

# Preserving Privacy in Distributed LASSO

Wen Zhang, Yufan Fan, and Marius Pesavento

Communication Systems Group, Technische Universität Darmstadt, Germany

**Abstract**—In this paper, we extend the Soft-Thresholding with Exact Line search Algorithm (STELA) to solve the LASSO problem in a fully decentralized manner, where each agent solves its local minimization problem, and cooperates only with its neighbors to update the local solution. Moreover, the privacy of the local data is maintained during the communication of agents via the privacy-preserving average consensus (PPAC) approach which avoids revealing local information from other agents as well as potential eavesdroppers. We examine the proposed algorithm with synthetic data. Simulation results show that with a similar privacy level, the proposed algorithm has a faster convergence speed and better accuracy compared to the state-of-the-art privacy-preserving Primal-Dual Method of Multipliers (p-PDMM) algorithm.

**Index Terms**—Decentralized optimization, LASSO, average consensus, privacy preserving, STELA

## I. MOTIVATION AND INTRODUCTION

In signal processing, the Least Absolute Shrinkage and Selection Operator (LASSO) problem plays an important role due to its wide applications, such as compressed sensing [1]–[3], audio processing [4]–[6], and recovery of a sparse signal [7]–[9]. The LASSO problem is well studied, and many algorithms have been proposed in the literature [10], [11], where the Soft-Thresholding with Exact Line search Algorithm (STELA) proposed in [12] outperforms others with a faster convergence speed [13].

Nevertheless, as the data size increases massively, it is difficult or even impractical to collect the data in a central processor, and perform the aforementioned algorithms. Hence, distributed algorithms that consist of a central coordinator and multiple working agents are proposed, where each working agent processes only its local data and the central coordinator collects the processed variables to produce the final solution. However, such a network is not reliable, where the failure of the central coordinator can lead to the breakdown of the whole network. Thus, the fully decentralized setup has attracted the attention in various scenarios, e.g., in wireless communication [14], machine learning [15], and Internet-of-Things (IoT) [16], where each agent evaluates its local data, and exchanges information only with adjacent neighbors. In this case, the central coordinator is avoided, and the network is more robust.

There exist different algorithms to help the agents collaborate with neighbors with the goal to compute the weighted average of the data available at each agent, e.g., the average consensus [17], the Push-Sum [18], the graph filter [19], [20]. Nonetheless, as in distributed algorithms the agents

transmit local information available at the respective location, their private information may leak into the network or to other adversary agents. In many applications, the protection of privacy is a fundamental requirement. For example, in image compression and face recognition, the LASSO problem can be used to find the sparse representation vector of a measurement vector with a given dictionary. However, the dictionary is often distributed or aggregated from different local datasets containing sensitive data and agents often want to prevent the leakage of private information over the network. Various privacy-preserving algorithms have been proposed in the literature. In [21], a private distributed matrix multiplication approach is proposed, where the matrix is encoded with polynomial codes before it is sent to different agents. In [22], a privacy-preserving Push-Sum algorithm is proposed to compute the average privately in a directed graph, where each agent only sends a part of its states rather than the complete state information.

In this paper, we propose a decentralized privacy-preserving implementation of the STELA algorithm, denoted by dp-STELA, to solve the LASSO problem, which is extended from the STELA algorithm proposed in [12] but without the central coordinator. The proposed algorithm solves the local optimization problem in separated agents parallelly, and adopts the Privacy-Preserving Average Consensus (PPAC) algorithm [23] to collaborate with neighbors to find the global optimum.

*Notation:* In this paper, we use the lowercase letter  $x$ , the boldface lowercase letter  $\mathbf{x}$ , the boldface uppercase letter  $\mathbf{X}$  and the calligraphic letters  $\mathcal{X}$  to denote a scalar, a vector, a matrix, and a set, respectively. Matrix  $\mathbf{X}^T$  denotes the transpose of matrix  $\mathbf{X}$ . The argument  $t$  denotes the outer iteration index, and the argument  $k$  denotes the inner iteration index. The vector  $\mathbf{d}(\mathbf{X})$  denotes the diagonal of matrix  $\mathbf{X}$ .  $[x]_a^b$  denotes  $\max(\min(x, a), b)$ . The operator  $\circ$  presents the Hadamard, i.e., the element-wise, product.

## II. REVIEW OF STELA

The LASSO problem to recover the sparse vector  $\mathbf{x}$  can be expressed as:

$$\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \mu \|\mathbf{x}\|_1, \quad (1)$$

where  $\mathbf{A} \in \mathbb{R}^{N \times K}$ ,  $\mathbf{x} \in \mathbb{R}^{K \times 1}$ ,  $\mathbf{b} \in \mathbb{R}^{N \times 1}$  and  $\mu > 0$ . Among different algorithms that solve the LASSO problem (1), the STELA algorithm proposed in [12] is an iterative algorithm that stands out by its simple implementation and fast convergence speed. It consists of two parts, i.e., the descent direction finding and the exact line search computation.

