An Asynchronous Distributed Proximal Gradient Method for Composite Convex Optimization

N. S. Aybat
Z. Wang
Penn State University, University Park, PA 16802 USA

G. Iyengar
Columbia University, New York, NY 10027 USA

Abstract
We propose a distributed first-order augmented Lagrangian (DFAL) algorithm to minimize the sum of composite convex functions, where each term in the sum is a private cost function belonging to a node, and only nodes connected by an edge can directly communicate with each other. This optimization model abstracts a number of applications in distributed sensing and machine learning. We show that any limit point of DFAL iterates is optimal; and for any \( \epsilon > 0 \), an \( \epsilon \)-optimal and \( \epsilon \)-feasible solution can be computed within \( O(\log(\epsilon^{-1})) \) DFAL iterations, which require \( O(\frac{\psi_{\max}}{\epsilon_{\min}} \epsilon^{-1}) \) proximal gradient computations and communications per node in total, where \( \psi_{\max} \) denotes the largest eigenvalue of the graph Laplacian, and \( d_{\min} \) is the minimum degree of the graph. We also propose an asynchronous version of DFAL by incorporating randomized block coordinate descent methods; and demonstrate the efficiency of DFAL on large scale sparse-group LASSO problems.

1. Introduction
Let \( \mathcal{G} = (\mathcal{N}, \mathcal{E}) \) denote a connected undirected graph of \( N \) computing nodes where nodes \( i \) and \( j \) can communicate information only if \((i, j) \in \mathcal{E}\). Each node \( i \in \mathcal{N} := \{1, \ldots, N\} \) has a private (local) cost function
\[
F_i(x) := \rho_i(x) + \gamma_i(x),
\]
where \( \rho_i : \mathbb{R}^n \to \mathbb{R} \) is a possibly non-smooth convex function, and \( \gamma_i : \mathbb{R}^n \to \mathbb{R} \) is a smooth convex function. We assume that the proximal map
\[
\text{prox}_{\rho_i}(x) := \underset{y \in \mathbb{R}^n}{\arg\min} \left\{ \rho_i(y) + \frac{1}{2} \|y-x\|_2^2 \right\}
\]
(2)
is efficiently computable for \( i \in \mathcal{N} \).

We propose a distributed augmented Lagrangian algorithm for efficiently computing a solution for the convex problem:
\[
F^* := \min_{x \in \mathbb{R}^n} F(x) := \sum_{i=1}^{N} F_i(x). \tag{3}
\]
Clearly, (3) can be solved in a "centralized" fashion by communicating all the private functions \( F_i \) to a central node, and solving the overall problem at this node. However, such an approach can be very expensive both from communication and computation perspectives. Suppose \( (A_i, b_i) \in \mathbb{R}^{m \times (n+1)} \) and \( F_i(x) = \|A_i x - b_i\|_2^2 + \lambda \|x\|_1 \) for \( i \in \mathcal{N} \) such that \( m \ll n \) and \( N \gg 1 \). Hence, (3) is a very large scale LASSO problem distributed data. To solve (3) in a centralized fashion the data \( \{(A_i, b_i) : i \in \mathcal{N}\} \) needs to be communicated to the central node. This can be prohibitively expensive, and may also violate privacy constraints. Furthermore, it requires that the central node have large enough memory to be able to accommodate all the data. On the other hand, at the expense of slower convergence, one can completely do away with a central node, and seek for consensus among all the nodes on an optimal decision using "local" decisions communicated by the neighboring nodes. In addition, for certain cases, computing partial gradients locally in an asynchronous manner can be even more computationally efficient when compared to computing the entire gradient at a central node. With these considerations in mind, we propose decentralized algorithms that can compute solutions to (3) using only local computations; thereby, circumventing all privacy, communication and memory issues. To facilitate the design of decentralized algorithms, we take advantage of the fact that graph \( \mathcal{G} \) is connected, and reformulate (3) as
\[
\min_{x_i \in \mathbb{R}^n, i \in \mathcal{N}} \left\{ \sum_{i=1}^{N} F_i(x_i) : x_i = x_j, \forall (i, j) \in \mathcal{E} \right\}. \tag{4}
\]
Optimization problems of form (4) model a variety of very important applications, e.g., distributed linear regression (Mateos et al., 2010), distributed control (Necoara & Suykens, 2008), machine learning (McDonald et al., 2010), and estimation using sensor networks (Lesser et al., 2003).
We call a solution \( \bar{x} = (\bar{x}_i)_{i \in N} \) \( \epsilon \)-feasible if the consensus violation \( \max_{(i,j) \in E} \|\bar{x}_i - \bar{x}_j\|_2 \leq \epsilon \) and \( \epsilon \)-optimal if \( \sum_{i \in N} F_i(\bar{x}_i) - F^* \leq \epsilon \). In this work, we propose a distributed first-order augmented Lagrangian (DFAL) algorithm, establish the following main result for the synchronous case in Section 2.2.3, and extend it to an asynchronous setting in Section 2.2.4.

**Main Result.** Let \( \{x^{(k)}\}_{k \in \mathbb{N}_+} \) denote the sequence of DFAL iterates. Then \( F^* = \lim_{k \to \infty} \sum_{i \in N} F_i(x^{(k)}_i) \). Furthermore, \( x^{(k)} \) is \( \epsilon \)-optimal and \( \epsilon \)-feasible within \( O(\log(\epsilon^{-1})) \) DFAL iterations, requiring \( O(\frac{1}{\delta_{\min} \epsilon^{-1}}) \) communications per node, and \( O(\epsilon^{-1}) \) gradient and proximal map computations for \( \gamma_i \) and \( \rho_i \), respectively, where \( \psi_{\max} \) denotes the largest eigenvalue of the Laplacian of \( \mathcal{G} \), and \( \delta_{\min} \) denotes the minimum degree over all nodes.

### 1.1. Previous work

Given the importance of (4), a number of different distributed optimization algorithms have been proposed to solve (4). Duchi et al. (2012) proposed a dual averaging algorithm to solve (3) in a distributed fashion over \( \mathcal{G} \) when each \( F_i \) is convex. This algorithm computes \( \epsilon \)-optimal solution in \( O(1/\epsilon^2) \) iterations; however, they do not provide any guarantees on the consensus violation \( \max_{(i,j) \in E} \|\bar{x}_i - \bar{x}_j\|_2 \). Nedic & Ozdaglar (2009) developed a subgradient method with constant step size \( \alpha > 0 \) for distributed minimization of (3) where the network topology is time-varying. Setting \( \alpha = O(\epsilon) \) in their method guarantees that consensus violation and suboptimality is \( O(\epsilon) \) in \( O(1/\epsilon^2) \) iterations; however, since the step size is constant none of the errors are not guaranteed to decrease further. Wei and Ozdaglar (2012, 2013), and recently Makhdoumi & Ozdaglar (2014) proposed an alternating direction method of multipliers (ADMM) algorithm that computes an \( \epsilon \)-optimal and \( \epsilon \)-feasible solution in \( O(1/\epsilon) \) proximal map evaluations for \( F_i \). There are several problems where one can compute the proximal map for \( \rho_i \) efficiently; however, computing the proximal map for \( F_1 = \rho_1 + \gamma_1 \) is hard - see Section 3 for an example. One can overcome this limitation of ADMM by locally splitting variables, i.e., setting \( F_1(x_1,y_1) := \rho_1(x_1) + \gamma_1(y_1) \), and adding a constraint \( x_i = y_i \) in (4). This approach doubles local memory requirement; in addition, in order for ADMM to be efficient, proximal maps for both \( \rho_i \) and \( \gamma_i \) must be efficiently computable. When each \( F_i \) is smooth and has bounded gradients, Jakovetic et al. (2011) developed a fast distributed gradient methods with \( O(1/\sqrt{\epsilon}) \) convergence rate. Note that for the quadratic loss, which is one of the most commonly used loss functions, the gradient is not bounded. Chen & Ozdaglar (2012) proposed an inexact proximal-gradient method for distributed minimization of (3) that is able to compute \( \epsilon \)-feasible and \( \epsilon \)-optimal solution in \( O(\epsilon^{-1/2}) \) iterations which require \( O(\epsilon^{-1}) \) communications per node over a time-varying network topology when \( F_1 = \rho_1 + \gamma_i \), assuming that the non-smooth term \( \rho \) is the same at all nodes, and \( \nabla \gamma_i \) is bounded for all \( i \in N \). In contrast, DFAL proposed in this paper is able to asynchronously compute an \( \epsilon \)-optimal \( \epsilon \)-feasible solution in \( O(\epsilon^{-1}) \) communications per node, allowing node specific non-smooth functions \( \rho_i \), and without assuming bounded \( \nabla \gamma_i \) for any \( i \in N \).

