\mu\right)_+$, for $i=2,\ldots,p+1$.

Remark 2 (Communication Cost): In order to separate the ℓ_2-norm loss from the ℓ_1-norm penalty in the Lasso, the constraint $\boldsymbol{\beta}_j=\boldsymbol{\gamma}_j$ needs to be enforced per agent. As a consequence, and in comparison to DQP-Lasso and DCD-Lasso, an additional Lagrange multiplier recursion has to be run [cf. (20)]. In doing so however, no additional communications are required since only local variables are involved in the aforementioned constraint. For the three algorithms developed, only the p scalars in $\boldsymbol{\beta}_j$ have to be broadcasted per iteration. When p is large, major savings can be attained by only exchanging the set of nonzero entries. Further, the inter-agent communication cost does not depend on the size of the local training sets.

Remark 3 (Efficient Local Computations and Load Balancing): Update (22) involves inversion of the $p\times p$ matrix $c\mathbf{I}_p+\mathbf{X}_j^T\mathbf{X}_j$, that may be computationally demanding for sufficiently large p. Fortunately, this operation can be carried out off-line before running the algorithm. Other than that, the updates comprising D-Lasso are extremely simple and solely involve scaling/addition of (eventually sparse) p-dimensional vectors and a soft thresholding operation in (21). More importantly, the matrix inversion lemma can be invoked to obtain $\left[c\mathbf{I}_p+\mathbf{X}_j^T\mathbf{X}_j\right]^{-1}=c^{-1}\left[\mathbf{I}_p-\mathbf{X}_j^T\left(c\mathbf{I}_{N_j}+\mathbf{X}_j\mathbf{X}_j^T\right)^{-1}\mathbf{X}_j\right]$. The dimensions of the matrix to invert become N_j, i.e., the number of locally acquired data. For highly underdetermined ($N\ll p$) regression problems commonly arising, e.g., in genomics, D-Lasso enjoys significant computational savings through the aforementioned matrix inversion identity. One also recognizes that the distributed operation parallelizes the numerical computation across agents: if D-Lasso is simplified to run centrally with all network-wide data at hand, then the matrix to invert has dimension $N=\sum_{j=1}^J N_j$ and increases linearly with the network size J. Beyond a networked scenario, D-Lasso provides an attractive alternative for computational load balancing in timely multi-processor architectures.

Interestingly, a convergence result that parallels Proposition 2 for the DQP-Lasso can be established for the D-Lasso as well. Similar to the former algorithm, the proof in the Appendix amounts to checking that the qualification conditions for the convergence of the AD-MoM [as per Proposition 4.2 in [5 , p. 257]] are satisfied by the optimization problem (16).

Proposition 4: *Let $\mathcal{G}$ be a connected graph and consider recursions (19)–(22) that comprise the D-Lasso algorithm. Then, for any value of the penalty coefficient $c > 0$, the iterates $\boldsymbol{\beta}_j(k)$ converge to the Lasso solution [cf. (2)] as $k \to \infty$, i.e.,* $\lim_{k\to\infty} \boldsymbol{\beta}_j(k) = \hat{\boldsymbol{\beta}}_{\text{Lasso}}$, $\forall\, j \in \mathcal{J}$.

V. Distributed Cross-Validation

The algorithms introduced in Sections III and IV assume knowledge of the ℓ_1-norm penalty parameter λ. This section presents an algorithm to select λ, that performs CV in a distributed fashion. Similar to the algorithms in the previous sections, data are not flooded across the network, and only local exchanges (within the neighborhood) are required.

To select the best parameter λ from a grid of candidate values $\lambda \in \Lambda := \{\lambda_1, \ldots, \lambda_L\}$ ordered as $\lambda_1 > \lambda_2 > \cdots > \lambda_L$, K-fold CV is utilized; see, e.g., [18]. For this purpose, the entire data set collected by all the agents is split into K parts $\{\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2, \ldots, \tilde{\mathbf{y}}_K\}$.

In the distributed scenario, agent j divides its data $\mathbf{y}_j$ in two sub-vectors $\mathbf{y}_j^\kappa := \mathbf{y}_j \cap \tilde{\mathbf{y}}_\kappa$ and $\mathbf{y}_j^{(-\kappa)} := \mathbf{y}_j \setminus \tilde{\mathbf{y}}_\kappa$, where the intersection and exclusion operators $\cap$ and $\setminus$ apply to the set of vector coordinates. Then it sets aside the data in $\mathbf{y}_j^\kappa$, and proceeds to fit a Lasso model based on $\mathbf{y}_j^{(-\kappa)}$. Agents collaborate to run D-Lasso after substituting $\mathbf{y}_j^{(-\kappa)}$ for $\mathbf{y}_j$ and $\mathbf{X}_j^{(-\kappa)}$ for $\mathbf{X}_j$ in (22), where $\mathbf{X}_j^{(-\kappa)}$ is a sub-matrix of $\mathbf{X}_j$ that retains the rows corresponding to $\mathbf{y}_j^{(-\kappa)}$, and removes those corresponding to $\mathbf{y}_j^\kappa$. Vector $\boldsymbol{\beta}_j^{(-\kappa)}(\lambda)$ is obtained after convergence, which corresponds to the optimizer of (16) and therefore of (2). The next step is performed locally and entails forming the prediction error estimate $e_j^\kappa(\lambda) := \|\mathbf{y}_j^\kappa - \mathbf{X}_j^\kappa \boldsymbol{\beta}_j^{(-\kappa)}\|_2^2$, where $\mathbf{X}_j^\kappa$ is defined in correspondence with $\mathbf{y}_j^\kappa$.

This procedure is repeated for all $\lambda = \lambda_1, \ldots, \lambda_L$ using "warm starts" as in [13], and the results are stored in the vector $\mathbf{e}_j^\kappa$ of length L. Observe that even if the per agent variables in Algorithm 3 are initialized to zero, the derivations in the Appendix show that this is not a strict requisite provided the multipliers satisfy the conditions stated after (30). These conditions are guaranteed at convergence, which allows using the limiting values of $\mathbf{p}_j(k)$ and $\boldsymbol{\beta}_j(k)$ obtained for λ_1 when starting the iterations for λ_2, and so on. Simulations demonstrate that convergence of Algorithm 3 with warm starts can be attained after a few iterations.

Upon repeating this procedure for all $\lambda \in \Lambda$ and all $\kappa = 1, 2, \ldots, K$, each agent has available the error vectors $\{\mathbf{e}_j^\kappa\}_{\kappa=1}^K$. The decision on which λ to use is based on the average of these errors across nodes $j \in \mathcal{J}$ and bins $\kappa = 1, 2, \ldots, K$, namely

$$\mathbf{e} := \frac{1}{JK} \sum_{j=1}^{J} \sum_{\kappa=1}^{K} \mathbf{e}_j^\kappa = \frac{1}{J} \sum_{j=1}^{J} \mathbf{e}_j \in \mathbb{R}^L.$$

Note that the inner average across bins can be performed locally to yield $\mathbf{e}_j := 1/K \sum_{\kappa=1}^K \mathbf{e}_j^\kappa$. In principle, the average across agents requires communication of the vectors $\mathbf{e}_j$. However, it can be achieved via consensus algorithms for distributed sample averaging in the same spirit of those in Sections III and IV; see, e.g., [37] and [38].

Once the vector $\mathbf{e} := [e(\lambda_1), e(\lambda_2), \ldots, e(\lambda_L)]$ becomes available to all agents, they can select the best λ by picking the one corresponding to the minimum entry of $\mathbf{e}$, i.e.,

$$\lambda_{\text{dcv}} = \arg \min_{\lambda_l \in \Lambda} \{e(\lambda_l)\}_{l=1}^L.$$

Note that the λ_{dcv} selected by this distributed procedure coincides with its centralized counterpart, that would be obtained by a standard CV scheme when applied to the entire data set. The overall distributed (D)CV procedure is summarized as Algorithm 4. An "one-standard error" rule [18, p. 244] can be used with the DCV procedure since standard errors can be also obtained via consensus averaging.

Algorithm 4: DCV

All agents $j \in \mathcal{J}$ initialize to zero $\{\boldsymbol{\beta}_j(0), \boldsymbol{\gamma}_j(0), \mathbf{p}_j(-1), \mathbf{v}_j(-1)\}$, and locally do

for $\kappa = 0, 1, \ldots$ **do**

Split the local data $\mathbf{y}_j$ into $\mathbf{y}_j^\kappa := \mathbf{y}_j \cap \tilde{\mathbf{y}}_\kappa$ and $\mathbf{y}_j^{(-\kappa)} := \mathbf{y}_j \setminus \tilde{\mathbf{y}}_\kappa$.

Correspondingly split $\mathbf{X}_j$ into $\mathbf{X}_j^\kappa$ and $\mathbf{X}_j^{(-\kappa)}$.

for $\lambda = \lambda_1, \ldots, \lambda_L$ **do**

Obtain $\boldsymbol{\beta}_j^{-\kappa}(\lambda)$ by running D-Lasso using $\{\mathbf{y}_j^{(-\kappa)}, \mathbf{X}_j^{(-\kappa)}\}$ and warm starts.