This work was financially supported by the Federal Ministry of Education and Research of Germany in the project “Open6GHub” (grant no. 16KISK014).

In the  $t$  iteration, we denote an approximate function of  $f(\mathbf{x})$  around the current point  $\mathbf{x}^t$  as  $\tilde{f}(\mathbf{x}, \mathbf{x}^t)$ . The optimal solution  $\mathbb{B}\mathbf{x}^t$ , i.e., the best response point of  $\mathbf{x}^t$ , which ensures  $\mathbb{B}\mathbf{x}^t - \mathbf{x}^t$  to be the descent direction of  $f(\mathbf{x})$ , can be found by

$$\begin{aligned}\mathbb{B}\mathbf{x}^t &= \arg \min_{\mathbf{x} \in \mathcal{X}} \left\{ \tilde{f}(\mathbf{x}, \mathbf{x}^t) + g(\mathbf{x}) \right\} \\ &= \mathbf{d}(\mathbf{A}^\top \mathbf{A})^{-1} \circ \mathcal{S}_\mu(\mathbf{r}(\mathbf{x}^t)),\end{aligned}\quad (2)$$

where the inverse  $\mathbf{d}(\mathbf{A}^\top \mathbf{A})^{-1}$  is performed element-wise,  $\mathcal{S}_\mu(\mathbf{r}(\mathbf{x}))$  is the soft-thresholding operator that is defined by  $\mathcal{S}_\mu(\mathbf{r}) = [\mathbf{r} - \mu \mathbf{1}]^+ - [-\mathbf{r} - \mu \mathbf{1}]^+$  [24], and  $\mathbf{r}(\mathbf{x}^t)$  is given by

$$\mathbf{r}(\mathbf{x}^t) \triangleq \mathbf{d}(\mathbf{A}^\top \mathbf{A}) \circ \mathbf{x}^t - \mathbf{A}^\top (\mathbf{A}\mathbf{x}^t - \mathbf{b}). \quad (3)$$

After finding the descent direction, the optimal step size  $\gamma^t$  taken along the descent direction of  $\mathbb{B}\mathbf{x}^t - \mathbf{x}^t$  can be computed by the exact line search method [12] as

$$\gamma^t = \left[ -\frac{(\mathbf{A}\mathbf{x}^t - \mathbf{b})^\top \mathbf{A}(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t) + \mu(\|\mathbb{B}\mathbf{x}^t\|_1 - \|\mathbf{x}^t\|_1)}{(\mathbf{A}(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t))^\top (\mathbf{A}(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t))} \right]_{(4)}^1.$$

This is a closed-form solution, which accelerates the convergence speed of the algorithm. At the end of each iteration, the variable  $\mathbf{x}^{t+1}$  for the next iteration is updated as

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \gamma^t(\mathbb{B}\mathbf{x}^{t+1} - \mathbf{x}^t). \quad (5)$$

### III. ALGORITHM DEVELOPMENT

When the dimension of  $\mathbf{A}$  is large and the hardware has insufficient resources for processing the data. It is impossible to solve the LASSO problem in a centralized approach. The authors in [12] also proposed a distributed implementation of the STELA algorithm when the size of  $\mathbf{A}$  is too large to be directly operated in a single processor. Nevertheless, a central coordinator is still required to distribute and gather the intermediate information to all agents in the network. Consequently, the distributed implementation of the STELA algorithm highly depends on the central coordinator and thus becomes unreliable when an attack or an error occurs at the central coordinator.

In this paper, we devise a decentralized implementation of the STELA algorithm to solve the LASSO problem (1) in a fully decentralized manner, where no centralized coordinator is required, and each agent evaluates only its local data and exchanges the local solution with adjacent neighbors privately and securely to find the globally optimal solution.

#### A. Decentralized privacy-preserving STELA

To employ the STELA algorithm fully decentralized, the matrix  $\mathbf{A}$  and the vector  $\mathbf{x}$  are partitioned into  $n$  subsets and offloaded to  $n$  agents to perform the computation locally. For example, in the application of image compression,  $\mathbf{b}$  is the global observation vector, the dictionary matrix  $\mathbf{A} = [\mathbf{A}_1, \dots, \mathbf{A}_n]$ , where submatrix  $\mathbf{A}_i \in \mathbb{R}^{N \times k_i}$  denotes the private dictionary of the agent  $i$ , with  $\sum_{i=1}^n k_i = K$ , and  $\mathbf{x} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top$ , where  $\mathbf{x}_i \in \mathbb{R}^{k_i \times 1}$ . The integer  $k_i$  denotes the dictionary size of agent  $i$ , i.e., the number of columns of  $\mathbf{A}_i$ . Such a network can be described by an undirected graph

consisting of  $n$  agents, which is denoted as  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ , where  $\mathcal{V} = \{1, \dots, n\}$  denotes the set of  $n$  agents, and  $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$  denotes the set of  $m$  edges. The agent  $i$  can communicate with agent  $j$  if  $(i, j) \in \mathcal{E}$ , and the set of the neighbors of the agent  $i$  is denoted as  $\mathcal{N}_i$ .