Aybat & Iyengar (2012) proposed an efficient first-order augmented Lagrangian (FAL) algorithm for the basis pursuit problem \( \min_{x \in \mathbb{R}^n} \{ ||x||_1 : Ax = b \} \) to compute an \( \epsilon \)-optimal and \( \epsilon \)-feasible solution to within \( O(\kappa^2(A)/\epsilon) \) matrix-vector multiplications, where \( A \in \mathbb{R}^{m \times n} \) such that \( \text{rank}(A) = m \) and \( \kappa(A) := \sigma_{\max}(A)/\sigma_{\min}(A) \) denotes the condition number of \( A \). In this work, we extend their FAL algorithm to solve a more general version of (4) in Section 2.2.1 and 2.2.2, and establish the **Main Result** for (4) in Section 2.2.3. In Section 2.2.4, we propose an asynchronous version of DFAL. It is important to emphasize that DFAL can be easily extended to solve (4) when there are global constraints on network resources of the form \( Ex - q \in K \), where \( K \) is a proper cone, and none of the algorithms discussed above can accommodate such global conic constraints efficiently. Due to space limitations, we do not discuss this extension here; however, the analysis would be similar to (Aybat & Iyengar, 2013; 2014).

### 2. Methodology

**Definition 1.** (a) Let \( \Gamma \) be the set of convex functions \( \gamma : \mathbb{R}^n \to \mathbb{R} \) such that \( \nabla \gamma \) is Lipschitz continuous with constant \( L_\gamma \), and \( \gamma(x) \geq \underline{\gamma} \) for all \( x \in \mathbb{R}^n \) for some \( \underline{\gamma} \in \mathbb{R} \).
Assumption 1. For all $i \in N$, we assume that $\gamma_i \in \Gamma$ and $\rho_i \in R$ with corresponding constants $L_{\gamma_i}, \gamma_i, B_i$ and $\tau_i$.

Most of the important regularizers and loss functions used in machine learning and statistics literature lie in $\Gamma$ and $\Lambda$, respectively. In particular, any norm, e.g., $\| \cdot \|_\alpha$ with $\alpha \in \{1, 2, \infty\}$, group norm (see Section 3), nuclear norm, etc., weighted sum of these norms, e.g., sparse group norm (see Section 3), all belong to $\Gamma$. Given $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, quadratic-loss $\|Ax - b\|^2_2$, Huber-loss $\sum_{i=1}^m \min\{a_i^2 x - b_i, 2\}$ (see Section 3), logistic-loss $\sum_{i=1}^m \log(1 + e^{-b_i a_i^T x})$, or fair-loss (Blatt et al., 2007) functions all belong to $\Gamma$.

Throughout the paper, we adopt the notation $x = (x_i; \xi_{-i})$ with $x_i = (x_{ij})_{j \neq i}$ to denote a vector where $x_i$ and $\xi_{-i}$ are treated as variable and parameter sub-vectors of $x$, respectively. Given $F : \mathbb{R}^{nN} \rightarrow \mathbb{R}$, $\nabla f(x) \in \mathbb{R}^n$ denotes the sub-vector of $\nabla f(x) \in \mathbb{R}^{nN}$ corresponding to components of $x_i \in \mathbb{R}^n$.

2.1. APG Algorithm for the Centralized Model

Consider the centralized version (3) where all the functions $F_i$ are available at a central node, and all computations are carried out at this node. Suppose $\{\rho_i\}_{i \in N}$ and $\{\gamma_i\}_{i \in N}$ satisfy Assumption 1. Let $\rho(x) := \sum_{i=1}^N \rho_i(x)$ and $\gamma(x) := \sum_{i=1}^N \gamma_i(x)$. Lipschitz continuity of each $\nabla \gamma_i$ with constant $L_{\gamma_i}$ implies that $\nabla \gamma$ is also Lipschitz continuous with constant $L_\gamma = \sum_{i=1}^N L_{\gamma_i}$. When $\text{prox}_{\rho/L}$ can be computed efficiently, the accelerated proximal gradient (APG) algorithm proposed in (Beck & Teboulle, 2009; Tseng, 2008) guarantees that

$$0 \leq F(x(t)) - F^* \leq \frac{2L_{\gamma}}{(t+1)^2}\|x(0) - x^*\|^2_2,$$

where $x(0)$ is the initial iterate and $x^* \in \text{argmin}_{x \in \mathbb{R}^n} F(x)$ — see Corollary 3 in (Tseng, 2008), and Theorem 4.4 in (Beck & Teboulle, 2009). Thus, APG can compute an $\epsilon$-optimal solution to (3) within $O(\sqrt{T_{\epsilon^{-2}}} \epsilon^{-2})$ iterations.

As discussed above, the centralized APG algorithm cannot be applied when the nodes are unwilling or unable to communicate the privately known functions $\{F_i\}_{i \in N}$ to a central node. There are many other setting where one may want to solve (3) as a “distributed” problem. For instance, although $\text{prox}_{\rho_i}$ can be computed efficiently for all $t > 0$ and $i \in N$, $\text{prox}_{\gamma_i/L_{\gamma_i}}$ may be hard to compute. As an example, consider a problem with $\rho_i(X) = \sum_{i,j} |X_{ij}|$ and $\rho_2 = \sum_{i=1}^{\text{rank}(X)} \sigma_i(X)$, where $\sigma(X)$ denotes the vector of singular values for $X \in \mathbb{R}^{n_1 \times n_2}$. Here, $\text{prox}_{\rho_2}$ is easy to compute for all $t > 0$ and $i \in \{1, 2\}$; however, $\text{prox}_{(\rho_1 + \rho_2)}$ is hard to compute. Thus, the “centralized” APG algorithm cannot be applied. In the rest of this paper, we focus on decentralized algorithms.

2.2. D-FAL Algorithm for the Decentralized Model

Let $x = (x_1, \ldots, x_N)^T \in \mathbb{R}^{nN}$ denotes a vector formed by concatenating $\{x_i\}_{i \in N} \subset \mathbb{R}^n$ as a long column vector. Consider the following optimization problem of the form:

$$F^* := \min_{x \in \mathbb{R}^{nN}} \left\{ \hat{F}(x) := \hat{\rho}(x) + \gamma(x) \text{ s.t. } Ax = b \right\},$$

where $\hat{\rho}(x) := \sum_{i=1}^N \rho_i(x_i)$, $\hat{\gamma}(x) := \sum_{i=1}^N \gamma_i(x_i)$, and $A \in \mathbb{R}^{m \times nN}$ has $\text{rank}(A) = m$, i.e., the linear map is surjective. In Section 2.2.3, we show that the distributed optimization problem in (4) is a special case of (6), i.e., for all connected $\mathcal{G}$, there exists a surjective $A$ such that (4) is equivalent to (6). In the rest of the section, we will use the following notation: Let $\{A_i\}_{i \in N} \subset \mathbb{R}^{m \times nN}$ such that $A = [A_1, A_2, \ldots, A_N]$; $\tilde{L} := \max_{i \in N} L_{\gamma_i}$, $\tilde{\tau} := \min_{i \in N} \tau_i$.