Compute $e_j^\kappa(\lambda) = \|\mathbf{y}_j^\kappa - \mathbf{X}_j^\kappa \boldsymbol{\beta}_j^{(-\kappa)}\|_2^2$.

end for

end for

Compute $\mathbf{e}_j = 1/K \sum_{\kappa=1}^K \mathbf{e}_j^\kappa$, $e_j^\kappa := [e_j^\kappa(\lambda_1), e_j^\kappa(\lambda_2), \ldots, e_j^\kappa(\lambda_L)]$.

Obtain $\mathbf{e} := [e(\lambda_1), e(\lambda_2), \ldots, e(\lambda_L)]$ by running a consensus averaging algorithm.

Select $\lambda_{\text{dcv}} = \arg\min_{\lambda_l} \{e(\lambda_l)\}$.

If for a particular fold κ, it holds that $\mathbf{y}_j^{(-\kappa)} = \emptyset$, i.e., all data collected by the jth agent are contained in the κth fold, then the agent can remove itself from the corresponding D-Lasso algorithm runs. Upon receiving $\boldsymbol{\beta}_j^{-\kappa}(\lambda)$ from one of its neighbors, it can proceed to compute $e_j^\kappa(\lambda)$. A particular case of this is when the data set is divided into $K = J$ folds, each one containing the data $\mathbf{y}_j$ available at agent j. One agent is then set aside at a time, which constitutes the simplified *leave-one-agent-out CV* algorithm.

VI. Numerical Experiments

A. Study of Prostate Cancer on Real Data

The distributed algorithms for the Lasso are tested here on a real data set corresponding to the prostate cancer study conducted in [31]. As described in e.g., [32], there are eight factors under consideration including log cancer volume (lcavol), log prostate weight (lweight), age, log of the amount of benign prostatic hyperplasia (lbph), seminal vesicle invasion (svi), log of capsular penetration (lcp), Gleason score (gleason), and percent of Gleason scores 4 or 5 (pgg45). These $p = 8$ predictors are measured in $N = 67$ patients together with the response variable, which is the log of the amount of prostate-specific antigen.

These training data are used to fit a Lasso model after standardizing the predictors, where the nonzero entries in the vector estimate $\hat{\boldsymbol{\beta}}_{\text{Lasso}}$ suggest which factors are more relevant in the generation of the antigen. Further details on the scientific background of the regression problem and the characteristics of the data set are given in [18].

For the purpose of illustration, the $N = 67$ samples are divided into $J = 7$ groups, and the data in each group are administrated by an agent. The local training data sets are such that $N_j = 10$ for $j \in [1, 6]$ and $N_7 = 7$. Inter-agent communications take place according to the following adjacency matrix

$$\mathbf{E} := \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

For example, agent 1 communicates with agents 4 and 5 (by convention it also "communicates" with itself).

By running the DCD-Lasso algorithm and using "warm starts", the path of Lasso solutions is computed at 100 different values of the regularization parameter λ. The values are evenly spaced on a logarithmic scale in the interval $[10^{-3}\lambda_{\max}, \lambda_{\max}]$, where $\lambda_{\max} = 61.62$ is the minimum value such that $\hat{\boldsymbol{\beta}}_{\text{Lasso}} \neq \mathbf{0}$ [13]. The penalty coefficient is set to $c = 6$, since several experiments suggested that this value leads to fastest convergence. Indeed, it turns out that c affects the convergence rate of the AD-MoM-based algorithms. However, a formal convergence rate analysis to further justify this choice is challenging, and goes beyond the scope of this paper. After 40 iterations the updates converge to the centralized solution of (1) within a relative error of 4.5×10^{-3}. The path of solutions $\hat{\boldsymbol{\beta}}_1(\lambda)$ obtained by a representative agent is depicted in Fig. 1 (top), where each coefficient $[\hat{\boldsymbol{\beta}}_1]_i, i = 1, \ldots, p$, obtained after convergence, is shown as a function of λ. As λ decreases, so does the amount of shrinkage, and more variables enter the model. The piecewise-linear coefficient profile appears as piecewise-nonlinear because a logarithmic scale is used in the abscissa. The dashed vertical line in Fig. 1 (top) shows the model for $\lambda_{\text{dcv}} = 8.53$, that is selected using the leave-one-agent-out (K =7-fold) CV strategy; see also Section V and the details of the CV procedure in the ensuing tests. The numerical experiment is repeated for D-Lasso which achieves faster convergence, attaining the same result as DCD-Lasso within a relative error of 3.1×10^{-3} after 23 iterations on average. The corresponding plot is hence omitted for brevity.

To further corroborate the convergence of DCD-Lasso and D-Lasso to $\hat{\boldsymbol{\beta}}_{\text{Lasso}}$, both algorithms are run again for $\lambda = \lambda_{\text{dcv}}$. The suggested all-zero initialization is tested against the alternative whereby local estimates are randomly initialized. DCD-Lasso and D-Lasso are also compared with the distributed subgradient methods in [28] and [22], for which equal neighbor combining weights [22], zero initial conditions, and a diminishing stepsize $\alpha(k) = 10^{-2}/k$ are adopted. On a per iteration basis, the global error metric $\epsilon(k) := J^{-1}\sum_{j=1}^{J} \|\hat{\boldsymbol{\beta}}_j(k) - \hat{\boldsymbol{\beta}}_{\text{Lasso}}\|_2^2$ is evaluated for all schemes. The glmnet package for

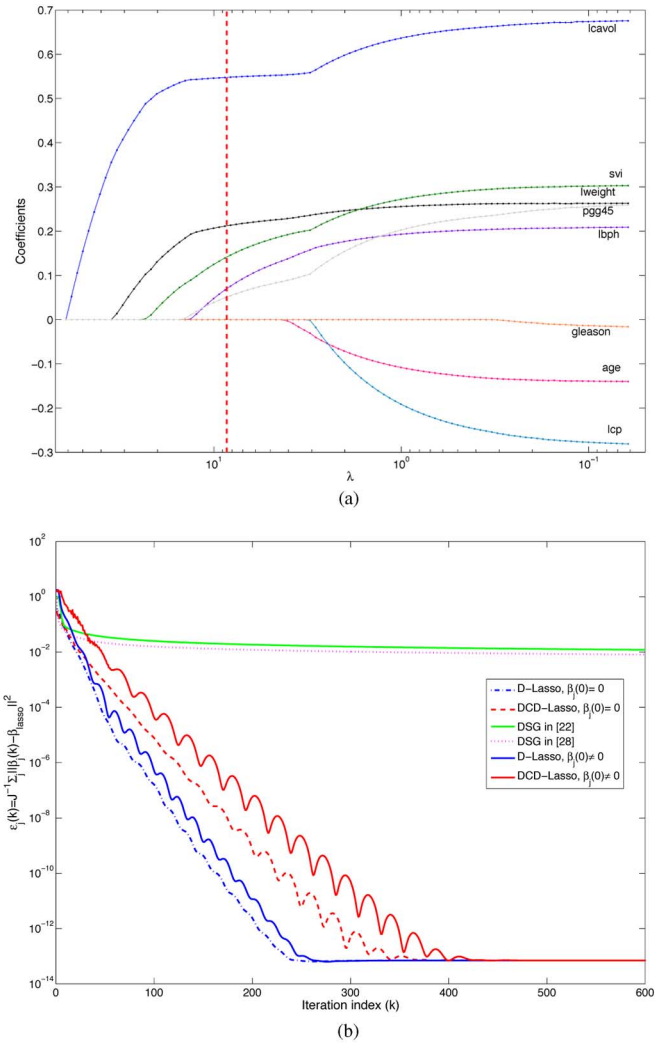

Fig. 1. Prostate cancer data: (top) DCD-Lasso path of solutions as a function of λ. The dashed vertical line shows the model for $\lambda_{\text{dcv}} = 8.53$, that is selected using the leave-one-agent-out CV strategy; (bottom) evolution of the global error $\epsilon(k)$ for DCD-Lasso, D-Lasso and the distributed subgradient (DSG) methods in [28] and [22] ($\lambda = \lambda_{\text{dcv}}$). DCD-Lasso and D-Lasso converge much faster to $\hat{\boldsymbol{\beta}}_{\text{Lasso}}$, especially when the initial conditions are null.