Since, the submatrix  $\mathbf{A}_i$  and the subvector  $\mathbf{x}_i$  are assumed to be available to agent  $i$  locally, the  $i$ -th entry of the best response point  $\mathbb{B}\mathbf{x}$ , i.e.,  $\mathbb{B}\mathbf{x}_i$ , is also decomposed into  $n$  independent subsets as  $\mathbb{B}\mathbf{x} = [\mathbb{B}\mathbf{x}_1, \dots, \mathbb{B}\mathbf{x}_n]^\top$ . In each iteration  $t$ , agent  $i$  computes  $\mathbb{B}\mathbf{x}_i^t$  locally as

$$\mathbb{B}\mathbf{x}_i^t = \mathbf{d}(\mathbf{A}_i^\top \mathbf{A}_i)^{-1} \circ \mathcal{S}_\mu(\mathbf{r}(\mathbf{x}_i^t)), \quad (6)$$

where  $\mathbf{r}(\mathbf{x}_i^t)$  is computed at agent  $i$  as

$$\mathbf{r}(\mathbf{x}_i^t) \triangleq \mathbf{d}(\mathbf{A}_i^\top \mathbf{A}_i) \circ \mathbf{x}_i^t - \mathbf{A}_i^\top (\mathbf{A}\mathbf{x}^t - \mathbf{b}). \quad (7)$$

Moreover, from (4), we notice that each agent requires the knowledge of  $\mathbf{A}(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)$ ,  $\mathbb{B}\mathbf{x}^t$  and  $\mathbf{A}\mathbf{x}^t - \mathbf{b}$  to compute the step size  $\gamma^t$  locally. Based on the data partition, the first vector  $\mathbf{A}(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)$  can be calculated by

$$\mathbf{A}(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t) = \sum_{i=1}^n \mathbf{A}_i(\mathbb{B}\mathbf{x}_i^t - \mathbf{x}_i^t). \quad (8)$$

In a distributed network, (8) can be realized via different approaches, e.g., the average consensus algorithm [17], the Push-Sum algorithm [18], and the finite-time average consensus [25]. However, when performing these distributed algorithms sensitive information leakage may occur when there are adversaries in the network. We consider two kinds of popular adversary models, i.e., the eavesdropping adversary and the passive adversary [26]. More precisely, the former indicates eavesdropping the communication channel, and the latter indicates an adversary agent attempting to infer the private data of others based on the received information. In our problem, we want to preserve the privacy of  $\mathbf{A}_i$  at each agent  $i$ ,  $i = 1, \dots, n$ , from other agents in the network, as well as the privacy of  $\mathbf{x}_i$  from potential eavesdroppers over the communication channel.

Furthermore, even though in (8), each agent transmits an intermediate variable  $\mathbf{A}_i(\mathbb{B}\mathbf{x}_i^t - \mathbf{x}_i^t)$  instead of the private data directly, it still does not guarantee the data privacy at a high level. For example, if adversary agents construct a set of linear equations based on the received data, they may be able to decode  $\mathbf{A}_i$ . Therefore, we adopt the PPAC algorithm [23] to compute the summation in (8) distributively over the network with a high level of privacy and security in our proposed algorithm as outlined in Section III-B.

The second term  $\mathbb{B}\mathbf{x}^t$  can be found by gathering  $\mathbb{B}\mathbf{x}_i^t$  over the network and stacking the component vectors as

$$\mathbb{B}\mathbf{x}^t = [(\mathbb{B}\mathbf{x}_1^t)^\top, \dots, (\mathbb{B}\mathbf{x}_n^t)^\top]^\top \quad (9)$$

which can be achieved by, e.g., broadcasting. However, sharing  $\mathbb{B}\mathbf{x}_i^t$  directly is not secure if potential eavesdroppers attempt to spy on the communication channel. Nevertheless, we can reformulate (9) to have a similar formulation as (8), i.e., a summation of the information in each agent, such that the

PPAC algorithm can be applied, and  $\mathbb{B}\mathbf{x}_i^t$  is collected securely. To do so we denote auxiliary vectors  $\mathbb{B}\tilde{\mathbf{x}}_i^t \in \mathbb{R}^{K \times 1}$  as

$$\begin{aligned} \mathbb{B}\tilde{\mathbf{x}}_1^t &= [(\mathbb{B}\mathbf{x}_1^t)^\top, \mathbf{0}^\top, \dots, \mathbf{0}^\top]^\top, \\ &\vdots \\ \mathbb{B}\tilde{\mathbf{x}}_n^t &= [\mathbf{0}^\top, \dots, \mathbf{0}^\top, (\mathbb{B}\mathbf{x}_1^t)^\top]^\top. \end{aligned} \quad (10)$$

Thus, (9) is reformulated as

$$\mathbb{B}\mathbf{x}^t = \sum_{i=1}^n \mathbb{B}\tilde{\mathbf{x}}_i^t, \quad (11)$$

which can be implemented using the PPAC algorithm as outlined in Section III-B.