We propose to solve (6) by inexactly solving the following sequence of subproblems in a distributed manner:

$$x_k \in \text{argmin}_{x \in \mathbb{R}^{nN}} P(k)(x) := \lambda(k) \hat{\rho}(x) + f(k)(x),$$

$$f(k)(x) := \lambda(k) \hat{\gamma}(x) + \frac{1}{2} \|Ax - b - \lambda(k) \theta(k)\|^2_2,$$

for appropriately chosen sequences of penalty parameters $\{\lambda(k)\}$ and dual variables $\{\theta(k)\}$ such that $\lambda(k) \searrow 0$. In particular, given $\{\alpha(k), \xi(k)\}$ satisfying $\alpha(k) \searrow 0$ and $\xi(k) \searrow 0$, the iterate sequence $\{x(k)\}$ is constructed such that every $x(k)$ satisfies one of the following conditions:

\begin{align}
&\left(a\right) \quad P(k)(x(k)) - P(k)(x^*) \leq \alpha(k), \\
&\left(b\right) \quad \|g_i(k)\|_{\mathcal{L}} \leq \xi(k), \quad \text{s.t.} \quad \max_{i \in N} \|g_i(k)\|_2 \leq \xi(k) \sqrt{\mathcal{G}},
\end{align}

where $\nabla f(k)(x)$ is Lipschitz continuous in $x \in \mathbb{R}^{nN}$ with constant $\lambda(k) \tilde{L} + \sigma_{\max}(A)$. Given $\{x(0), \lambda(0), \alpha(0), \xi(0)\}$ and $c \in (0, 1)$, we choose the sequence $\{\lambda(k), \alpha(k), \xi(k), \theta(k)\}$ as shown in Fig. 1.

Algorithm D-FAL $\left(\lambda^{(1)}, \alpha^{(1)}, \xi^{(1)}\right)$

Step 0: Set $\theta^{(1)} = 0$, $k = 1$

Step k: ($k \geq 1$)

1. Compute $x^{(k)}$ such that (9)(a) or (9)(b) holds
2. $\theta^{(k+1)} = \theta^{(k)} - \lambda^{(k)} \tilde{L} \sigma_{\max}(A)$
3. $\lambda^{(k+1)} = c \lambda^{(k)}$, $\alpha^{(k+1)} = c^2 \alpha^{(k)}$, $\xi^{(k+1)} = c^2 \xi^{(k)}$

\begin{figure}[h]
  \centering
  \includegraphics[width=\textwidth]{figure1.png}
  \caption{First-order Augmented Lagrangian algorithm}
\end{figure}

In Section 2.2.1, we show that D-FAL can compute an $\epsilon$-optimal and $\epsilon$-feasible $x_k$ to (6), i.e., $\|Ax_k - b\|_2 \leq \epsilon$ and $|F(x_k) - F^*| \leq \epsilon$, in at most $O(\log(1/\epsilon))$ iterations.

\begin{comment}
\begin{figure}[h]
  \centering
  \includegraphics[width=\textwidth]{figure1.png}
  \caption{First-order Augmented Lagrangian algorithm}
\end{figure}
\end{comment}
Next, in Section 2.2.2, we show that computing an \( \varepsilon \)-optimal, \( \varepsilon \)-feasible solution \( x^e \) requires at most \( \mathcal{O} \left( \min_{A \in \mathcal{A}} \left( \sigma_{\max}^2(A) \varepsilon^{-1} \right) \right) \) floating point operations. Using this result, in Section 2.2.3 we establish that DFAL can compute \( x^e \) in a distributed manner within \( \mathcal{O}(\varepsilon^{-1}) \) communication steps, i.e., the Main Result stated in Section 1. Finally, in Section 2.2.4 we show how to modify DFAL for an asynchronous computing set-up.

### 2.2.1. DFAL iteration complexity

We first show that \( \{x^{(k)}\} \) is a bounded sequence, and then argue that this also implies boundedness of \( \{\theta^{(k)}\} \). First, we start with a technical lemma that will be used in establishing the main results of this section.

**Lemma 1.** Let \( \bar{\rho} : \mathbb{R}^n \to \mathbb{R} \) be defined as \( \bar{\rho}(x) = \sum_{i \in \mathcal{N}} \rho_i(x_i) \), where \( \rho_i \in \mathcal{R} \) with uniform bound \( B_i \) on its subdifferential for all \( x \in \mathbb{R}^n \) and for all \( i \in \mathcal{N} \). Let \( f : \mathbb{R}^n \to \mathbb{R} \) denote a convex function such that there exist constants \( \{L_i\}_{i=1}^{\mathcal{N}} \subset \mathbb{R}_{++} \) that satisfy

\[
 f(y) \leq f(y) + \nabla f(\bar{y})^T (y - \bar{y}) + \frac{1}{2} \sum_{i=1}^{\mathcal{N}} L_i \|y_i - \bar{y}_i\|^2
\]

for all \( y, \bar{y} \in \mathbb{R}^n \). Given \( \alpha, \lambda \geq 0 \), and \( \bar{x} \in \mathbb{R}^n \) such that \( \lambda \rho(\bar{x}) + f(\bar{x}) - \min_{x \in \mathbb{R}^n} \{\lambda \rho(x) + f(x)\} \leq \alpha \), it follows that \( \|\nabla x_i f(\bar{x})\|_{\mathbb{R}^n} \leq \sqrt{2L_i \alpha} + \lambda B_i \) for all \( i \in \mathcal{N} \).

In Lemma 2 we show that function \( f^{(k)} \) defined in (8) satisfies the condition given in Lemma 1.

**Lemma 2.** The function \( f^{(k)} \) in (7) satisfies the condition in Lemma 1 with the constants \( L_i = L_i^{(k)} \), where \( L_i^{(k)} := \lambda^{(k)} L_{\gamma_i} + \sigma_{\max}^2(A_{i}) \) for all \( i \in \mathcal{N} \).

Lemma 1 and Lemma 2 allow us to bound \( \|\theta^{(k+1)}\|_2 \) in terms of \( \|\nabla x_i f(x^{(k)})\|_2 \) for all \( i \in \mathcal{N} \). We later use this bound in an inductive argument to establish that the sequence \( \{x^{(k)}\} \) is bounded.

**Lemma 3.** Let \( \{x^{(k)}\} \) be the DFAL iterate sequence, i.e., at least one of the conditions in (9) hold for all \( k \geq 1 \). Define \( \Theta_i^{(k)} := \max \left\{ \sqrt{2L_i^{(k)} \alpha^{(k)} (\lambda^{(k)} + \lambda^{(k)} B_i)} + B_i + \|\nabla \gamma_i(x^{(k)})\|_2 \right\} \). Then for all \( k \geq 1 \), we have

\[
\|\theta^{(k+1)}\|_2 \leq \min_{i \in \mathcal{N}} \left\{ \frac{\Theta_i^{(k)}}{\sigma_{\min}(A_i)} \right\}.
\]

Theorem 1 establishes that the DFAL iterate sequence \( \{x^{(k)}\} \) is bounded whenever \( \{\rho_i, \gamma_i\} \in \mathcal{N} \) satisfy Assumption 1; therefore, the sequence of dual variables \( \{\theta^{(k)}\} \) is bounded according to Lemma 3.