Matlab was utilized to obtain $\hat{\boldsymbol{\beta}}_{\text{Lasso}}$[14], and the resulting errors are depicted in Fig. 1 (bottom) as functions of the iteration index k. The decreasing trend of $\epsilon(k)$ confirms that all local estimates converge to $\hat{\boldsymbol{\beta}}_{\text{Lasso}}$, as stated in Propositions 3 and 4. Also, results in Fig. 1 (bottom) confirm that D-Lasso provides a faster alternative when an off-line matrix inversion is affordable. All-zero initial vectors significantly speed up the algorithms. It is important to remark that additional numerical tests have evidenced that the convergence of DCD-Lasso requires a sufficiently large value of c, while D-Lasso converges for any $c > 0$ as per Proposition 4. With regards to the subgradient methods, the speed of convergence is extremely slow since a descent along a subgradient direction is not effective in setting to zero entries of the local estimates. The (most favorable) minimum ℓ_2-norm subgradient [30] was utilized for the simulations. Nonetheless, the schemes in [22], [28] can tackle general nonsmooth but separable convex problems.

As a final illustration to highlight the consensus property of the algorithms, Fig. 2 shows the evolution of the coefficients

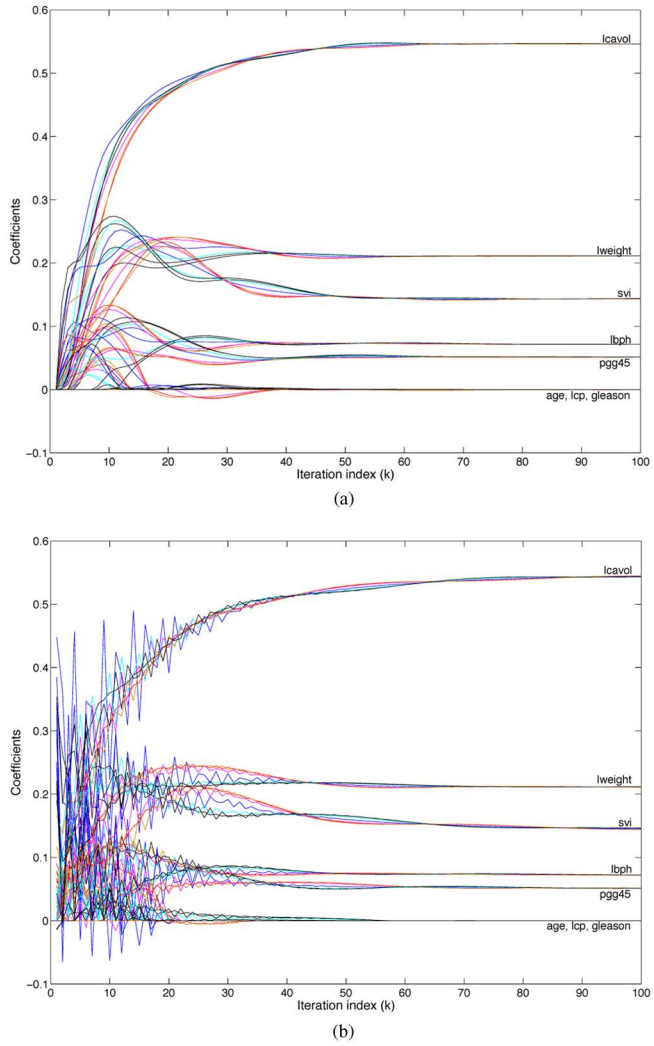

Fig. 2. Prostate cancer data: evolution of the per agent estimates $[\hat{\boldsymbol{\beta}}]_i$, $i = 1, \ldots, p$, for $\lambda = \lambda_{\rm dcv}$: (top) D-Lasso updates; (bottom) DCD-Lasso updates. Consensus is attained after a few iterations.

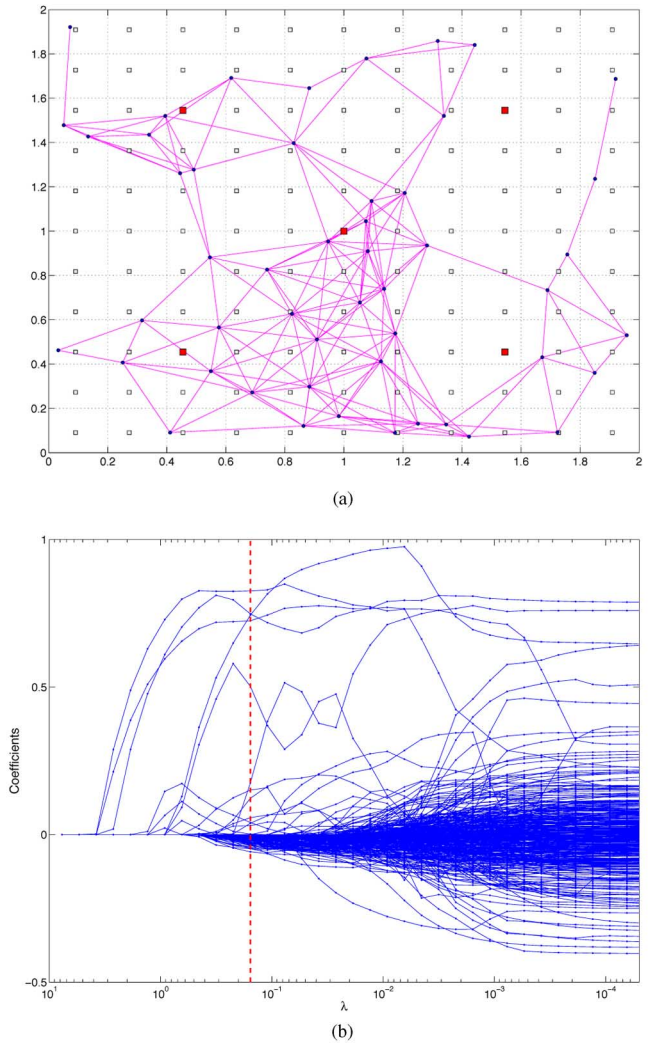

Fig. 3. Spectrum sensing data: (top) network setup for the distributed spectrum sensing task. Transmitters (red squares) are placed on 5 of the possible $N_s = 121$ candidate grid locations (grey squares) over an area of 4 $\rm Km^2$; $N_r = 50$ agents (blue circles) are randomly deployed and two agents can communicate if their Euclidian distance is less than 460 m. The resulting connected communication graph is depicted in magenta; (bottom) path of solutions for the transmit-powers revealed by D-Lasso.

estimated by each agent as the iteration index grows. For each one of the $p = 8$ predictors, $J = 7$ different curves are shown, one per agent. As established in Propositions 3 and 4 all local estimates reach a common limiting value after a transient period.

B. Sparsity-Aware Spectrum Cartography

In order to evaluate DQP-Lasso, DCD-Lasso, and D-Lasso in the context of the spectrum sensing task described in Section II-A, a numerical example will be considered here. Specifically, the spectrum generated by a set of 5 sources is simulated, where each source's PSD $\Phi_s(f)$ corresponds to one of $N_b = 8$ non-overlapping rectangular pulses $\psi_b(f)$ of 10 MHz over a total bandwidth of 80 MHz. The active transmitters are shown in Fig. 3 (top) as red squares. Samples of the PSD field at $N_f = 8$ frequencies are acquired by $N_r = 50$ CRs, corresponding to smoothed periodogram estimates at the frequencies of interest. The CRs collaborate to locate the sources on a rectangular grid of $N_s = 121$ candidate positions in an area of 4 $\rm Km^2$ [represented by gray squares in Fig. 3 (top)]. The sensing radios are deployed uniformly at random in the area of interest; see Fig. 3 (top) where the CRs are denoted by blue circles. Fig. 3 (top) also shows the communication links which connect neighboring agents separated by less than 460 m. The data are generated according to (3), where the gain g_{sr} is modeled as six-coefficients Rayleigh distributed channel with mean depending on the distance between the source $\boldsymbol{x}_s$ and the CR $\boldsymbol{x}_r$. Specifically, this gain is selected as $g_{sr} = \min\{1, (200\ \text{m}/\|\boldsymbol{x}_s - \boldsymbol{x}_r\|_2)^3\}$. This corresponds to a standard multipath wireless fading channel model, in the presence of rich scattering and without line of sight in the $\boldsymbol{x}_s$-to-$\boldsymbol{x}_r$ link. The mean attenuation follows an inverse polynomial path loss law, which captures the dissipation of power due to free-space propagation and other type of signal obstructions, giving rise to diffraction and/or absorption effects. All in all, this setup results in a linear regression problem with $N = N_r N_f = 400$ data samples and $p = N_s N_b = 968$ parameters, where only five of them are nonzero.