Finally, the last term, i.e.,  $\mathbf{A}\mathbf{x}^t - \mathbf{b}$ , can be easily computed by each agent locally after computing the step size  $\gamma^{t-1}$ , i.e.,

$$\begin{aligned} \mathbf{A}\mathbf{x}^t - \mathbf{b} &= \mathbf{A}(\mathbf{x}^{t-1} + \gamma^{t-1}(\mathbb{B}\mathbf{x}^{t-1} - \mathbf{x}^{t-1})) - \mathbf{b} \\ &= (\mathbf{A}\mathbf{x}^{t-1} - \mathbf{b}) + \gamma^{t-1}\mathbf{A}(\mathbb{B}\mathbf{x}^{t-1} - \mathbf{x}^{t-1}). \end{aligned} \quad (12)$$

### B. Privacy-Preserving Average Consensus (PPAC)

The algorithm described in Section III-A requires the distributed computation of the summations in (8) and (11). This can be privately computed using the PPAC algorithm, which is described in the following. Let  $\mathbf{s}_i \in \mathbb{R}^{u \times 1}$  denote private data in agent  $i$ , we want to privately compute the average  $\mathbf{y}^* = \frac{1}{n} \sum_{i=1}^n \mathbf{s}_i$  at each agent. The private data  $\mathbf{s}_i$  represents  $\mathbf{A}_i(\mathbb{B}\mathbf{x}_i^t - \mathbf{x}_i^t)$  and  $\mathbb{B}\tilde{\mathbf{x}}_i^t$  in (8) and (11), respectively, and  $\mathbf{y}^*$  represents the associated averages. In the PPAC algorithm, the local averages in each agent are computed as the solution of the following optimization problem.

$$\begin{aligned} \min_{\mathbf{y}_i} \quad & \sum_{i=1}^n \frac{1}{2} \|\mathbf{y}_i - \mathbf{s}_i\|_2^2 \\ \text{s.t.} \quad & \mathbf{y}_i = \mathbf{y}_j, \forall (i, j) \in \mathcal{E}. \end{aligned} \quad (13)$$

This optimization problem can be solved with the Primal-Dual Method of Multipliers (PDMM) [27], where the local update scheme is given by (details cf. [23] and the references therein):

$$\mathbf{y}_i^{k+1} = \frac{\mathbf{s}_i + \sum_{j \in \mathcal{N}_i} (c\mathbf{y}_j^k - \mathbf{B}_{i|j}\lambda_{j|i}^k)}{1 + cd_i}, \quad (14a)$$

$$\forall j \in \mathcal{N}_i: \lambda_{i|j}^{k+1} = \lambda_{j|i}^k + c\mathbf{B}_{i|j}(\mathbf{y}_i^{k+1} - \mathbf{y}_j^k), \quad (14b)$$

where  $c$  is a constant to control the convergence rate,  $d_i$  is the degree of agent  $i$ ,  $\mathbf{B}_{i|j} \in \mathbb{R}^{u \times u}$  is an identity matrix for  $(i, j) \in \mathcal{E}$  with  $\mathbf{B}_{i|j} = \mathbf{I}$  if  $i > j$  and  $\mathbf{B}_{i|j} = -\mathbf{I}$  if  $i < j$ , and  $\lambda_{i|j} \in \mathbb{R}^{u \times 1}$  denotes the dual variable at agent  $i$  sent to agent  $j$ . At convergence the dual variables  $\lambda_{i|j}$  and  $\lambda_{j|i}$  coincide. We collect all dual variables in  $\lambda \in \mathbb{R}^{2mu \times 1}$  with

$$\lambda = [\underbrace{\lambda_{1|2}^\top, \dots, \lambda_{m-1|m}^\top}_{i < j}, \underbrace{\lambda_{2|1}^\top, \dots, \lambda_{m|m-1}^\top}_{i > j}]^\top. \quad (15)$$

In the PPAC algorithm, a subspace perturbation technique is introduced to initialize vector  $\lambda^0$  with a large perturbation to ensure the privacy. In particular, the dual variable  $\lambda^k$  is shown

