# An Asynchronous Distributed Proximal Gradient Method for Composite Convex Optimization

**N. S. Aybat** NSA10@PSU.EDU
**Z. Wang** ZXW121@PSU.EDU
Penn State University, University Park, PA 16802 USA

**G. Iyengar** GARUD@IEOR.COLUMBIA.EDU
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 $\mathcal{O}(\log(\epsilon^{-1}))$ **DFAL** iterations, which require $\mathcal{O}(\frac{\psi_{\max}^{1.5}}{d_{\min}}\ \epsilon^{-1})$ 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

$$F_i(x) := \rho_i(x) + \gamma_i(x), \tag{1}$$

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

$$\mathbf{prox}_{\rho_i}(x) := \underset{y \in \mathbb{R}^n}{\operatorname{argmin}} \left\{\rho_i(y) + \tfrac{1}{2}\|y - x\|_2^2\right\} \tag{2}$$

is *efficiently* computable for $i \in \mathcal{N}$.

*Proceedings of the $32^{nd}$ International Conference on Machine Learning*, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

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).

| **Reference** | **assumption on $F_i$** | operation / iter. | iter # for $\epsilon$-feas. | iter # for $\epsilon$-opt. | comm. steps $\epsilon$-opt. | asnych. | Can handle constraints? |
|---|---|---|---|---|---|---|---|
| Duchi et al. (2012) | convex, Lipschitz cont. | subgrad., projection | unknown | $\mathcal{O}(1/\epsilon^2)$ | $\mathcal{O}(1/\epsilon^2)$ | no | no |
| Nedic & Ozdaglar (2009) | convex | subgrad. | $\mathcal{O}(1)$ | $\mathcal{O}(1/\epsilon^2)$ | $\mathcal{O}(1/\epsilon^2)$ | no | no |
| Wei & Ozdaglar (2012) | strictly convex | $\mathbf{prox}_{F_i}$ | unknown | $\mathcal{O}(1/\epsilon)$ | $\mathcal{O}(1/\epsilon)$ | no | no |
| Makhdoumi & Ozdaglar (2014) | convex | $\mathbf{prox}_{F_i}$ | $\mathcal{O}(1/\epsilon)$ | $\mathcal{O}(1/\epsilon)$ | $\mathcal{O}(1/\epsilon)$ | no | no |
| Wei & Ozdaglar (2013) | convex | $\mathbf{prox}_{F_i}$ | $\mathcal{O}(1/\epsilon)$ | $\mathcal{O}(1/\epsilon)$ | $\mathcal{O}(1/\epsilon)$ | yes | no |
| Jakovetic et al. (2011) | smooth convex bounded $\nabla F_i$ | $\nabla F_i$ | $\mathcal{O}(1/\sqrt{\epsilon})$ | $\mathcal{O}(1/\sqrt{\epsilon})$ | $\mathcal{O}(1/\sqrt{\epsilon})$ | no | no |
| Chen & Ozdaglar (2012) | composite convex $F_i = \rho + \gamma_i$ bounded $\nabla\gamma_i$ | $\mathbf{prox}_\rho$, $\nabla\gamma_i$ | $\mathcal{O}(1/\sqrt{\epsilon})$ | $\mathcal{O}(1/\sqrt{\epsilon})$ | $\mathcal{O}(1/\epsilon)$ | no | no |
| Our work | composite convex $F_i = \rho_i + \gamma_i$ | $\mathbf{prox}_{\rho_i}$, $\nabla\gamma_i$ | $\mathcal{O}(1/\epsilon)$ | $\mathcal{O}(1/\epsilon)$ | $\mathcal{O}(1/\epsilon)$ | **yes** | **yes** |