The three algorithms are run using "warm starts" to obtain the path of solutions for a range of $L = 40$ decreasing values of

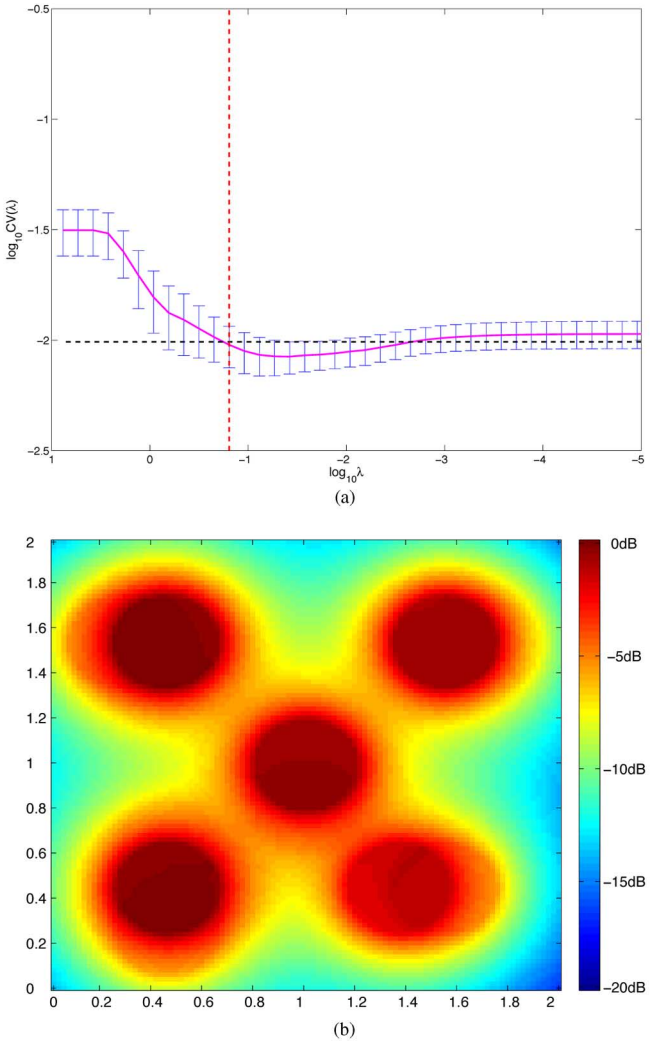

Fig. 4. Spectrum sensing data: (top) CV for the selection of the ℓ_1-norm penalty parameter λ; (bottom) estimated PSD map by the 37th agent; the five active sources are localized.

λ between 500 and 5×10^{-4}. The results of the three schemes converge to the same solution; the path is shown for D-Lasso in Fig. 3 (bottom). Five coefficients emerge at $\lambda = 1$ and become dominant at $\lambda_{\rm dcv} = 0.156$, revealing the positions and frequency bands of the sources present; see also the estimated PSD map in Fig. 4 (bottom). As $\lambda \longrightarrow 0$, the solution tends to the LS estimate, which introduces spurious components in all candidate locations.

The dashed vertical line in Fig. 3 (bottom) represents the data-driven optimal selection of $\lambda_{\rm dcv}$, which is obtained using the CV technique described in Section V. To obtain $\lambda_{\rm dcv}$, the measurements acquired by the $N_r = 50$ CRs are divided in $K = 5$ folds of 10 CRs that cooperate to compute the estimated prediction errors $e(\lambda)$ and their standard errors. Then the optimal $\lambda_{\rm dcv}$ is selected as the largest $\lambda_l \in \Lambda := \{\lambda_1, \ldots, \lambda_L\}$ yielding $e(\lambda_l)$ within one standard error of $\arg\min_{\lambda_l \in \Lambda}\{e(\lambda_l)\}_{l=1}^{L}$. Fig. 4 (top) shows the result of this procedure, where the continuous line represents the estimated prediction errors, and the vertical bars represent their standard errors. The dashed vertical line represents the obtained value of $\lambda_{\rm dcv} = 0.156$.

Finally, Fig. 4 (bottom) depicts the PSD map estimated by a representative CR, which is obtained by plugging into (3) its D-Lasso solution $\hat{\boldsymbol{\beta}}_{37}$ for $\lambda = 0.156$. The figure depicts the power level across the $4\,\text{Km}^2$ area under consideration, aggregated across frequency, with higher decibel levels corresponding to "warm" colors, and lower decibel levels corresponding to "cold" colors. It corroborates that the use of sparsity-aware techniques is effective in separating the most significative coefficients, thus revealing the position of the emitting sources. Because of consensus, after convergence all CRs agree on the same PSD map estimates (one per frequency). After the sensing task is concluded, the network of CRs enters an operational mode in which the PSD map becomes an instrumental aid. Specifically, a CR transmitter uses this information to assess the interference level at its intended receiver, even if the spectrum occupancy at the remote location of the receiver differs from the local interference at the transmitter. Thus, knowing the PSD at any location enables remote CRs to reuse idle bands dynamically. It also enables CRs to adapt their transmit-power so as to minimally interfere with legacy systems.

VII. Discussion

Distributed Lasso algorithms are developed in this paper, that are suitable for sparse linear regression tasks when the training data sets are distributed across collaborating agents. These tools can be applied to field estimation and source localization, as illustrated in the context of cooperative spectrum sensing.

The novel distributed Lasso estimators are implemented via consensus-based in-network processing, whereby agents iteratively refine their local estimates by exchanging low-overhead messages within the neighborhood. Thanks to these inter-agent communications, the training data efficiently percolate throughout the entire network. As a result, the agents' local estimates asymptotically consent on $\hat{\boldsymbol{\beta}}_{\rm Lasso}$ [cf. (2)], the (global) Lasso estimate obtained if all local training data sets were centrally available. The Lasso is reformulated into an equivalent constrained form, whose structure lends itself naturally to distributed implementation via the AD-MoM. Capitalizing on this favorable structure, three algorithmic variants of the distributed Lasso are developed with complementary features. The first one sets the framework for distributed implementation; hence, it is also important from a conceptual perspective. The so-termed QP-Lasso algorithm is applicable when the agents have sufficient computational power to solve a QP per iteration. This however, may be infeasible in distributed estimation applications using WSNs. Second, the DCD-Lasso algorithm relies on cyclic coordinate descent to reduce complexity, but requires careful selection of a step-size parameter to attain convergence. Finally, D-Lasso is developed after separating the ℓ_1-norm penalty from the quadratic term in the Lasso cost, through additional auxiliary optimization variables and suitable constraints. Computational savings become possible by taking advantage of Lasso's closed-form solution in the orthonormal design case. Apart from an off-line matrix inversion, the resulting per agent D-Lasso updates are extremely simple; solely involving linear combinations of vectors and soft thresholding operations. Convergence of the D-Lasso algorithm is also established for all values of the step-size, and local estimates provably consent on $\hat{\boldsymbol{\beta}}_{\rm Lasso}$. An

attractive feature common to all three algorithms is that agents only exchange sparse messages within the neighborhood, and the communication cost is independent of the size of the local training sets.

The framework and techniques introduced here to develop distributed algorithms for the Lasso are readily applicable to related tools as well. These include the adaptive Lasso which guarantees consistency of estimation and variable selection [39]; the nonnegative garrote which served as a precursor of the Lasso [6]; the elastic net for correlated variables [40]; and the smoothness-encouraging fused Lasso [33].

APPENDIX

A. Proof of (7) and (8)

Recall the augmented Lagrangian function in (6), and for notational convenience define $\boldsymbol{\beta} := \{\boldsymbol{\beta}_j\}_{j\in\mathcal{J}}$. The AD-MoM entails three steps per iteration k of the algorithm:

[S1] Local estimate updates:

$$\boldsymbol{\beta}(k+1) = \arg\min_{\boldsymbol{\beta}} \mathcal{L}_a\left[\boldsymbol{\beta}, \boldsymbol{\gamma}(k), \mathbf{v}(k)\right]. \tag{23}$$

[S2] Auxiliary variable updates:

$$\boldsymbol{\gamma}(k+1) = \arg\min_{\boldsymbol{\gamma}\in C_{\boldsymbol{\gamma}}} \mathcal{L}_a\left[\boldsymbol{\beta}(k+1), \boldsymbol{\gamma}, \mathbf{v}(k)\right]. \tag{24}$$

[S3] Lagrange multiplier updates:

$$\breve{\mathbf{v}}_j^{j'}(k+1) = \breve{\mathbf{v}}_j^{j'}(k) + c[\boldsymbol{\beta}_j(k+1) - \breve{\boldsymbol{\gamma}}_j^{j'}(k+1)] \tag{25}$$

$$\bar{\mathbf{v}}_j^{j'}(k+1) = \bar{\mathbf{v}}_j^{j'}(k) + c[\boldsymbol{\beta}_{j'}(k+1) - \bar{\boldsymbol{\gamma}}_j^{j'}(k+1)]. \tag{26}$$