### Algorithm 1 Decentralized privacy-preserving STELA in Agent $i$

---

```

1: Initialization:  $\mathbf{x}^0 = \mathbf{0}$ .
2: for  $t = 1, \dots, T$  do
3:   In-Agent Computation
4:   Compute  $\mathbb{B}\mathbf{x}_i$  using (6).
5:   Compute  $\mathbf{A}_i(\mathbb{B}\mathbf{x}_i - \mathbf{x}_i)$ .
6:   Network Communication
7:   Initialize  $\mathbf{y}_{1,i}^0 = \mathbb{B}\tilde{\mathbf{x}}_i$ ,  $\mathbf{y}_{2,i}^0 = \mathbf{A}_i(\mathbb{B}\mathbf{x}_i - \mathbf{x}_i)$ , and  $\lambda_{1,i}^0$ ,  $\lambda_{2,i}^0$  with  $\sigma^2$ .
8:   for all  $\ell \in \{1, 2\}$  do in parallel
9:     for  $k = 1, \dots, K$  do
10:      Update  $\mathbf{y}_{\ell,i}^{k+1}$  using (14a).
11:      Update  $\lambda_{\ell,i}^{k+1}$  using (14b).
12:     end for
13:   end for
14:   In-Agent Computation
15:    $\mathbb{B}\mathbf{x}^t \leftarrow n\mathbf{y}_{1,i}^{K+1}$ .
16:    $\mathbf{A}(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t) \leftarrow n\mathbf{y}_{2,i}^{K+1}$ .
17:   Compute  $\gamma^t$  using (4)
18:   Update  $\mathbf{x}^{t+1}$  using (5)
19:   Update  $\mathbf{A}\mathbf{x}^{t+1} - \mathbf{b}$  using (12)
20:    $t \leftarrow t + 1$ 
21: end for

```

---

in [27] to converge in a subspace, denoted as  $H$ , where the orthogonal projection onto  $H$  is denoted as  $\Pi_H$ , and  $\lambda^k$  consists of two parts, i.e., the convergent component  $(\Pi_H)\lambda^k \rightarrow \lambda^*$ , and the non-convergent component  $\mathbf{P}^k(\mathbf{I} - \Pi_H)\lambda^0$ , where  $\mathbf{P}$  is a symmetric permutation matrix exchanging the first half rows with last half rows of the matrix it applies, and  $\mathbf{P}^2 = \mathbf{I}$ .

Furthermore, the observation in adversaries is summarized in [27] as  $\mathbf{y}_i^{k+1} - \mathbf{s}_i + \sum_{j \in \mathcal{N}_{i,h}} \mathbf{B}_{i|j}\lambda_{j|i}^k$ , where  $\mathcal{N}_{i,h}$  denotes the set of honest neighbors of agent  $i$ , and the last term can be seen as an additional noise that prevent adversaries from inferring the private data  $\mathbf{s}_i$  as the optimization variable  $\mathbf{y}_i^{k+1}$  will be shared with neighbors directly and be collected by adversaries easily. The noise term can be reformulated as a convergent term, i.e.,  $\sum_{j \in \mathcal{N}_{i,h}} \mathbf{B}_{i|j}(\Pi_H\lambda^k)_{j|i}$ , plus a non-convergent term, i.e.,  $\sum_{j \in \mathcal{N}_{i,h}} \mathbf{B}_{i|j}(\mathbf{P}^k(\mathbf{I} - \Pi_H)\lambda^0)_{j|i}$ . To protect the private data, we want the information leakage, e.g., the mutual information between the private data and the observation at adversaries  $I((\mathbf{y}_i^{k+1} - \mathbf{s}_i); (\mathbf{y}_i^{k+1} - \mathbf{s}_i + \sum_{j \in \mathcal{N}_{i,h}} \mathbf{B}_{i|j}\lambda_{j|i}^k))$ , to be sufficiently small, which can be achieved by increasing the variance of the noise term [27]. Noticing that only the variance of the non-convergent term of the noise can be manipulated, which only depends on the initialization of the dual variable, the mutual information can be reduced by initializing  $\lambda^0$  with sufficiently large variance  $\sigma^2$ , and thus, the privacy of the PPAC algorithm in (14) is maintained.

Consequently, the privacy of our proposed algorithm is guaranteed by applying the PPAC algorithm to carry out (8) and (11), where  $\mathbf{A}(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)$  and  $\mathbb{B}\mathbf{x}^t$  are computed by multiplying

the number of agent  $n$  with the corresponding output of the PPAC algorithm. Moreover, since the perturbation is inserted only to the dual variables  $\lambda_{i|j}$ , it has no influence on the convergence of the optimization variable  $y_i$ , thus, the accuracy is also guaranteed [27].