**Theorem 1.** Suppose Assumption 1 holds. Then there exist constants \( B_x, B_\theta, \bar{\lambda} > 0 \) such that \( \max \{\|x^{(k)}\|_2, \|x^{(k)}\|_2\} \leq B_x \) and \( \|\theta^{(k)}\|_2 \leq B_\theta \) for all \( k \geq 1 \), whenever \( \lambda^{(1)} \) and \( \varepsilon^{(1)} \) are chosen such that \( 0 < \lambda^{(1)} \leq \bar{\lambda} \) and \( \frac{\lambda^{(1)}}{\lambda^{(1)}} < \frac{\varepsilon^{(1)}}{\varepsilon^{(1)}} < \frac{\bar{\lambda}}{\bar{\lambda}} \).

We are now ready to state a key result that will imply the iteration complexity of DFAL.

**Theorem 2.** Suppose Assumption 1 holds and \( \lambda^{(1)} \) and \( \varepsilon^{(1)} \) are chosen according to Theorem 1. Then the primal-dual iterate sequence \( \{x^{(k)}, \theta^{(k)}\} \) generated by DFAL satisfy

\[
(a) \quad \|2x^{(k)} - b\|_2 \leq 2B_\theta \lambda^{(k)},
\]

\[
(b) \quad F(x^{(k)}) - F^* \geq -\lambda^{(k)} \left( \frac{\|\theta^{(k)}\|_2^2 + B_\theta^2}{2} \right),
\]

\[
(c) \quad F(x^{(k)}) - F^* \leq \lambda^{(k)} \left( \frac{B_\theta^2}{2} + \max_k \left\{ \Theta_i^{(1)}, \varepsilon^{(1)} B_i \right\} \right),
\]

where \( \theta^{*} \) denotes any optimal dual solution to (6).

**Corollary 1.** The DFAL iterates \( x^{(k)} \) are \( \varepsilon \)-feasible, i.e., \( \|A x^{(k)} - b\|_2 \leq \varepsilon \), and \( \varepsilon \)-optimal, i.e., \( |F(x^{(k)}) - F^*| \leq \varepsilon \), for all \( k \geq N(\varepsilon) \) and \( N(\varepsilon) = \log \frac{C}{\varepsilon} \) for some \( C > 0 \).

### 2.2.2. Overall computational complexity for the synchronous algorithm

Efficiency of DFAL depends on the complexity of the oracle for Step 1 in Fig. 1. In this section, we construct an oracle MS-APG that computes an \( x^{(k)} \) satisfying (9) within \( \mathcal{O}(1/\lambda^{(k)}) \) gradient and prox computations. This result together with Theorem 2 guarantees that for any \( \varepsilon > 0 \), DFAL can compute an \( \varepsilon \)-optimal and \( \varepsilon \)-feasible iterate within \( \mathcal{O}(\varepsilon^{-1}) \) floating point operations. Following lemma gives the iteration complexity of the oracle MS-APG displayed in Fig. 2.

**Lemma 4.** Let \( \bar{\rho} : \mathbb{R}^n \to \mathbb{R} \) such that \( \bar{\rho}(x) = \sum_{i \in \mathcal{N}} \rho_i(x_i) \), where \( \rho_i : \mathbb{R}^n \to \mathbb{R} \) is a convex function for all \( i \in \mathcal{N} \), and \( f : \mathbb{R}^n \to \mathbb{R} \) be a convex function such that it satisfies the condition in Lemma 1 for some constants \( \{L_i\}_{i=1}^{\mathcal{N}} \subset \mathbb{R}_{++} \). Suppose that \( y^* \in \argmin_{y \in \mathbb{R}^n} \Phi(y) := \bar{\rho}(y) + f(y) \). Then the MS-APG iterate sequence \( \{y^{(k)}\}_{k \in \mathbb{Z}_+} \), computed as in Fig. 2, satisfies

\[
0 \leq \Phi(y^{(k)}) - \min_{y \in \mathbb{R}^n} \Phi(y) \leq \frac{\sum_{i=1}^{\mathcal{N}} 2L_i \|y_i^{(0)} - y_i^{(k)}\|^2_{\mathbb{R}^n}}{(\ell + 1)^2}.
\]

**Proof.** (10) follows from adapting the proof of Theorem 4.4 in Beck & Teboulle (2009) for the case here. □

**Algorithm MS-APG (\( \bar{\rho}, f, y^{(0)} \))**

Step 0: Take \( y^{(0)} = y^{(0)}, \ell^{(1)} = 1 \)

Step 1: \( \ell^{(1)} \geq 1 \)

1. \( y_i^{(\ell)} = \text{prox}_{\rho_i/L_i} \left( y_i^{(\ell-1)} - \nabla y_i f(y^{(\ell-1)}/L_i) \right) \) \( \forall i \in \mathcal{N} \)

2. \( t^{(\ell+1)} = (1 + \sqrt{1 + 4 t^{(\ell)}})^2/2 \)

3. \( y^{(\ell+1)} = y^{(\ell)} + \frac{t^{(\ell+1)} - 1}{t^{(\ell+1)}} \left( y^{(\ell)} - y^{(\ell-1)} \right) \)

**Figure 2.** Multi Step - Accelerated Prox. Gradient (MS-APG) alg.
Consider the problem $\Phi^* = \min \Phi(y) := \tilde{b}(y) + f(y)$ defined in Lemma 4. Note that $\nabla f$ is Lipschitz continuous with constant $L = \max_{i \in \mathcal{N}} L_i$. In MS-APG algorithm, the step length $1/L_i \geq 1/L$ is different for each $i \in \mathcal{N}$. Instead, if one were to use the APG algorithm (Beck & Teboulle, 2009; Tseng, 2008), then the step length would have been $1/L$ for all $i \in \mathcal{N}$. When $\{L_i\}_{i \in \mathcal{N}}$ are close to each other, the performances of MS-APG and APG are on par; however, when $\max_{i \in \mathcal{N}} L_i \gg 1$, APG can only take very tiny steps for all $i \in \mathcal{N}$; hence, MS-APG is likely to converge much faster in practice.

Since the subproblem (7) is in the form given in Lemma 4, the following result immediately holds.

**Lemma 5.** The iterate sequence $\{y^{(t)}\}_{t \in \mathbb{Z}^+}$ generated when we call MS-APG($\lambda^{(k)} \tilde{b}, f^{(k)}, x^{(k-1)}$) satisfies $P^{(k)}(y^{(t)}) - P^{(k)}(x^{(k-1)}) \leq \alpha(t)$, for all $\ell \geq \sqrt{\sum_{i=1}^{N} \frac{2L_i^2(x^{(k-1)} - x_i^{(k)})^2}{2\alpha(t)}} - 1$, where $L_i^{(k)}$ is defined in Lemma 2 and $x_i^{(k)}$ represents the $i$-th block of $x^{(k)}$. Hence, one can compute $x^{(k)}$ satisfying (9) within $O(1/\lambda^{(k)})$ MS-APG iterations.

Theorem 2 and Lemma 5 together imply that DFAL can compute an $\epsilon$-feasible, and $\epsilon$-optimal solution to (6) within $O(1/\epsilon)$ MS-APG iterations. Due to space considerations, we will only state and prove this result for the case where $\nabla \tilde{y}$ is bounded in $\mathbb{R}^N$ since the bounds $B_B$ and $B_x$ are more simple for this case. Note that Huber-loss, logistic-loss, and fair-loss functions indeed have bounded gradients.