*Table 1.* Comparison of our method with the previous work

We call a solution $\bar{\mathbf{x}} = (\bar{x}_i)_{i\in\mathcal{N}}$ $\epsilon$-feasible if the consensus violation $\max_{(i,j)\in\mathcal{E}}\left\{\|\bar{x}_i - \bar{x}_j\|_2\right\} \leq \epsilon$ and $\epsilon$-optimal if $\left|\sum_{i\in\mathcal{N}} F_i(\bar{x}_i) - F^*\right| \leq \epsilon$. 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 $\{\mathbf{x}^{(k)}\}_{k\in\mathbb{Z}_+}$ denote the sequence of **DFAL** iterates. Then $F^* = \lim_{k\in\mathbb{Z}_+}\sum_{i\in\mathcal{N}} F_i(x_i^{(k)})$. Furthermore, $\mathbf{x}^{(k)}$ is $\epsilon$-optimal and $\epsilon$-feasible within $\mathcal{O}(\log(\epsilon^{-1}))$ **DFAL** iterations, requiring $\mathcal{O}\big(\frac{\psi_{\max}^{1.5}}{d_{\min}}\epsilon^{-1}\big)$ communications per node, and $\mathcal{O}(\epsilon^{-1})$ 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 $d_{\min}$ 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 $\mathcal{O}(1/\epsilon^2)$ iterations; however, they do not provide any guarantees on the consensus violation $\max\{\|\bar{x}_i - \bar{x}_j\|_2 : (i,j)\in\mathcal{E}\}$. 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 $\alpha = \mathcal{O}(\epsilon)$ in their method guarantees that consensus violation and suboptimality is $\mathcal{O}(\epsilon)$ in $\mathcal{O}(1/\epsilon^2)$ 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 $\mathcal{O}(1/\epsilon)$ 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 $F_i = \rho_i + \gamma_i$ is hard -see Section 3 for an example. One can overcome this limitation of ADMM by locally splitting variables, i.e., setting $F_i(x_i, y_i) := \rho_i(x_i) + \gamma_i(y_i)$, 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 $\mathcal{O}(1/\sqrt{\epsilon})$ 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 $\mathcal{O}(\epsilon^{-1/2})$ iterations which require $\mathcal{O}(\epsilon^{-1})$ communications per node over a time-varying network topology when $F_i = \rho + \gamma_i$, assuming that the non-smooth term $\rho$ is the *same* at all nodes, and $\nabla\gamma_i$ is *bounded* for all $i\in\mathcal{N}$. In contrast, **DFAL** proposed in this paper is able to *asynchronously* compute an $\epsilon$-optimal $\epsilon$-feasible solution in $\mathcal{O}(\epsilon^{-1})$ communications per node, allowing node specific non-smooth functions $\rho_i$, and without assuming bounded $\nabla\gamma_i$ for any $i\in\mathcal{N}$.

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 $\mathcal{O}(\kappa^2(A)/\epsilon)$ matrix-vector multiplications, where $A\in\mathbb{R}^{m\times n}$ such that $\mathbf{rank}(A) = m$, 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 $Ex - q \in \mathcal{K}$, where $\mathcal{K}$ 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 $\underline{\gamma}\in\mathbb{R}$.*

(b) *Let $\mathcal{R}$ be the set of convex functions $\rho : \mathbb{R}^n \to \mathbb{R}$ such that subdifferential of $\rho$ is uniformly bounded on $\mathbb{R}^n$, i.e., there exists $B > 0$ such that $\|q\|_2 \leq B$ for all $q \in \partial\rho(x)$, $x \in \mathbb{R}^n$; and $\tau\|x\|_2 \leq \rho(x)$ for all $x \in \mathbb{R}^n$ for some $\tau > 0$.*

**Assumption 1.** *For all $i \in \mathcal{N}$, we assume that $\gamma_i \in \Gamma$ and $\rho_i \in \mathcal{R}$ with corresponding constants $L_{\gamma_i}$, $\underline{\gamma_i}$, $B_i$ and $\tau_i$.*

Most of the important regularizers and loss functions used in machine learning and statistics literature lie in $\mathcal{R}$ and $\Gamma$, 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 $\mathcal{R}$. Given $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, quadratic-loss $\|Ax - b\|_2^2$, Huber-loss $\sum_{i=1}^m h(a_i^\mathsf{T} x - b_i)$ (see Section 3), logistic-loss $\sum_{i=1}^m \log\left(1 + e^{-b_i a_i^\mathsf{T} x}\right)$, or fair-loss (Blatt et al., 2007) functions all belong to $\Gamma$.