We summarized the complete dp-STELA in Algorithm 1, where  $T$  denotes the maximum number of the outer loop, i.e., the STELA algorithm, iteration, and  $K$  denotes the maximum number of the inner loop, i.e., the PPAC algorithm, iteration. We remark that although the private data, i.e.,  $\mathbf{A}_i$  and  $\mathbf{x}_i$  in our proposed algorithm, can be partitioned with different sizes, i.e., different values of  $k_i$ , the dimension  $u$  of the private data is the same through all agents, i.e.,  $u = N$  for  $\mathbf{A}_i(\mathbb{B}\mathbf{x}_i^t - \mathbf{x}_i^t)$  in (8), and  $u = K$  for  $\mathbb{B}\mathbf{x}_i^t$  in (11).

#### IV. SIMULATION RESULTS

In this section, we show the simulation results of the proposed dp-STELA algorithm, where the distributed private PDMM (p-PDMM) algorithm described in [27] is used as a comparison. To have a fair comparison, we adopt the same setup as in [27], where a random graph is considered. Two agents are connected with each other if the distance between them is less than  $\sqrt{2 \log n / n}$  to enable the graph to be connected with a high probability [28]. The parameters of the LASSO problem are chosen as follows. The matrix  $\mathbf{A} \in \mathbb{R}^{N \times K}$  with  $N = 80, K = 200$  is generated randomly from a Gaussian distribution with zero mean and unit variance. The true sparse vector denotes as  $\mathbf{x}_{\text{true}}$  has the sparsity level of 0.4 with the elements drawn from a Gaussian distribution with zero mean and unit variance. The observation vector  $\mathbf{b} = \mathbf{A}\mathbf{x}_{\text{true}}$ . The dual variable  $\lambda^0$  in the PPAC algorithm, if not specified, is initialized with the variance  $\sigma^2 = 10^6$  to guarantee a high level of data privacy. Simulation results are averaged over 20 Monte Carlo iterations.

We consider one iteration as one communication instance of all agents in the network, where the number of iterations of the dp-STELA algorithm consists of two parts, i.e., the outer loop of the STELA algorithm and the inner loop of the PPAC algorithm. In our simulation, the maximum number of iterations of the STELA algorithm and the PPAC algorithm are set to be  $T = 50$ , and  $K = 30$ , respectively, where one step of the STELA algorithm consists of two instances of the PPAC algorithm to carry out (8) and (11). Thus, the total number of iterations for the dp-STELA algorithm is the product of the maximum number of iterations of the STELA algorithm and twice the maximum number of iterations of the PPAC algorithm, i.e.,  $2TK$ , which is also used as the total number of iterations in the p-PDMM algorithm. The relative error is defined as

$$\text{relative error} = \|\hat{\mathbf{X}}^t - \hat{\mathbf{X}}_{\text{opt}}\|_2^2 / \|\hat{\mathbf{X}}_{\text{opt}}\|_2^2, \quad (16)$$

where  $\hat{\mathbf{X}}^t = [\hat{\mathbf{x}}_1^t, \dots, \hat{\mathbf{x}}_n^t]$  denotes the matrix whose  $i$ -th column  $\hat{\mathbf{x}}_i^t$  is the solution at agent  $i$  in the  $t$ -th iteration, and each column of the matrix  $\hat{\mathbf{X}}_{\text{opt}}$  is the optimal solution computed by the centralized STELA algorithm with 60 iterations.

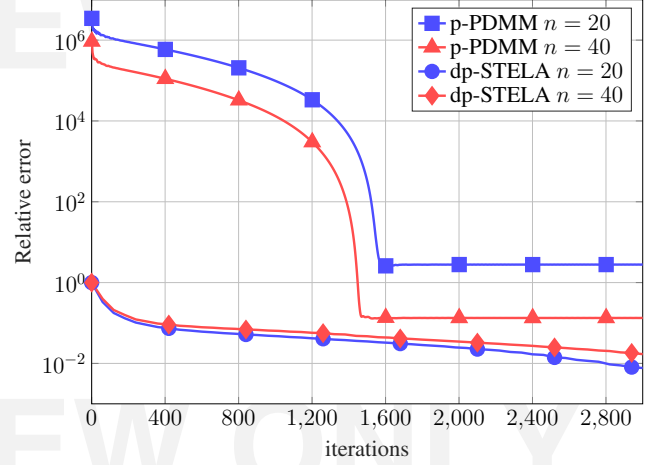


Fig. 1. Relative error performance of the dp-STELA algorithm and the p-PDMM algorithm solving the LASSO problem with  $N = 80, K = 200$  in networks with  $n = 20$  and  $n = 40$  agents.