**Theorem 3.** Suppose that $\exists G_i > 0$ such that $\|\nabla \tilde{y}_i(x)\|_2 \leq G_i$ for all $x \in \mathbb{R}^n$ and for all $i \in \mathcal{N}$. Let $N_{\text{DFAL}}(\epsilon)$ and $N_f^{\text{DFAL}}(\epsilon)$ denote the number of DFAL-iters to compute an $\epsilon$-optimal, and an $\epsilon$-feasible solutions to (6), respectively. Let $N(\epsilon)$ denote MS-APG iteration number required to compute $x^{(k)}$ satisfying at least one of the conditions in (9). Then

$$
\sum_{k=1}^{N_{\text{DFAL}}(\epsilon)} N(k) = O\left(\Theta \sigma_{\text{max}}(A)^{-1}\epsilon^{-1}\right),
$$

$$
\sum_{k=1}^{N_f^{\text{DFAL}}(\epsilon)} N(k) = O\left(\Theta \sigma_{\text{max}}(A)^{-1}\epsilon^{-1}\right),
$$

where $\Theta = \min_{i \in \mathcal{N}} \sigma_{\text{max}}(A_i)$.  

### 2.2.3. Synchronous Algorithm for Distributed Optimization

In this section, we show that the decentralized optimization problem (4) is a special case of (6); therefore, Theorem 3 establishes the Main Result stated in the Introduction. We also show that the steps in DFAL can be further simplified in this context.

Construct a directed graph by introducing an arc $(i, j)$ where $i < j$ for every edge $(i, j)$ in the undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. Then the constraints $x_i - x_j = 0$ for all $(i, j) \in \mathcal{E}$ in the distributed optimization problem (4) can be reformulated as $Cx = 0$, where $C \in \mathbb{R}^{n|\mathcal{E}| \times nN}$ is a block matrix such that the block $C_{(i,j)} \in \mathbb{R}^{n \times n}$ corresponding to the edge $(i, j) \in \mathcal{E}$ and node $l \in \mathcal{N}$, i.e., $C_{(i,j),l}$ is equal to $I_n$ if $l = i$, $-I_n$ if $l = j$, and $0_n$ otherwise, where $I_n$ and $0_n$ denote $n \times n$ identity and zero matrices, respectively. Let $\Omega \in \mathbb{R}^{nN \times nN}$ be the Laplacian of $\mathcal{G}$, i.e., for all $i \in \mathcal{N}$, $\Omega_{ij} = d_i$, and for all $(i,j) \in \mathcal{N} \times \mathcal{N}$ such that $i \neq j$, $\Omega_{ij} = -1$ if either $(i,j) \in \mathcal{E}$ or $(j,i) \in \mathcal{E}$, where $d_i$ denotes the degree of the $i \in \mathcal{N}$. Then it follows that

$$
\Psi := C^T C = \Omega \otimes I_n,
$$

where $\otimes$ denotes the Kronecker product. Let $\psi_{\max} := \psi_1 \geq \psi_2 \geq \ldots \geq \psi_N$ be the eigenvalues of $\Omega$. Since $\mathcal{G}$ is connected, $\text{rank}(\Omega) = N - 1$, i.e., $\psi_{N-1} > 0$ and $\psi_N = 0$. From the structure of $\Psi$ it follows that that $\{\psi_i\}_{i=1}^N$ are also the eigenvalues of $\Psi$, each with algebraic multiplicity $n$. Hence, $\text{rank}(C) = n(N - 1)$.

Let $C = U \Sigma V^T$ denote the reduced singular value decomposition (SVD) of $C$, where $U \in \mathbb{R}^{n|\mathcal{E}| \times n(N-1)}$, $\Sigma = \text{diag}(\sigma)$, $\sigma \in \mathbb{R}^{n(N-1)}$, and $V \in \mathbb{R}^{n(N-1) \times nN}$. Note that $\sigma^2_{\max}(C) = \psi_{\max}$, and $\sigma^2_{\min}(C) = \psi_{N-1}$. Define $A := \Sigma V^T$. $A \in \mathbb{R}^{n(N-1) \times nN}$ has linearly independent rows; more importantly, $A^T A = C^T C = \Psi$; hence, $\sigma^2_{\max}(A) = \psi_{\max}$, and $\sigma^2_{\min}(A) = \psi_{N-1}$. We also have $\{x \in \mathbb{R}^{nN} : Ax = 0\} = \{x \in \mathbb{R}^{nN} : Cx = 0\}$. Hence, the general problem in (6) with $A := \Sigma V^T$ and $b = 0 \in \mathbb{R}^{n(N-1)}$ is equivalent to (4). Let $A_i \in \mathbb{R}^{n(N-1) \times n}$ and $C_i \in \mathbb{R}^{n|\mathcal{E}| \times nN}$ be the submatrices of $A$ and $C$, respectively, corresponding to $x_i$, i.e., $A = [A_1, A_2, \ldots, A_N]$, and $C = [C_1, C_2, \ldots, C_N]$. Clearly, it follows from the definition of $C$ that $\sigma_{\max}(C_i) = \sigma_{\min}(C_i) = \sqrt{d_i}$ for all $i \in \mathcal{N}$. Using the property of SVD, it can also be shown for $A = \Sigma V^T$ that $\sigma_{\max}(A_i) = \sigma_{\min}(A_i) = \sqrt{d_i}$ for all $i \in \mathcal{N}$. Thus, Theorem 3 establishes the Main Result.

We now show that we do not have to compute the SVD of $C$, or $A$, or even the dual multipliers $\theta^{(k)}$ when DFAL is used to solve (4). In DFAL the matrix $A$ is used in Step 1 (i.e. within the oracle MS-APG) to compute $\nabla f^{(k)}$, and in Step 2 to compute $\theta^{(k+1)}$. Since $\theta^{(1)} = 0$, Step 2 in DFAL and (8) imply that $\theta^{(k+1)} = -\sum_{k=1}^{k} A_{\theta^{(k)}}$, and $\nabla f^{(k)}(x) = \lambda^{(k)} \nabla \gamma_{I}(x) + \lambda^{(k)} (Ax - \lambda^{(k)} \theta^{(k)}) = \lambda^{(k)} \nabla \gamma_{I}(x) + \psi (x + \lambda^{(k)} \sum_{t=1}^{k-1} \frac{1}{\lambda^{(t)}} x^{(t)})$. Moreover, from the definition of $\Psi$, it follows that

$$
\nabla x, f^{(k)}(x) = \lambda^{(k)} \nabla \gamma_{I}(x) + d_i \left( x_i + \tilde{x}_i^{(k)} \right) - \sum_{j \in O_i} \left( x_j + \tilde{x}_j^{(k)} \right),
$$