The goal is to show that [S1]–[S3] can be simplified to (7) and (8). Focusing first on [S2], from the decomposable structure of the augmented Lagrangian [cf.(6)] (24) decouples into $\sum_{j=1}^{J}|\mathcal{N}_j|$ quadratic sub-problems

$$\begin{aligned}\breve{\boldsymbol{\gamma}}_j^{j'}(k+1) &= \bar{\boldsymbol{\gamma}}_j^{j'}(k+1)\\ &= \arg\min_{\breve{\boldsymbol{\gamma}}_j^{j'}} \Big\{ -[\breve{\mathbf{v}}_j^{j'}(k) + \bar{\mathbf{v}}_j^{j'}(k)]^T \breve{\boldsymbol{\gamma}}_j^{j'}\\ &\quad + \frac{c}{2}\Big[\|\boldsymbol{\beta}_j(k+1) - \breve{\boldsymbol{\gamma}}_j^{j'}\|_2^2 + \|\boldsymbol{\beta}_{j'}(k+1) - \breve{\boldsymbol{\gamma}}_j^{j'}\|_2^2\Big]\Big\}\end{aligned} \tag{27}$$

which admit the closed-form solutions

$$\begin{aligned}\breve{\boldsymbol{\gamma}}_j^{j'}(k+1) &= \bar{\boldsymbol{\gamma}}_j^{j'}(k+1)\\ &= \frac{1}{2c}[\breve{\mathbf{v}}_j^{j'}(k) + \bar{\mathbf{v}}_j^{j'}(k)] + \frac{1}{2}\left[\boldsymbol{\beta}_j(k+1) + \boldsymbol{\beta}_{j'}(k+1)\right].\end{aligned} \tag{28}$$

Note that in formulating (27), $\bar{\boldsymbol{\gamma}}_j^{j'}$ was eliminated using the constraint $\breve{\boldsymbol{\gamma}}_j^{j'} = \bar{\boldsymbol{\gamma}}_j^{j'}$. Using (28) to eliminate $\breve{\boldsymbol{\gamma}}_j^{j'}(k+1)$ and $\bar{\boldsymbol{\gamma}}_j^{j'}(k+1)$ from (25) and (26) respectively, a simple induction argument establishes that if the initial Lagrange multipliers obey $\breve{\mathbf{v}}_j^{j'}(0) = -\bar{\mathbf{v}}_j^{j'}(0) = \mathbf{0}$, then $\breve{\mathbf{v}}_j^{j'}(k) = -\bar{\mathbf{v}}_j^{j'}(k)$ for all $k \geq 0$ where $j \in \mathcal{J}$ and $j' \in \mathcal{N}_j$. The set $\{\bar{\mathbf{v}}_j^{j'}\}$ of multipliers has been shown redundant, and (28) readily simplifies to

$$\breve{\boldsymbol{\gamma}}_j^{j'}(k+1) = \bar{\boldsymbol{\gamma}}_j^{j'}(k+1) = \frac{1}{2}\left[\boldsymbol{\beta}_j(k+1) + \boldsymbol{\beta}_{j'}(k+1)\right], \quad j \in \mathcal{J},\ j' \in \mathcal{N}_j. \tag{29}$$

It then follows that $\breve{\boldsymbol{\gamma}}_j^{j'}(k) = \breve{\boldsymbol{\gamma}}_{j'}^{j}(k)$ for all $k \geq 0$, an identity that will be used later on. By plugging (29) in (25), the multiplier update becomes

$$\breve{\mathbf{v}}_j^{j'}(k+1) = \breve{\mathbf{v}}_j^{j'}(k) + \frac{c}{2}[\boldsymbol{\beta}_j(k+1) - \boldsymbol{\beta}_{j'}(k+1)], \quad j \in \mathcal{J},\ j' \in \mathcal{N}_j. \tag{30}$$

If $\breve{\mathbf{v}}_j^{j'}(0) = -\breve{\mathbf{v}}_{j'}^{j}(0) = \mathbf{0}$, then the structure of (30) reveals that $\breve{\mathbf{v}}_j^{j'}(k) = -\breve{\mathbf{v}}_{j'}^{j}(k)$ for all $k \geq 0$, where $j \in \mathcal{J}$ and $j' \in \mathcal{N}_j$.

The minimization (23) in [S1] also decouples in J simpler sub-problems, namely [see the equation at the bottom of the page], where in deriving the second equality we used that i) $\breve{\mathbf{v}}_j^{j'}(k) = \bar{\mathbf{v}}_{j'}^{j}(k)$ which follows from the identities $\breve{\mathbf{v}}_j^{j'}(k) = -\bar{\mathbf{v}}_j^{j'}(k)$ and $\breve{\mathbf{v}}_j^{j'}(k) = -\breve{\mathbf{v}}_{j'}^{j}(k)$ established earlier; ii) the definition $\mathbf{p}_j(k) := 2\sum_{j'\in\mathcal{N}_j}\breve{\mathbf{v}}_j^{j'}(k)$; and iii) the identity $\breve{\boldsymbol{\gamma}}_j^{j'}(k) = \bar{\boldsymbol{\gamma}}_{j'}^{j}(k)$ which allows to merge the identical quadratic penalty terms and eliminate both $\breve{\boldsymbol{\gamma}}_j^{j'}(k)$ and $\bar{\boldsymbol{\gamma}}_{j'}^{j}(k)$ using (29). This establishes (8). Finally, note that upon scaling by two the recursions (30) and summing them over $j' \in \mathcal{N}_j$, (7) follows. □

B. Proof of Proposition 2

From the result in Corollary 1, it suffices to show that the iterates $\boldsymbol{\beta}_j(k)$ generated by (8) converge to $\hat{\boldsymbol{\beta}}_j$ in (5). To this end, observe that (5) can be written as [5, eq. 4.77, p. 255]

$$\begin{aligned}\min_{\mathbf{x},\mathbf{z}} \quad & G_1(\mathbf{x}) + G_2(\mathbf{z})\\ \text{s.t.} \quad & \mathbf{x} \in C_1,\ \mathbf{z} \in C_2,\ \mathbf{A}\mathbf{x} = \mathbf{z}\end{aligned}$$

with the definitions

$$\mathbf{x} := [\boldsymbol{\beta}_1^T, \ldots, \boldsymbol{\beta}_J^T]^T, \quad C_1 := \mathbb{R}^{Jp}$$

$$\begin{aligned}\boldsymbol{\beta}_j(k+1) &= \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2}\left[\|\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\beta}_j\|_2^2 + \frac{2\lambda}{J}\|\boldsymbol{\beta}_j\|_1\right] + \sum_{j'\in\mathcal{N}_j}\left[\breve{\mathbf{v}}_j^{j'}(k) + \bar{\mathbf{v}}_{j'}^{j}(k)\right]^T\boldsymbol{\beta}_j \right.\\ &\qquad \left. + \frac{c}{2}\sum_{j'\in\mathcal{N}_j}\left[\|\boldsymbol{\beta}_j - \breve{\boldsymbol{\gamma}}_j^{j'}(k)\|_2^2 + \|\boldsymbol{\beta}_j - \bar{\boldsymbol{\gamma}}_{j'}^{j}(k)\|_2^2\right]\right\}\\ &= \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2}\left[\|\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\beta}_j\|_2^2 + \frac{2\lambda}{J}\|\boldsymbol{\beta}_j\|_1\right] + \mathbf{p}_j^T(k)\boldsymbol{\beta}_j + c\sum_{j'\in\mathcal{N}_j}\left\|\boldsymbol{\beta}_j - \frac{\boldsymbol{\beta}_j(k) + \boldsymbol{\beta}_{j'}(k)}{2}\right\|_2^2\right\}\end{aligned}$$

$$\mathbf{z} := [\breve{\boldsymbol{\gamma}}^T, \bar{\boldsymbol{\gamma}}^T]^T, \quad C_2 := C_\gamma$$

$$G_1(\mathbf{x}) := \frac{1}{2}\sum_{j=1}^{J}\left[||\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\beta}_j||_2^2 + \frac{2\lambda}{J}||\boldsymbol{\beta}_j||_1\right], \quad G_2(\mathbf{z}) := 0$$

where $\breve{\boldsymbol{\gamma}} := \left[(\breve{\boldsymbol{\gamma}}_1^{i_1(1)})^T, \ldots, (\breve{\boldsymbol{\gamma}}_1^{i_1(|\mathcal{N}_1|)})^T, \ldots, (\breve{\boldsymbol{\gamma}}_J^{i_J(1)})^T, \ldots, (\breve{\boldsymbol{\gamma}}_J^{i_J(|\mathcal{N}_J|)})^T\right]^T$ and likewise for $\bar{\boldsymbol{\gamma}}$. The integer valued functions $i_j : \{1, \ldots, |\mathcal{N}_j|\} \to \mathcal{J}$ are such that $i_j(j')$ is the index of the j'th neighbor of agent j. The linear constraint matrix is $\mathbf{A} = [\mathbf{A}_1^T \mathbf{A}_2^T]^T$, where ($\otimes$ denotes Kronecker product)

$$\mathbf{A}_1 := \begin{bmatrix} \mathbf{A}_{11} \\ \vdots \\ \mathbf{A}_{1J} \end{bmatrix}, \quad \mathbf{A}_{1j} := \left(\mathbf{1}_{|\mathcal{N}_j|}\boldsymbol{\nu}_j^T\right) \otimes \mathbf{I}_p,\ j \in \mathcal{J}$$

$$\mathbf{A}_2 := \begin{bmatrix} \mathbf{A}_{21} \\ \vdots \\ \mathbf{A}_{2J} \end{bmatrix}, \quad \mathbf{A}_{3j} := \begin{bmatrix} \boldsymbol{\nu}_{i_j(1)}^T \\ \vdots \\ \boldsymbol{\nu}_{i_j(|\mathcal{N}_j|)}^T \end{bmatrix} \otimes \mathbf{I}_p,\ j \in \mathcal{J}$$

and $\boldsymbol{\nu}_j$ is the jth vector in the canonical basis for $\mathbb{R}^J$. Using these definitions, it is straightforward to check that G_1 and G_2 are convex functions, C_1 and C_2 are nonempty polyhedral sets, and $\mathbf{A}^T\mathbf{A}$ is invertible (because $\mathbf{A}$ is full column rank). By virtue of [5, Proposition 4.2, p. 256], for every value of $c > 0$ the iterates generated by (23) in [S1] converge to the optimal solution $\hat{\boldsymbol{\beta}}_j = \hat{\boldsymbol{\beta}}_{\text{Lasso}},\ j \in \mathcal{J}$ of problem (5). Because (23) simplifies to (8), the same is true for the iterates generated by Algorithm 1. □