Throughout the paper, we adopt the notation $\mathbf{x} = (x_i; \mathbf{x}_{-i})$ with $\mathbf{x}_{-i} = (x_j)_{j\neq i}$ to denote a vector where $x_i$ and $\mathbf{x}_{-i}$ are treated as variable and parameter sub-vectors of $\mathbf{x}$, respectively. Given $f : \mathbb{R}^{nN} \to \mathbb{R}$, $\nabla_{x_i} f(\mathbf{x}) \in \mathbb{R}^n$ denotes the sub-vector of $\nabla f(\mathbf{x}) \in \mathbb{R}^{nN}$ 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 $\{\rho_i\}_{i\in\mathcal{N}}$ and $\{\gamma_i\}_{i\in\mathcal{N}}$ 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 $\mathbf{prox}_{\rho/L_\gamma}$ can be computed efficiently, the accelerated proximal gradient (APG) algorithm proposed in (Beck & Teboulle, 2009; Tseng, 2008) guarantees that

$$0 \leq F(x^{(\ell)}) - F^* \leq \frac{2L_\gamma}{(\ell+1)^2}\|x^{(0)} - x^*\|_2^2, \tag{5}$$

where $x^{(0)}$ is the initial iterate and $x^* \in \operatorname{argmin}_{x\in\mathbb{R}^n} F(x)$ –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 $\mathcal{O}(\sqrt{L_\gamma}\epsilon^{-\frac{1}{2}})$ iterations.

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

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

$$\bar{F}^* := \min_{\mathbf{x}\in\mathbb{R}^{nN}} \left\{\bar{F}(\mathbf{x}) := \bar{\rho}(\mathbf{x}) + \bar{\gamma}(\mathbf{x}) \text{ s.t. } A\mathbf{x} = b\right\}, \tag{6}$$

where $\bar{\rho}(\mathbf{x}) := \sum_{i=1}^N \rho_i(x_i)$, $\bar{\gamma}(\mathbf{x}) := \sum_{i=1}^N \gamma_i(x_i)$, and $A \in \mathbb{R}^{m\times nN}$ has $\mathbf{rank}(A) = m$, 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 $\mathcal{G}$[1], 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 $\{A_i\}_{i\in\mathcal{N}} \subset \mathbb{R}^{m\times n}$ such that $A = [A_1, A_2, \ldots, A_N]$; $\bar{L} := \max_{i\in\mathcal{N}} L_{\gamma_i}$, $\bar{\tau} := \min_{i\in\mathcal{N}} \tau_i$.

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

$$\mathbf{x}_*^{(k)} \in \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^{nN}} P^{(k)}(\mathbf{x}) := \lambda^{(k)}\bar{\rho}(\mathbf{x}) + f^{(k)}(\mathbf{x}), \tag{7}$$

$$f^{(k)}(\mathbf{x}) := \lambda^{(k)}\bar{\gamma}(\mathbf{x}) + \tfrac{1}{2}\|A\mathbf{x} - b - \lambda^{(k)}\theta^{(k)}\|_2^2, \tag{8}$$

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 $\{\mathbf{x}^{(k)}\}$ is constructed such that every $\mathbf{x}^{(k)}$ satisfies one of the following conditions:

$$\begin{aligned} &(a) \quad P^{(k)}(\mathbf{x}^{(k)}) - P^{(k)}(\mathbf{x}_*^{(k)}) \leq \alpha^{(k)}, \\ &(b) \quad \exists g_i^{(k)} \in \partial_{x_i} P^{(k)}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^{(k)}} \\ &\qquad \text{s.t. } \max_{i\in\mathcal{N}} \|g_i^{(k)}\|_2 \leq \tfrac{\xi^{(k)}}{\sqrt{N}}, \end{aligned} \tag{9}$$