where $\tilde{x}^{(k)} := \sum_{t=1}^{k-1} \frac{\lambda^{(t)}}{\lambda^{(t)}} x^{(t)}$, and $O_i$ denotes the set of nodes adjacent to $i \in \mathcal{N}$. Thus, it follows that Step 1 of MS-APG can be computed in a distributed manner by only communicating with the adjacent nodes without explicitly computing $\theta^{(k)}$ in Step 2 of DFAL.
In particular, for the k-th DFAL iteration, each node $i \in \mathcal{N}$ stores $\bar{x}_i^{(k)}$ and $\{\bar{x}_j^{(k)}\}_{j \in \mathcal{O}_i}$, which can be easily computed locally if $\{\bar{x}_j^{(k)}\}_{j \in \mathcal{O}_i}$ is transmitted to $i$ at the end of Step 1 of the previous DFAL iterations $1 \leq t < k - 1$. Hence, during the $\ell$-th iteration of MS-APG ($\lambda^{(k)} \bar{\rho}, f^{(k)}, x^{(k-1)}$) call, each node $i \in \mathcal{N}$ can compute $\nabla y_i f^{(k)}(\bar{y}^{(k)})$ locally if $\{\bar{y}_j^{(k)}\}_{j \in \mathcal{O}_i}$ is transmitted to $i$ at the end of Step 3 in MS-APG. It is important to note that every node can independently check (9)(b), i.e., $\exists y_i^{(k)} \in \partial \rho_i(x_i) \mid x_i = x_i^{(k)} + \nabla x_i f^{(k)}(x^{(k)})$ for all $i \in \mathcal{N}$ such that $\max_{i \in \mathcal{N}} ||y_i^{(k)}||_2 \leq \frac{\varepsilon}{\sqrt{N}}$. Hence, nodes can reach a consensus to move to the next DFAL iteration without communicating their private information. If (9)(b) does not hold for $\ell_{\text{max}}^{(k)} := B_2 \sqrt{\frac{2 \sum_{i \in \mathcal{N}} \ell_i^{(k)}}{\alpha k}}$ MS-APG iterations, then Lemma 5 implies that (9)(a) must be true. Hence, all the nodes move to the next DFAL iteration after $\ell_{\text{max}}^{(k)}$ many MS-APG updates. For implementable version of DFAL, see Figure 3, where $B_2$ is the bound in Theorem 1, and $\mathcal{N}_t := \mathcal{N}_0 \cup \{t\}$.

**2.2.4. Asynchronous implementation**

Here we propose an asynchronous version of DFAL. Due to limited space, and for the sake of simplicity of the exposition, we only consider a simple randomized block coordinate descent (RBCD) method, which will lead to an asynchronous implementation of DFAL that can compute an $\epsilon$-optimal and $\epsilon$-feasible solution to (4) with probability $1 - p$ within $O\left(\frac{1}{\epsilon^2} \log \left(\frac{1}{p}\right)\right)$ RBCD iterations. In Section 4.9 of the supplementary file, we discuss how to improve this rate to $O\left(\frac{1}{\epsilon^2} \log \left(\frac{1}{p}\right)^3\right)$ using an accelerated RBCD.

**Algorithm RBCD ($\bar{\rho}, f, y^{(0)}$)**

Step $\ell$ ($\ell \geq 0$)
1. $i \in \mathcal{N}$ is realized with probability $\frac{1}{N}$
2. $y_i^{(\ell+1)} = \text{prox}_{\rho_i/L_i} \left( y_i^{(\ell)} - \nabla y_i f(y^{(\ell)}) / L_i \right)$
3. $y^{(\ell+1)} = y^{(\ell)}$

**Figure 4. Randomized Block Coordinate Descent (RBCD) alg.**

Nesterov (2012) proposed an RBCD method for solving $\min_{y \in \mathbb{R}^n} f(y)$, where $f$ is convex with block Lipschitz continuous gradient, i.e., $\nabla y_i f(y_i; y - i)$ is Lipschitz continuous in $y_i$ with constant $L_i$ for all $i$. Later, Richtárik & Takáč (2012) extended the convergence rate results to $\min_{y \in \mathbb{R}^n} \Phi(y) := \sum_{i=1}^{N} \rho_i(y_i) + f(y)$, such that $\text{prox}_{\rho_i}$ can be computed efficiently for all $\ell > 0$ and $i \in \mathcal{N}$, and established that given $\alpha > 0$, and $p \in (0, 1)$, for $\ell \geq \frac{2NC}{\alpha} \left( 1 + \log \frac{1}{p} \right)$, the iterate sequence $\{y^{(\ell)}\}$ computed by RBCD displayed in Fig. 4 satisfies

$$P(\Phi(y^{(\ell)}) - \Phi^* \leq \alpha) \geq 1 - p,$$

where $C := \max \{R_2^2(y^{(0)}), \Phi(y^{(0)}) - \Phi^*\}$, $R_2^2(y^{(0)}) := \max_{y, y^*} \left\{ \sum_{i=1}^{N} L_i \|y_i - y_i^*\|_2^2 : \Phi(y) \leq \Phi(y^*), y^* \in \mathcal{Y}^* \right\}$, and $\mathcal{Y}^*$ denotes the set of optimal solutions. RBCD is significantly faster in practice for very large scale problems, particularly when the partial gradient $\nabla y_i f(y)$ can be computed more efficiently as compared to the full gradient $\nabla f(y)$. The RBCD algorithm can be implemented for the distributed minimization problem when the nodes in $\mathcal{G}$ work asynchronously. Assume that for any $y = (y_i)_{i \in \mathcal{N}} \in \mathbb{R}^{N \times N}$, each node $i$ is equally likely to be the first to complete computing $\text{prox}_{\rho_i/L_i} (y_i - \nabla y_i f(y^{(\ell)}) / L_i)$, i.e., each node has an exponential clock with equal rates. Suppose node $i \in \mathcal{N}$ is the first node to complete Step 2 of RBCD. Then, instead of waiting for the other nodes to finish, node $i$ sends a message to its neighbors $j \in \mathcal{O}_i$ to terminate their computations, and shares $y_i^{(\ell+1)}$ with them. Note that RBCD can be easily incorporated into DFAL as an oracle to solve subproblems in (7) by replacing (9)(a) with

$$\mathbb{P} \left( P^{(k)}(x^{(k)}) - P^{(k)}(x^*) \leq \alpha^{(k)} \right) \geq (1 - p)^{\frac{1}{\alpha^{(k)}}},$$

(12)
where $N(e) = \log_2 \left( \frac{C}{e} \right)$ defined in Corollary 1. Since $(1-p) \frac{\log N}{N} \leq 1 - \frac{p}{\log N}$ for $p \in (0, 1)$, the total number of RBCD iterations for the $k$-th subproblem is bounded: $N^{(k)} \leq O\left( 2^k \log \left( \log \frac{1}{p} \right) \right) = O\left( \frac{1}{p} \log \left( \frac{1}{p} \right) + \log \left( \frac{1}{p} \right) \right)$. Hence, Corollary 1 and (11) imply that asynchronous DFAL, i.e., (9)(a) replaced with (12), can compute an $\epsilon$-optimal and $\epsilon$-feasible solution to (4) with probability $1 - p$ within $O\left( \frac{1}{p} \log \left( \frac{1}{p} \right) \right)$ RBCD iterations. These results can be extended to the case where each node has different clock rates using (Qu & Richtárik, 2014).

3. Numerical results

In this section, we compared DFAL with an ADMM method proposed in (Makhdoumi & Ozdaglar, 2014) on the sparse group LASSO problem with Huber loss:

$$
\min_{x \in \mathbb{R}^N} \sum_{i=1}^N \beta_1 \|x\|_1 + \beta_2 \|x\|_{G,i} + \sum_{j=1}^{m_i} h_\delta \left( a_i^T(j)x - b_i \right),
$$

where $\beta_1, \beta_2 > 0$, $A_i \in \mathbb{R}^{m_i \times \infty}$, $b_i \in \mathbb{R}^{m_i}$, $a_i^T(j)$ denotes the $j$-th row of $A_i$, the Huber loss function $h_\delta(x) = \max\{tx - t^2/2 : t \in [-\delta, \delta]\}$, and $\|x\|_{G,i} := \sum_{k=1}^K \|x_{g,(k)}\|_2$ denotes the group norm with respect to the partition $G_i$ of $[1, n] := \{1, \ldots, n\}$ for all $i \in \mathcal{N}$, i.e., $G_i = \{g_i(k)\}_{k=1}^K$ such that $\bigcup_{k=1}^K g_i(k) = [1, n]$, and $g_i, g_j$, and $g_{i j} \in G_i$ for all $j \neq k$. In this case, $\gamma_i(x) := \sum_{j=1}^{m_i} h_\delta(a_i^T(j)x - b_i)$ and $\rho_i(x) := \beta_1 \|x\|_1 + \beta_2 \|x\|_{G,i}$. Next, we briefly describe the ADMM algorithm in (Makhdoumi & Ozdaglar, 2014), and propose a more efficient variant, SADMM, for (13).