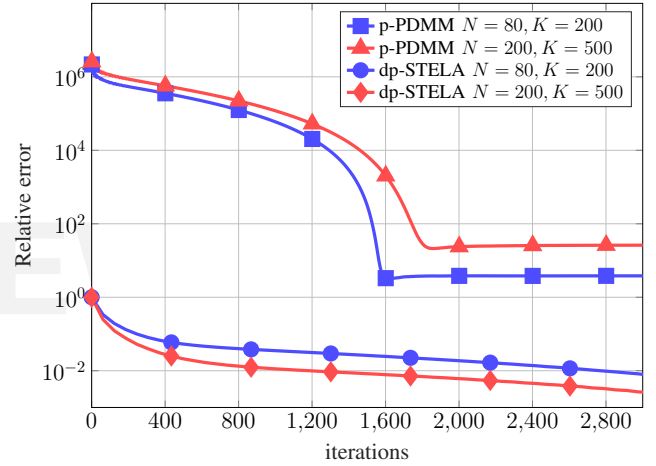


Fig. 2. Relative error performance of the dp-STELA algorithm and the p-PDMM algorithm solving the LASSO problem in the  $n = 20$  agents network with  $N = 80, K = 200$  and  $N = 200, K = 500$ .

In Figures 1 and 2, the relative error of the two algorithms is displayed versus the number of iterations. In Fig. 1, the blue curves represent the graph with  $n = 20$  agents, and the red curves describe the results with  $n = 40$  agents for a sensing matrix  $\mathbf{A}$  of dimension  $\mathbb{R}^{80 \times 200}$ . It shows that the dp-STELA algorithm has a faster convergence speed and can reach a lower relative error than the p-PDMM algorithm under the two different network sizes considered.

In Fig. 2 the relative error performance of the two algorithms is displayed for different dimensions of the sensing matrix  $\mathbf{A}$ , i.e.,  $\mathbf{A} \in \mathbb{R}^{80 \times 200}$  and  $\mathbf{A} \in \mathbb{R}^{200 \times 500}$  with a network size of  $n = 20$  agents. We observe that with the sensing matrix of a larger dimension, the p-PDMM algorithm attains a lower precision than in the case of the small sensing matrix, whereas the dp-STELA algorithm achieves a better precision with a similar convergence speed.



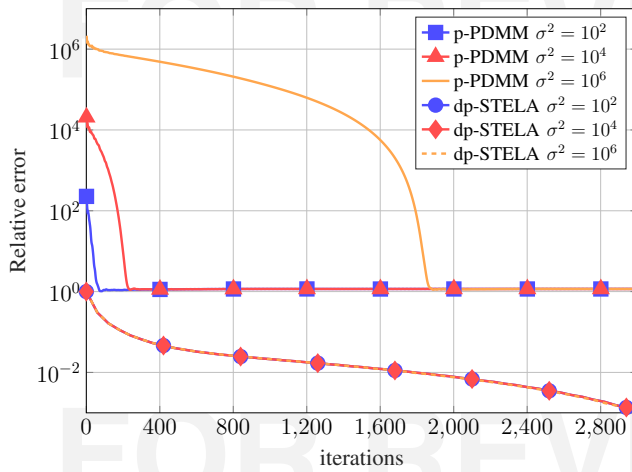


Fig. 3. Relative error performance of the dp-STELA algorithm and the p-PDMM algorithm solving the LASSO problem in the  $n = 20$  agents network with the different initialized variances  $\sigma^2$  of  $\lambda^0$

Moreover, the starting point of the p-PDMM algorithm is affected by the variance  $\sigma^2$  of the perturbation used in the initialization of  $\lambda^0$  which in turn defines the privacy level as can be observed in Fig. 3, where the relative error performance of the two algorithms is displayed for different initialization variance  $\sigma^2$  of  $\lambda^0$ . With a larger initialization variance of  $\lambda^0$ , the p-PDMM algorithm has a larger relative error at the initial stage and requires more iterations to eliminate the bias. On the contrary, the dp-STELA algorithm has a similar convergence speed regardless of the variance of the perturbation, as the perturbation is introduced only in the inner iteration during the PPAC algorithm and disappears when  $\lambda$  converges.

## V. CONCLUSION

In this paper, a fully decentralized STELA algorithm is proposed to solve the LASSO problem, where each agent can also keep its local data secure and private during the network communication with the help of the privacy-preserving average consensus approach. The proposed algorithm benefits from the fast convergence of STELA algorithm and exhibits lower relative error performance. In addition, there is no significant offset of the initialization point of our algorithm when inserting a large variance of perturbation to maintain the data with a high privacy level.