$\partial_{x_i} P^{(k)}(\mathbf{x})|_{\mathbf{x}=\bar{\mathbf{x}}} := \lambda^{(k)}\partial\rho_i(x_i)|_{x_i=\bar{x}_i} + \nabla_{x_i} f^{(k)}(\bar{\mathbf{x}})$. Note that $\nabla f^{(k)}(\mathbf{x})$ is Lipschitz continuous in $\mathbf{x} \in \mathbb{R}^{nN}$ with constant $\lambda^{(k)}\bar{L} + \sigma_{\max}^2(A)$. Given $\{\mathbf{x}^{(0)}, \lambda^{(0)}, \alpha^{(0)}, \xi^{(0)}\}$ and $c \in (0, 1)$, we choose the sequence $\{\lambda^{(k)}, \alpha^{(k)}, \xi^{(k)}, \theta^{(k)}\}$ as shown in Fig. 1.

```
Algorithm DFAL $\left(\lambda^{(1)}, \alpha^{(1)}, \xi^{(1)}\right)$
Step 0: Set $\theta^{(1)} = \mathbf{0}$, $k = 1$
Step $k$: ($k \geq 1$)
  1. Compute $\mathbf{x}^{(k)}$ such that (9)(a) or (9)(b) holds
  2. $\theta^{(k+1)} = \theta^{(k)} - \frac{A\mathbf{x}^{(k)} - b}{\lambda^{(k)}}$
  3. $\lambda^{(k+1)} = c\lambda^{(k)}$,   $\alpha^{(k+1)} = c^2\,\alpha^{(k)}$,   $\xi^{(k+1)} = c^2\,\xi^{(k)}$
```

*Figure 1.* First-order Augmented Lagrangian algorithm

In Section 2.2.1, we show that **DFAL** can compute an $\epsilon$-optimal and $\epsilon$-feasible $\mathbf{x}_\epsilon$ to (6), i.e., $\|A\mathbf{x}_\epsilon - b\|_2 \leq \epsilon$ and $|\bar{F}(\mathbf{x}_\epsilon) - F^*| \leq \epsilon$, in at most $\mathcal{O}(\log(1/\epsilon))$ iterations.

[1] $\mathcal{G}$ can contain cycles.

Next, in Section 2.2.2, we show that computing an $\epsilon$-optimal, $\epsilon$-feasible solution $x_\epsilon$ requires at most $\mathcal{O}\left(\frac{\sigma^3_{\max}(A)}{\min_{i\in\mathcal{N}}\sigma^2_{\min}(A_i)}\epsilon^{-1}\right)$ floating point operations. Using this result, in Section 2.2.3 we establish that **DFAL** can compute $x_\epsilon$ in a distributed manner within $\mathcal{O}(\epsilon^{-1})$ 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* computation setting.

### 2.2.1. DFAL ITERATION COMPLEXITY

We first show that $\{\mathbf{x}^{(k)}\}$ 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 $\bar\rho:\mathbb{R}^{nN}\to\mathbb{R}$ be defined as $\bar\rho(\mathbf{x})=\sum_{i\in\mathcal{N}}\rho_i(x_i)$, 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 $f:\mathbb{R}^{nN}\to\mathbb{R}$ denote a convex function such that there exist constants $\{L_i\}_{i=1}^N\subset\mathbb{R}_{++}$ that satisfy*

$$f(\mathbf{y})\le f(\bar{\mathbf{y}})+\nabla f(\bar{\mathbf{y}})^\mathsf{T}(\mathbf{y}-\bar{\mathbf{y}})+\sum_{i=1}^N\frac{L_i\|y_i-\bar y_i\|_2^2}{2}$$