C. Proof of Proposition 3

Let $\bar{\boldsymbol{\beta}}_j := \lim_{k\to\infty}\boldsymbol{\beta}_j(k)$ and $\bar{\mathbf{p}}_j := \lim_{k\to\infty}\mathbf{p}_j(k)$. From (7), we have $\sum_{j'\in\mathcal{N}_j}[\bar{\boldsymbol{\beta}}_j - \bar{\boldsymbol{\beta}}_{j'}] = \mathbf{0}\ \forall j \in \mathcal{J}$. This system of equations implies that the supervector $\bar{\boldsymbol{\beta}} := [\bar{\boldsymbol{\beta}}_1^T, \ldots, \bar{\boldsymbol{\beta}}_J^T]^T$ belongs to the nullspace of $\mathbf{L}$, the Laplacian of the network graph $\mathcal{G}$. This guarantees, under the assumption that the network is connected, that $\bar{\boldsymbol{\beta}}_1 = \bar{\boldsymbol{\beta}}_2 = \cdots = \bar{\boldsymbol{\beta}}_J$, i.e., the consensus result in (14) holds. Hence, it is possible to proceed and define $\bar{\boldsymbol{\beta}} := \bar{\boldsymbol{\beta}}_j$.

The rest of the proof involves defining two vectors $\boldsymbol{\mu}$ and $\boldsymbol{\rho}$, which together with $\bar{\boldsymbol{\beta}}$ satisfy the Karush-Kuhn-Tucker (KKT) conditions of optimality [4] for problem (2). In this direction, consider the matrix $\mathbf{X}_j^T\mathbf{X}_j$ and its decomposition in its strict lower triangular, strict upper triangular, and diagonal parts, $\mathbf{L}_j$, $\mathbf{U}_j$, and $\mathbf{D}_j$, respectively. Consider as well the auxiliary vectors $\boldsymbol{\theta}_j := \mathbf{X}_j^T\mathbf{y}_j - \mathbf{U}_j\bar{\boldsymbol{\beta}} - \mathbf{L}_j\bar{\boldsymbol{\beta}} + 2c|\mathcal{N}_j|\bar{\boldsymbol{\beta}} - \bar{\mathbf{p}}_j$. In the limit as $k \to \infty$, (13) can be written in matrix-vector form as $(2c|\mathcal{N}_j|\mathbf{I}_p + \mathbf{D}_j)\bar{\boldsymbol{\beta}} = \mathcal{S}(\boldsymbol{\theta}_j, \lambda/J)$. Proceed by defining, $\boldsymbol{\psi}_j := \boldsymbol{\theta}_j - (2c|\mathcal{N}_j|\mathbf{I}_p + \mathbf{D}_j)\bar{\boldsymbol{\beta}}$. The strict positivity of the diagonal elements of $\mathbf{D}_j$ together with the input-output relationship of the soft thresholding operator $\mathcal{S}$ yield

$$[\boldsymbol{\psi}_j]_i = \begin{cases} \frac{\lambda}{J}, & [\bar{\boldsymbol{\beta}}]_i > 0 \\ \frac{-\lambda}{J}, & [\bar{\boldsymbol{\beta}}]_i < 0 \\ \xi_{ji} \text{ such that } |\xi_{ji}| \le \frac{\lambda}{J}, & [\bar{\boldsymbol{\beta}}]_i = 0. \end{cases} \tag{31}$$

Given (31), it is possible to define the vectors $\boldsymbol{\mu} := 1/2\left(\lambda\mathbf{1}_p + \sum_{j=1}^{J}\boldsymbol{\psi}_j\right)$ and $\boldsymbol{\rho} := 1/2\left(\lambda\mathbf{1}_p - \sum_{j=1}^{J}\boldsymbol{\psi}_j\right)$ and show that they satisfy the following properties: i) $\boldsymbol{\mu}, \boldsymbol{\rho} \succeq 0$ (coordinatewise); ii) $[\boldsymbol{\mu}]_i = 0$, if $[\bar{\boldsymbol{\beta}}]_i < 0$; iii) $[\boldsymbol{\rho}]_i = 0$, if $[\bar{\boldsymbol{\beta}}]_i > 0$; iv) $\boldsymbol{\mu} + \boldsymbol{\rho} = \lambda\mathbf{1}_p$; and v) $\sum_{j=1}^{J}\mathbf{X}_j^T\mathbf{X}_j\bar{\boldsymbol{\beta}} - \sum_{j=1}^{J}\mathbf{X}_j^T\mathbf{y}_j + \boldsymbol{\rho} - \boldsymbol{\mu} = \mathbf{0}$. Properties i)–iii) follow after adding up the result in (31) w.r.t. $j = 1, 2, \ldots, J$. Property iv) is readily checked from the definitions of $\boldsymbol{\mu}$ and $\boldsymbol{\rho}$. In order to show v), observe first that by plugging $\boldsymbol{\theta}_j$ into $\boldsymbol{\psi}_j$ one obtains $\boldsymbol{\psi}_j = \mathbf{X}_j^T\mathbf{y}_j - (\mathbf{L}_j + \mathbf{D}_j + \mathbf{U}_j)\bar{\boldsymbol{\beta}} - \bar{\mathbf{p}}_j$ after canceling and rearranging terms. Summing up the last equations w.r.t. $j = 1, 2, \ldots, J$ yields

$$\sum_{j=1}^{J}(\mathbf{L}_j + \mathbf{D}_j + \mathbf{U}_j)\bar{\boldsymbol{\beta}} - \sum_{j=1}^{J}\mathbf{X}_j\mathbf{y}_j + \sum_{j=1}^{J}\boldsymbol{\psi}_j = \sum_{j=1}^{J}\bar{\mathbf{p}}_j = \mathbf{0} \tag{32}$$

where the identity $\sum_{j=1}^{J}\bar{\mathbf{p}}_j = \mathbf{0}$ results from the definition of $\mathbf{p}_j(k) := 2\sum_{j'\in\mathcal{N}_j}\breve{\mathbf{v}}_j^{j'}(k)$, and the symmetry properties of $\breve{\mathbf{v}}_j^{j'}(k)$ that were established after (30). To prove that (32) and the equation in property v) are equivalent, confirm first the identity $\mathbf{L}_j + \mathbf{D}_j + \mathbf{U}_j = \mathbf{X}_j^T\mathbf{X}_j$ from the definition of $\mathbf{L}_j$, $\mathbf{D}_j$, and $\mathbf{U}_j$. Also, from the definition of $\boldsymbol{\mu}$ and $\boldsymbol{\rho}$ it follows that $\sum_{j=1}^{J}\boldsymbol{\psi}_j = \boldsymbol{\mu} - \boldsymbol{\rho}$.