Algorithm SADMM ($c, x^{(0)}$)

Initialization: $y^{(0)} = x^{(0)}$, $p_1^{(k)} = \hat{p}_1^{(k)} = 0$, $i \in \mathcal{N}$

Step $t$: ($t \geq 0$) for $i \in \mathcal{N}$ compute

1. $x_i^{(k+1)} = \text{prox}_{\gamma_i} \left( x_i^{(k)} - \frac{1}{\delta} \left( r_j^{(k)} - a_i^T g_i^{(k)}(x_i^{(k)})/d_i 1 + 1 \right) \right)$
2. $y_i^{(k+1)} = \text{prox}_{\gamma_i} \left( y_i^{(k)} - \frac{1}{\delta} \left( r_j^{(k)} - a_i^T g_i^{(k)}(x_i^{(k)})/d_i 1 + 1 \right) \right)$
3. $s_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} \Omega_{i j} x_j^{(k+1)}/d_i 1 + 1$
4. $p_i^{(k+1)} = \hat{p}_i^{(k)} + s_i^{(k+1)}$
5. $\hat{p}_i^{(k+1)} = p_i^{(k+1)} - s_i^{(k+1)}$

Figure 5. Split ADMM algorithm

3.1. A distributed ADMM Algorithm

Let $\Omega \in \mathbb{R}^{N \times N}$ denote the Laplacian of the graph $G = (\mathcal{N}, \mathcal{E})$, $\mathcal{O}_i$ denote the set of neighboring nodes of $i \in \mathcal{N}$, and define $\mathcal{N}_i := \mathcal{O}_i \cup \{i\}$. Let $Z_i := \{z_i \in \mathbb{R}^{d_i + 1} : \sum_{j \in \mathcal{N}_i} z_{ij} = 0\}$. Makhdoumi & Ozdaglar (2014) show that (4) can be equivalently written as

$$
\min_{x_i \in \mathbb{R}^{n_i}, z_{ij} \in Z_i} \left\{ \sum_{i=1}^N F_i(x_i) = \rho_i(x_i) + \gamma_i(x_i) \right\}
\text{s.t. } \Omega_{ij} x_j = z_{ij}, \quad i \in \mathcal{N}, \ j \in \mathcal{N}_i.
$$

Only (14) is penalized when forming the augmented Lagrangian, which is alternatingly minimized in $x \in \mathbb{R}^{n N}$, $z_i^T = [z_{i1}, \ldots, z_{in_i}]$, where $Z_i \ni z_i = [z_{ij}]_{j \in \mathcal{N}_i} \in \mathbb{R}^{d_i + 1}$. Makhdoumi & Ozdaglar (2014) establish that suboptimality and consensus violation converge to 0 with a rate $O(1/k)$, and in each iteration every node communicates $3n$ scalars. From now on, we refer to this algorithm that directly works with $F_i$ as ADMM. Computing $\text{prox}_{p_i}^g$ for each $i \in \mathcal{N}$ is the computational bottleneck in each iteration of ADMM. Note that computing $\text{prox}_{g_i}^g$ for (13) is almost as hard as solving the problem. To deal with this issue, we considered the following reformulation:

$$
\min_{x_i, y_i \in \mathbb{R}^{n_i}} \sum_{i \in \mathcal{N}} \rho_i(x_i) + \gamma_i(y_i)
\text{s.t. } \Omega_{ij} x_j = z_{ij}, \quad i \in \mathcal{N}, \ j \in \mathcal{N}_i
\Omega_{ij} y_j = z_{ij}, \quad i \in \mathcal{N}, \ j \in \mathcal{N}_i
x_i = q_i, \ y_i = q_i, \ i \in \mathcal{N}.
$$

ADMM algorithm for this formulation is displayed in Fig. 5, where $c > 0$ denotes the penalty parameter. Steps of ADMM can be derived by minimizing the augmented Lagrangian alternately in $(x, y)$, and in $(z, \bar{z}, q)$ while fixing the other. As in (Makhdoumi & Ozdaglar, 2014), computing $(z, \bar{z}, q)$ can be avoided by exploiting the structure of optimality conditions. Prox centers in ADMM are $x_i^{(k+1)} = x_i^{(k)} - \sum_{j \in \mathcal{N}_i} \Omega_{ij} (x_j^{(k+1)} + r_j^{(k)})/d_i 1 + 1 - \gamma_i^{(k)}$, $y_i^{(k+1)} = y_i^{(k)} - \sum_{j \in \mathcal{N}_i} \Omega_{ij} (y_j^{(k+1)}) - r_j^{(k)} / d_i 1 + 1$, respectively; and $r_i^{(k+1)} = r_i^{(k)} + (r_i^{(k+1)} - y_i^{(k+1)})/2$.

3.2. Implementation details and numerical results

The following lemma shows that in DFAL implementation, each node $i \in \mathcal{N}$ can check (9)(b) very efficiently. For $x \in \mathbb{R}^n$, define $\text{sgn}(x) = 1$, 0 and 1 when $x < 0$, $x = 0$, and $x > 0$, respectively; and for $x \in \mathbb{R}^n$, define $\text{sgn}(x) = \text{sgn}(x_1), \text{sgn}(x_2), \ldots, \text{sgn}(x_n)^T$.

Lemma 6. Let $f : \mathbb{R}^n \to \mathbf{R}$ be a differentiable function, $G = \{g_i(\mathbf{x})\}_{i=1}^K$ be a partition of $[1, n]$, $\beta_1, \beta_0, \beta_2 > 0$, be define $P = \lambda \rho + f$, where $\rho(x) := \beta_1 \|x\|_1 + \beta_2 \|x\|_2$. Then, for all $x \in \mathbb{R}^n$ and $z > 0$, there exists $\nu \in \partial P(x) \mid x = \bar{x}$ such that $\|\nu\|_2 \leq \xi$ if and only if $\|\nu^* + \omega^* + \nabla f(x)\|_2 \leq \xi$ for $\nu^*, \omega^*$ such that for each $k$, $\|\nu_{g(k)}\| \neq 0$, then $\nu_{g(k)}^* = \lambda \beta_1 \text{sgn}(x_{g(k)}) + (1 - \text{sgn}(|x_{g(k)}|)) \circ \eta_{g(k)}$, and $\omega_{g(k)}^* = \lambda \beta_2 x_{g(k)}^* x_{g(k)}^* \circ \eta_{g(k)}$, otherwise, if $x_{g(k)} = 0$, then $\nu_{g(k)}^* = \eta_{g(k)}$, and $\omega_{g(k)}^* = 0$.

$$
\min \left\{ \frac{\nu^* + \nabla f(x)}{\eta}, \quad \nu^* + \nabla f(x) \right\} = \left\{ \frac{1}{\eta}, \frac{\lambda \beta_2}{\eta} \right\}
\text{min} \left\{ \frac{\nu^* + \nabla f(x)}{\eta}, \lambda \beta_2 \right\}
\text{where } \circ \text{ denotes componentwise multiplication, and } \eta_{g(k)} = -\text{sgn} \left( \nabla x_{g(k)} f(x) \right) \circ \text{min} \left\{ \nabla x_{g(k)} f(x), \lambda \beta_1 \right\}.
$$
Both DFAL and SADM call for $\text{prox}_{\rho_i} \beta_i$. In Lemma 7, we show that it can be computed in closed form. On the other hand, when ADMM, and SADM call are implemented on (13), one needs to compute $\text{prox}_{\rho_i} f_i$ and $\text{prox}_{\rho_i} \gamma_i$, respectively; however, these proximal operations do not assume closed form solutions. Therefore, in order to be fair, we computed them using an efficient interior point solver MOSEK (ver. 7.1.0.12).