## REFERENCES

- [1] S. Pejoski and V. Kafedziski, "Asymptotic capacity lower bound for an OFDM system with LASSO compressed sensing channel estimation for Bernoulli-Gaussian channel," *IEEE Commun. Lett.*, vol. 19, no. 3, pp. 379–382, 2015.
- [2] M. Vehkaperä, Y. Kabashima, and S. Chatterjee, "Analysis of regularized ls reconstruction and random matrix ensembles in compressed sensing," *IEEE Trans. Inf. Theory*, vol. 62, no. 4, pp. 2100–2124, 2016.
- [3] D. Angelosante, G. B. Giannakis, and E. Grossi, "Compressed sensing of time-varying signals," in *2009 16th International Conference on Digital Signal Processing*. IEEE, 2009, pp. 1–8.
- [4] G. N. Lilis, D. Angelosante, and G. B. Giannakis, "Sound field reproduction using the LASSO," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, no. 8, pp. 1902–1912, 2010.
- [5] Y. Vaizman, B. McFee, and G. Lanckriet, "Codebook-based audio feature representation for music information retrieval," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 22, no. 10, pp. 1483–1493, 2014.
- [6] N. Radmanesh and I. S. Burnett, "Generation of isolated wideband sound fields using a combined two-stage LASSO-LS algorithm," *IEEE Trans. Audio, Speech, Language Process.*, vol. 21, no. 2, pp. 378–387, 2012.
- [7] X. Lv, G. Bi, and C. Wan, "The group LASSO for stable recovery of block-sparse signal representations," *IEEE Trans. Signal Process.*, vol. 59, no. 4, pp. 1371–1382, 2011.
- [8] H. Mansour, "Beyond  $\ell_1$ -norm minimization for sparse signal recovery," in *2012 IEEE Statistical Signal Processing Workshop (SSP)*. IEEE, 2012, pp. 337–340.
- [9] M. S. Asif and J. Romberg, "Sparse recovery of streaming signals using  $\ell_1$ -homotopy," *IEEE Trans. Signal Process.*, vol. 62, no. 16, pp. 4209–4223, 2014.
- [10] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.
- [11] T. T. Wu and K. Lange, "Coordinate descent algorithms for lasso penalized regression," *The Ann. App. Stat.*, vol. 2, no. 1, Mar. 2008.
- [12] Y. Yang and M. Pesavento, "A unified successive pseudoconvex approximation framework," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3313–3328, 2017.
- [13] J. Kuske and A. M. Tillmann, "Solving basis pursuit: Update, technical report," Tech. Rep., 2016.
- [14] J. Park, S. Samarakoon, A. Elgabli, J. Kim, M. Bennis, S.-L. Kim, and M. Debbah, "Communication-efficient and distributed learning over wireless networks: principles and applications," *Proc. IEEE*, vol. 109, no. 5, pp. 796–819, 2021.
- [15] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: a brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, 2017.
- [16] L. Liu, E. G. Larsson, W. Yu, P. Popovski, C. Stefanovic, and E. de Carvalho, "Sparse signal processing for grant-free massive connectivity: a future paradigm for random access protocols in the internet of things," *IEEE Signal Process. Mag.*, vol. 35, no. 5, pp. 88–99, 2018.
- [17] R. Saber and R. Murray, "Consensus protocols for networks of dynamic agents," in *Proceedings of the 2003 American Control Conference*, 2003, vol. 2, 2003, pp. 951–956.
- [18] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Push-sum distributed dual averaging for convex optimization," in *2012 IEEE 51st IEEE conference on decision and control (cdc)*. IEEE, 2012, pp. 5453–5458.
- [19] A. Loukas, A. Simonetto, and G. Leus, "Distributed autoregressive moving average graph filters," *IEEE Signal Process. Lett.*, vol. 22, no. 11, pp. 1931–1935, 2015.
- [20] Y. Fan, M. Trinh-Hoang, C. E. Ardic, and M. Pesavento, "Decentralized eigendecomposition for online learning over graphs with applications," *arXiv preprint arXiv:2209.01257*, 2022.
- [21] R. Bitar, M. Xhemrishi, and A. Wachter-Zeh, "Adaptive private distributed matrix multiplication," *IEEE Trans. Inf. Theory*, vol. 68, no. 4, pp. 2653–2673, 2022.
- [22] H. Wang, W. Xu, and J. Lu, "Privacy-preserving push-sum average consensus algorithm over directed graph via state decomposition," in *2021 3rd International Conference on Industrial Artificial Intelligence (IAI)*, 2021, pp. 1–6.
- [23] Q. Li, J. S. Gundersen, K. Tjell, R. Wisniewski, and M. G. Christensen, "Privacy-preserving distributed expectation maximization for gaussian mixture model using subspace perturbation," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 4263–4267.
- [24] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.
- [25] A. Sandryhaila, S. Kar, and J. M. Moura, "2014 IEEE international conference on acoustics, speech and signal processing (icassp)," *ICASSP 2014*, pp. 1080–1084, 2014.
- [26] Q. Do, B. Martini, and K.-K. R. Choo, "The role of the adversary model in applied security research," *Comput Secur.*, vol. 81, pp. 156–181, 2019.
- [27] Q. Li, R. Heusdens, and M. G. Christensen, "Privacy-preserving distributed optimization via subspace perturbation: a general framework," *IEEE Trans. Signal Process.*, vol. 68, pp. 5983–5996, 2020.
- [28] J. Dall and M. Christensen, "Random geometric graphs," *Phys. Rev. E*, vol. 66, no. 11, p. 016121, 2002.