The proof is concluded by noticing that properties i)–v) are indeed the KKT conditions for the following optimization problem

$$\min_{\boldsymbol{\beta}} \quad \frac{1}{2}\sum_{j=1}^{J}||\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\beta}||_2^2 + \lambda\sum_{i=1}^{p}t_i, \quad \text{s.t.} \quad -t_i \le [\bar{\boldsymbol{\beta}}]_i \le t_i$$

that is equivalent to problem (2). □

D. Proof of (19)–(22)

Recall the augmented Lagrangian function in (17), and let $\boldsymbol{\beta} := \{\boldsymbol{\beta}_j\}_{j\in\mathcal{J}}$ for notational simplicity. When used to solve (16), the three steps in AD-MoM are given by:

[S1] Local estimate updates:

$$\boldsymbol{\beta}(k+1) = \arg\min_{\boldsymbol{\beta}}\mathcal{L}_a\left[\beta, \boldsymbol{\gamma}(k), \mathbf{v}(k)\right]. \tag{33}$$

[S2] Auxiliary variable updates:

$$\boldsymbol{\gamma}(k+1) = \arg\min_{\boldsymbol{\gamma}\in C_{\boldsymbol{\gamma}}}\mathcal{L}_a\left[\boldsymbol{\beta}(k+1), \boldsymbol{\gamma}, \mathbf{v}(k)\right]. \tag{34}$$

[S3] Multiplier updates:

$$\mathbf{v}_j(k+1) = \mathbf{v}_j(k) + c[\boldsymbol{\beta}_j(k+1) - \boldsymbol{\gamma}_j(k+1)] \tag{35}$$

$$\breve{\mathbf{v}}_j^{j'}(k+1) = \breve{\mathbf{v}}_j^{j'}(k) + c[\boldsymbol{\beta}_j(k+1) - \breve{\boldsymbol{\gamma}}_j^{j'}(k+1)] \tag{36}$$

$$\bar{\mathbf{v}}_j^{j'}(k+1) = \bar{\mathbf{v}}_j^{j'}(k) + c[\boldsymbol{\beta}_{j'}(k+1) - \bar{\boldsymbol{\gamma}}_j^{j'}(k+1)]. \tag{37}$$

Clearly, (35) coincides with (20) so the work left amounts to establishing (19), (21) and (22).

Focusing first on [S2], observe that (17) is separable across the collection of variables $\{\boldsymbol{\gamma}_j\}$ and $\{\breve{\boldsymbol{\gamma}}_j^{j'}, \bar{\boldsymbol{\gamma}}_j^{j'}\}$ that comprise $\boldsymbol{\gamma}$. The minimization w.r.t. the latter group is identical to (27); hence, the solutions are given by (28). If the initial Lagrange multipliers obey $\breve{\mathbf{v}}_j^{j'}(0) = -\bar{\mathbf{v}}_j^{j'}(0) = \mathbf{0}$, then exactly as before $\breve{\mathbf{v}}_j^{j'}(k) = -\bar{\mathbf{v}}_j^{j'}(k)$ and $\breve{\mathbf{v}}_j^{j'}(k) = -\breve{\mathbf{v}}_{j'}^{j}(k)$ for all $k \ge 0$, where $j \in \mathcal{J}$ and $j' \in \mathcal{N}_j$. As a result, the updates for $\{\breve{\boldsymbol{\gamma}}_j^{j'}, \bar{\boldsymbol{\gamma}}_j^{j'}\}$ simplify to (29) and the nonredundant multipliers $\breve{\mathbf{v}}_j^{j'}(k)$ are given

$$\boldsymbol{\beta}_j(k+1) = \arg\min_{\boldsymbol{\beta}_j} \left\{ \frac{\lambda}{J}\|\boldsymbol{\beta}_j\|_1 + \mathbf{v}_j^T(k)\boldsymbol{\beta}_j + \frac{c}{2}\|\boldsymbol{\beta}_j - \boldsymbol{\gamma}_j(k)\|_2^2 + \sum_{j'\in\mathcal{N}_j}\left[\breve{\mathbf{v}}_j^{j'}(k) + \bar{\mathbf{v}}_{j'}^{j}(k)\right]^T\boldsymbol{\beta}_j \right.$$
$$\left. + \frac{c}{2}\sum_{j'\in\mathcal{N}_j}\left[\|\boldsymbol{\beta}_j - \breve{\boldsymbol{\gamma}}_j^{j'}(k)\|_2^2 + \|\boldsymbol{\beta}_j - \bar{\boldsymbol{\gamma}}_{j'}^{j}(k)\|_2^2\right]\right\}$$
$$= \arg\min_{\boldsymbol{\beta}_j} \left\{ \frac{\lambda}{J}\|\boldsymbol{\beta}_j\|_1 + \mathbf{v}_j^T(k)\boldsymbol{\beta}_j + \frac{c}{2}\|\boldsymbol{\beta}_j - \boldsymbol{\gamma}_j(k)\|_2^2 + \mathbf{p}_j^T(k)\boldsymbol{\beta}_j + c\sum_{j'\in\mathcal{N}_j}\left\|\boldsymbol{\beta}_j - \frac{\boldsymbol{\beta}_j(k)+\boldsymbol{\beta}_{j'}(k)}{2}\right\|_2^2\right\} \quad (38)$$

by (30). The remaining minimization (34) w.r.t. $\{\boldsymbol{\gamma}_j\}$ decouples into J quadratic sub-problems [cf. (17)], i.e.,

$$\boldsymbol{\gamma}_j(k+1) = \arg\min_{\boldsymbol{\gamma}_j}\left\{\frac{1}{2}\left\|\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\gamma}_j\right\|_2^2 - \mathbf{v}_j^T(k)\boldsymbol{\gamma}_j + \frac{c}{2}\|\boldsymbol{\beta}_j(k+1) - \boldsymbol{\gamma}_j\|_2^2\right\}$$

which admit the closed-form solutions in (22).

Towards obtaining the updates for the local variables in $\boldsymbol{\beta}$, the optimization (33) in [S1] can be split into J sub-problems as well [see (38), shown at the top of the page], where to arrive at the second equality, we used: i) (29) to eliminate $\breve{\boldsymbol{\gamma}}_j^{j'}(k)$ and $\bar{\boldsymbol{\gamma}}_{j'}^{j}(k)$ which are identical for all $k \geq 0$; and ii) the aforementioned Lagrange multiplier identities ($\bar{\mathbf{v}}_{j'}^{j}(k) = -\breve{\mathbf{v}}_{j'}^{j}(k) = \breve{\mathbf{v}}_j^{j'}(k)$) to deduce that

$$\sum_{j'\in\mathcal{N}_j}\left[\breve{\mathbf{v}}_j^{j'}(k) + \bar{\mathbf{v}}_{j'}^{j}(k)\right] = 2\sum_{j'\in\mathcal{N}_j}\breve{\mathbf{v}}_j^{j'}(k) := \mathbf{p}_j(k). \quad (39)$$

Also note that upon scaling by two the recursions (30) and summing them over $j' \in \mathcal{N}_j$, (19) follows. Back to establishing (21), use a simple complete of squares argument to recast (38) as

$$\boldsymbol{\beta}_j(k+1) = \arg\min_{\boldsymbol{\beta}_j}\left\{\lambda_j\|\boldsymbol{\beta}_j\|_1 + \frac{1}{2}\left\|\boldsymbol{\beta}_j - \frac{c\boldsymbol{\gamma}_j(k) - \mathbf{p}_j(k)}{c(2|\mathcal{N}_j|+1)} - \frac{c\sum_{j'\in\mathcal{N}_j}\left[\boldsymbol{\beta}_j(k) + \boldsymbol{\beta}_{j'}(k)\right] - \mathbf{v}_j(k)}{c(2|\mathcal{N}_j|+1)}\right\|_2^2\right\} \quad (40)$$

where $\lambda_j := \lambda/(Jc(2|\mathcal{N}_j|+1))$. Interestingly, (40) amounts to a Lasso for the orthonormal design special case, which admits the closed-form solution in the right-hand side of (21). □

E. Proof of Proposition 4

Exactly as in the proof of Proposition 2, observe that (16) can be written as

$$\min_{\mathbf{x},\mathbf{z}} \quad G_1(\mathbf{x}) + G_2(\mathbf{z})$$
$$\text{s.t.} \quad \mathbf{x}\in C_1,\ \mathbf{z}\in C_2,\ \mathbf{A}\mathbf{x} = \mathbf{z}$$

with the same definitions used therein except for the modifications $\mathbf{z} = [\boldsymbol{\gamma}_1^T, \ldots, \boldsymbol{\gamma}_J^T, \breve{\boldsymbol{\gamma}}^T, \bar{\boldsymbol{\gamma}}^T]^T$ and

$$G_1(\mathbf{x}) = \frac{\lambda}{J}\sum_{j=1}^{J}\|\boldsymbol{\beta}_j\|_1, \quad G_2(\mathbf{z}) = \frac{1}{2}\sum_{j=1}^{J}\|\mathbf{y}_j - \mathbf{X}_j\boldsymbol{\gamma}_j\|_2^2.$$

The linear constraint matrix in this case is adapted to $\mathbf{A} \leftarrow [\mathbf{I}_{Jp}\ \mathbf{A}^T]^T$, in order to accommodate the additional constraints $\boldsymbol{\beta}_j = \boldsymbol{\gamma}_j$, $j \in \mathcal{J}$ present in (16). Because G_1 and G_2 are clearly convex functions and $\mathbf{A}^T\mathbf{A}$ is still invertible, the arguments used in the proof of Proposition 2 lead to the desired result. □

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Comput. Netw.*, vol. 38, pp. 393–422, Mar. 2002.

[2] S. Barbarossa, G. Scutari, and T. Battisti, "Cooperative sensing for cognitive radio using decentralized projection algorithms," in *Proc. Workshop Signal Processing Advances in Wireless Communications*, Perugia, Italy, Jun. 2009, pp. 116–120.