**Lemma 7.** Let $\rho(x) = \beta_1 \|x\|_1 + \beta_2 \|x\|_G$. For $t > 0$ and $x \in \mathbb{R}^n$, $x^\ast = \text{prox}_{\rho/n}(x)$ is given by $x^\ast = \theta_{g(x)} \max \left\{1 - \frac{\beta_2}{\|g(x)\|^2}, 0\right\}$, for $1 \leq k \leq K$, where $\theta = \text{sgn}(\tilde{x}) \circ \max\{|\tilde{x}| - \beta_1, 0\}$.

In our experiments, the network was either a star tree or a clique with either 5 or 10 nodes. The remaining problem parameters defining $\{\rho_i, \gamma_i\}_{i \in N}$ were set as follows. We set $\beta_1 = \beta_2 = \frac{1}{2}$, $\delta = 1$, and $K = 10$. Let $n = Kn_g$ for $n_g \in \{100, 300\}$, i.e., $n \in \{100, 300\}$.

We generated partitions $\{G_i\}_{i \in N}$ in two different ways. For test problems in CASE 1, we created a single partition $G = \{g(k)\}_{k=1}^n$ by generating $K$ groups uniformly at random such that $|g(k)| = n_g$ for all $k$; and set $G_i = G$ for all $i \in N$, i.e., $\rho_i(x) = \rho(x) = \beta_1 \|x\|_1 + \beta_2 \|x\|_G$ for all $i \in N$. For the test problems in CASE 2, we created a different partition $G_i$ for each node $i$, in the same manner as in CASE 1. For all $i \in N$, $m_i = \frac{n}{K}$, and the elements of $A_i \in \mathbb{R}^{m_i \times n}$ are i.i.d. with standard Gaussian, and we set $b_i = A_i \tilde{x}$ for $\tilde{x} = (-1)^j e^{(-j-1)/n}$ for $j \in [1, n]$.

We solved the distributed optimization problem (4) using DFAL, AFAL (asynchronous version of DFAL with accelerated RBBCD –see the supplementary file for details), ADMM, and SADM for both cases, on both star trees, and cliques, and for $N \in \{5, 10\}$ and $n_g \in \{100, 300\}$. For each problem setting, we randomly generated 5 instances. For benchmarking, we solved the centralized problem (3) using SDPT3 for both cases. Note that for CASE 1, $\sum_{i \in N} \rho_i(x)$ does not assume a simple prox map. Therefore, for the first case we were also able to use APG, described in Section 2.1, to solve (3) by exploiting the result of Lemma 7. All the algorithms are terminated when the relative suboptimality, $|F(k) - F^*|/|F^*|$, is less than $10^{-3}$, and consensus violation, $CV(k)$, is less than $10^{-4}$, where $F(k)$ equals to $\sum_{i \in N} F_i(x_i^k)$ for DFAL and ADMM, and to $\sum_{i \in N} F_i(x_i^k)$ for SADM; $CV(k)$ equals to $\max_{(i,j) \in E} \|x_i^k - x_j^k\|_2/\sqrt{n}$ for DFAL and SADM, and to $\max_{(i,j) \in E} \|x_i^k - x_j^k\|_2, \max_{i \in N} \|x_i^k - y_i(0)\|_2/\sqrt{n}$ for SADM. If the stopping criteria are not satisfied in 30min., we terminated the algorithm and report the statistics corresponding to the iterate at the termination.

In Table 2, `xxx (C)` stands for “algorithm xxx is used to solve the centralized problem”. Similarly, `xxx (D)` for the decentralized one. For the results presented by comma, the left and right ones are for the star tree and clique, resp. Table 2 displays the means over 5 replications for each case. The number of iterations in each case clearly illustrates the topology of the network plays an important role in the convergence speed of DFAL, which coincides to our analysis in Section 2.2.2.

<table>
<thead>
<tr>
<th>Size</th>
<th>Alg.</th>
<th>Rel. Suboptimality</th>
<th>Consensus Violation (CV)</th>
<th>CPU Time (sec.)</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Case 1</td>
<td>Case 2</td>
<td>Case 1</td>
<td>Case 2</td>
<td>Case 1</td>
</tr>
<tr>
<td>$n_g = 100$</td>
<td>SDPT3 (C)</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>85</td>
</tr>
<tr>
<td>$N = 5$</td>
<td>APG (C)</td>
<td>1E-3</td>
<td>N/A</td>
<td>10</td>
<td>N/A</td>
</tr>
<tr>
<td>$n_g = 100$</td>
<td>DFAL (D)</td>
<td>6E-4, 7E-4</td>
<td>6E-4, 4E-4</td>
<td>5E-5, 2E-5</td>
<td>5E-5, 3E-5</td>
</tr>
<tr>
<td>$N = 5$</td>
<td>ADMM (D)</td>
<td>3E-4, 4E-4</td>
<td>4E-4, 3E-4</td>
<td>2E-6, 5E-6</td>
<td>2E-6, 5E-6</td>
</tr>
<tr>
<td>$n_g = 100$</td>
<td>SDPT3 (C)</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>89</td>
</tr>
<tr>
<td>$N = 10$</td>
<td>APG (C)</td>
<td>1E-3</td>
<td>N/A</td>
<td>10</td>
<td>N/A</td>
</tr>
<tr>
<td>$n_g = 100$</td>
<td>DFAL (D)</td>
<td>4E-4, 7E-4</td>
<td>4E-4, 3E-4</td>
<td>6E-5, 1E-5</td>
<td>6E-5, 2E-5</td>
</tr>
<tr>
<td>$N = 10$</td>
<td>ADMM (D)</td>
<td>9E-3, 2E-2</td>
<td>8E-3, 1E-2</td>
<td>8E-4, 4E-4</td>
<td>8E-4, 4E-4</td>
</tr>
<tr>
<td>$n_g = 100$</td>
<td>SDPT3 (C)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$N = 10$</td>
<td>APG (C)</td>
<td>1E-3</td>
<td>N/A</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>$n_g = 100$</td>
<td>DFAL (D)</td>
<td>2E-4, 5E-4</td>
<td>2E-4, 4E-4</td>
<td>5E-5, 4E-5</td>
<td>5E-5, 5E-5</td>
</tr>
<tr>
<td>$N = 10$</td>
<td>ADMM (D)</td>
<td>9E-4, 2E-3</td>
<td>8E-4, 1E-2</td>
<td>5E-3, 1E-3</td>
<td>5E-3, 1E-3</td>
</tr>
<tr>
<td>$n_g = 100$</td>
<td>SDPT3 (C)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$N = 10$</td>
<td>APG (C)</td>
<td>1E-3</td>
<td>N/A</td>
<td>0</td>
<td>N/A</td>
</tr>
</tbody>
</table>
References