[3] J. A. Bazerque and G. B. Giannakis, "Distributed spectrum sensing for cognitive radio networks by exploiting sparsity," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1847–1862, Mar. 2010.

[4] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA: Athena Scientific, 1999.

[5] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, 2nd ed. Belmont, MA: Athena Scientific, 1999.

[6] L. Breiman, "Better subset regression using the nonnegative garrote," *Technometrics*, vol. 37, pp. 373–384, Nov. 1995.

[7] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, pp. 33–61, 1998.

[8] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *ACM SIGKDD Explorations*, vol. 4, pp. 28–34, 2002.

[9] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," Mar. 2010 [Online]. Available: http://arxiv.org/abs/1003.5309

[10] B. Efron, T. Hastie, I. M. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Stat.*, vol. 32, pp. 407–499, 2004.

[11] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *J. Amer. Stat. Assoc.*, vol. 96, pp. 1348–1360, 2001.

[12] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *J. Mach. Learn. Res.*, vol. 11, pp. 1663–1707, May 2010.

[13] J. Friedman, T. Hastie, H. Hofling, and R. Tibshirani, "Pathwise coordinate optimization," *Ann. Appl. Stat.*, vol. 1, pp. 302–332, 2007.

[14] J. Friedman, T. Hastie, and R. Tibshirani, "Regularized paths for generalized linear models via coordinate descent," *J. Stat. Softw.*, vol. 33, 2010.

[15] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite-element approximations," *Comput. Math. Appl.*, vol. 2, pp. 17–40, 1976.

[16] R. Glowinski and A. Marrocco, "Sur l'approximation, paréléments finis d'orde un, et la résolution par pénalisation-dualité d'une classe de problèmes de Dirichlet non-linéaires," (in French) *Rev. Francaise d'Aut. Inf. Rech. Oper.*, vol. 2, pp. 41–76, 1975.
[17] T. Goldstein and S. Osher, "The split bregman method for L1 regularized problems," *SIAM J. Imag. Sci.*, vol. 2, pp. 323–343, 2009.
[18] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. New York: Springer, 2009.
[19] X. Hong, C. Wang, H. Chen, and Y. Zhang, "Secondary spectrum access networks," *IEEE Veh. Technol. Mag.*, vol. 4, no. 2, pp. 36–43, 2009.
[20] A. Jadbabaie and S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Autom, Control*, vol. 48, no. 6, pp. 988–1001, 2003.
[21] A. Nedic and D. P. Bertsekas, "Incremental subgradient methods for nondifferentiable optimization," *SIAM J. Optim.*, vol. 12, pp. 109–138, Jan. 2001.
[22] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.
[23] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, pp. 215–233, Jan. 2007.
[24] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "A collaborative training algorithm for distributed learning," *IEEE Trans. Inf. Theory*, vol. 55, no. 4, pp. 1856–1871, Apr. 2009.
[25] M. G. Rabbat and R. D. Nowak, "Quantized incremental algorithms for distributed optimization," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 798–808, 2005.
[26] M. G. Rabbat, R. D. Nowak, and J. A. Bucklew, "Generalized consensus computation in networked systems with erasure links," in *Proc. Workshop Signal Processing Advances in Wireless Communications*, New York, Jun. 2005, pp. 1088–1092.
[27] S. S. Ram, A. Nedic, and V. V. Veeravalli, "Stochastic incremental gradient descent for estimation in sensor networks," in *Proc. 41st Asilomar Conf. Signals, Systems, Computers*, Pacific Grove, CA, 2007, pp. 582–586.
[28] S. S. Ram, A. Nedic, and V. V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," Nov. 2008 [Online]. Available: http://arxiv.org/abs/0811.2595
[29] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc WSNs with noisy links—Part I: Distributed estimation of deterministic signals," *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 350–364, Jan. 2008.
[30] M. Schmidt, G. Fung, and R. Rosales, "Fast optimization methods for L1 regularization: A comparative study and two new approaches," in *Proc. Euro. Conf. Machine Learning*, Warsaw, Poland, 2007, pp. 286–297.
[31] T. Stamey, J. Kabalin, I. Johnstone, F. Freiha, E. Redwine, and N. Yang, "Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of prostate ii radical prostatectomy treated patients," *J. Urology*, vol. 16, pp. 1076–1083, 1989.
[32] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Stat. Soc. B*, vol. 58, pp. 267–288, 1996.
[33] R. Tibshirani, M. Saunders, R. Rosset, J. Zhu, and K. Knight, "Sparsity and smoothness via the fused lasso," *J. Roy. Stat. Soc. B*, vol. 67, pp. 91–108, 2005.
[34] P. Tseng, "Convergence of block coordinate descent method for nondifferentiable maximization," *J. Optim. Theory Appl.*, vol. 109, pp. 473–492, 2001.
[35] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo, "Sparse reconstruction by separable approximation," *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2479–2493, Jul. 2009.
[36] T. Wu and K. Lange, "Coordinate descent algorithms for lasso penalized regression," *Ann. Appl. Stat.*, vol. 2, pp. 224–244, 2008.
[37] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Control Lett.*, vol. 53, pp. 65–78, Sep. 2004.
[38] H. Zhu, G. B. Giannakis, and A. Cano, "Distributed in-network channel decoding," *IEEE Trans. Signal Process.*, vol. 57, no. 10, pp. 3970–3983, Oct. 2009.
[39] H. Zou, "The adaptive lasso and its oracle properties," *J. Amer. Stat. Assoc.*, vol. 101, pp. 1418–1429, 2006.
[40] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. Roy. Stat. Soc. B*, vol. 67, pp. 301–320, 2005.

Gonzalo Mateos (S'07) received the B.Sc. degree in electrical engineering from Universidad de la República Oriental del Uruguay, Montevideo, Uruguay, in 2005 and the M.Sc. degree in electrical and computer engineering from the University of Minnesota, Minneapolis, in 2009. Since August 2006, he has been working towards the Ph.D. degree as a Research Assistant with the Department of Electrical and Computer Engineering, University of Minnesota.

Since 2003, he has been a Research and Teaching Assistant with the Department of Electrical Engineering, Universidad de la República Oriental del Uruguay. From 2004 to 2006, he worked as a Systems Engineer at Asea Brown Boveri (ABB), Uruguay. His research interests lie in the areas of communication theory, signal processing and networking. His current research focuses on distributed signal processing, sparse linear regression and social networks.

Juan Andrés Bazerque (S'06) received the B.Sc. degree in electrical engineering from Universidad de la República (UdelaR), Montevideo, Uruguay, in 2003 and the M.Sc. degree in electrical engineering from the University of Minnesota (UofM), Minneapolis, in August 2009. He is currently working towards the Ph.D. degree at UofM.

From 2003 to 2006, he worked as a Telecommunications Engineer at Uniotel S.A., Uruguay, developing applications for Voice over IP. From 2000 to 2006, he was a Teaching Assistant with the Department of Mathematics and Statistics, and with the Department of Electrical Engineering (UdelaR). Since August 2006, he has been a Research Assistant at UofM. His broad research interests lie in the general areas of networking, communications, and signal processing. His current research focuses on decentralized algorithms for in-network processing, cooperative wireless communications, cognitive radios, compressive sampling, and sparsity-aware statistical models.

Mr. Bazerque is the recipient of the UofM's Distinguished Master's Thesis Award 2009–2010 and corecipient of the Best Student Paper Award at the Second International Conference on Cognitive Radio Oriented Wireless Networks and Communication 2007.

Georgios B. Giannakis (F'97) received the Diploma degree in electrical engineering from the National Technical University of Athens, Greece, in 1981 and the M.Sc. degree in electrical engineering, the M.Sc. degree in mathematics, and the Ph.D. degree in electrical engineering from the University of Southern California (USC) in 1983, 1986, and 1986, respectively.

Since 1999, he has been a Professor with the University of Minnesota, where he now holds an ADC Chair in Wireless Telecommunications in the Electric and Computer Engineering Department and serves as Director of the Digital Technology Center. His general interests span the areas of communications, networking and statistical signal processing subjects on which he has published more than 300 journal papers, 500 conference papers, two edited books, and two research monographs. Current research focuses on compressive sensing, cognitive radios, network coding, cross-layer designs, mobile ad hoc networks, wireless sensor, and social networks.

Dr. Giannakis is the (co-)inventor of 20 patents issued, and the (co)recipient of seven paper awards from the IEEE Signal Processing (SP) and Communications Societies, including the G. Marconi Prize Paper Award in Wireless Communications. He also received Technical Achievement Awards from the SP Society (2000), from EURASIP (2005), a Young Faculty Teaching Award, and the G. W. Taylor Award for Distinguished Research from the University of Minnesota. He is a Fellow of EURASIP, and has served the IEEE in a number of posts, including that of Distinguished Lecturer for the IEEE SP Society.