*for all $\mathbf{y},\bar{\mathbf{y}}\in\mathbb{R}^{nN}$. Given $\alpha,\lambda\ge0$, and $\bar{\mathbf{x}}\in\mathbb{R}^{nN}$ such that $\lambda\rho(\bar{\mathbf{x}})+f(\bar{\mathbf{x}})-\min_{\mathbf{x}\in\mathbb{R}^{nN}}\{\lambda\rho(\mathbf{x})+f(\mathbf{x})\}\le\alpha$, it follows that $\|\nabla_{x_i}f(\bar{\mathbf{x}})\|_2\le\sqrt{2L_i\alpha}+\lambda B_i$ 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 $L_i^{(k)}:=\lambda^{(k)}L_{\gamma_i}+\sigma^2_{\max}(A)$ for all $i\in\mathcal{N}$.*

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

**Lemma 3.** *Let $\{\mathbf{x}^{(k)}\}$ be the **DFAL** iterate sequence, i.e., at least one of the conditions in (9) hold for all $k\ge1$. Define $\Theta_i^{(k)}:=\max\left\{\sqrt{2L_i^{(k)}\frac{\alpha^{(k)}}{(\lambda^{(k)})^2}},\ \frac{1}{\sqrt N}\frac{\xi^{(k)}}{\lambda^{(k)}}\right\}+B_i+\|\nabla\gamma_i(x_i^{(k)})\|_2$. Then for all $k\ge1$, we have*

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

Theorem 1 establishes that the **DFAL** iterate sequence $\{\mathbf{x}^{(k)}\}$ is bounded whenever $\{\rho_i,\gamma_i\}_{i\in\mathcal{N}}$ 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 $\max\{\|\mathbf{x}_*^{(k)}\|_2,\|\mathbf{x}^{(k)}\|_2\}\le B_x$ and $\|\theta^{(k)}\|_2\le B_\theta$ for all $k\ge1$, whenever $\lambda^{(1)}$ and $\xi^{(1)}$ are chosen such that $0<\lambda^{(1)}\le\bar\lambda$ and $\frac{\xi^{(1)}}{\lambda^{(1)}}<\bar\tau$.*

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 $\xi^{(1)}$ are chosen according to Theorem 1. Then the primal-dual iterate sequence $\{\mathbf{x}^{(k)},\theta^{(k)}\}$ generated by **DFAL** satisfy*

*(a)* $\|A\mathbf{x}^{(k)}-b\|_2\le2B_\theta\lambda^{(k)}$,

*(b)* $\bar F(\mathbf{x}^{(k)})-\bar F^*\ge-\lambda^{(k)}\frac{(\|\theta^*\|_2+B_\theta)^2}{2}$

*(c)* $\bar F(\mathbf{x}^{(k)})-\bar F^*\le\lambda^{(k)}\left(\frac{B_\theta^2}{2}+\frac{\max\{\alpha^{(1)},\,\xi^{(1)}B_x\}}{\left(\lambda^{(1)}\right)^2}\right)$,

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

**Corollary 1.** *The **DFAL** iterates $\mathbf{x}^{(k)}$ are $\epsilon$-feasible, i.e., $\|A\mathbf{x}^{(k)}-b\|_2\le\epsilon$, and $\epsilon$-optimal, i.e., $|\bar F(\mathbf{x}^{(k)})-\bar F^*|\le\epsilon$, for all $k\ge N(\epsilon)$ and $N(\epsilon)=\log_{\frac{1}{c}}(\frac{\bar C}{\epsilon})$ for some $\bar C>0$.*

### 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 $\mathbf{x}^{(k)}$ satisfying (9) within $\mathcal{O}(1/\lambda^{(k)})$ gradient and prox computations. This result together with Theorem 2 guarantees that for any $\epsilon>0$, **DFAL** can compute an $\epsilon$-optimal and $\epsilon$-feasible iterate within $\mathcal{O}\left(\epsilon^{-1}\right)$ floating point operations. Following lemma gives the iteration complexity of the oracle **MS-APG** displayed in Fig. 2.

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

$$0\le\Phi(\mathbf{y}^{(\ell)})-\min_{\mathbf{y}\in\mathbb{R}^{nN}}\Phi(\mathbf{y})\le\frac{\sum_{i=1}^N2L_i\|y_i^{(0)}-y_i^*\|_2^2}{(\ell+1)^2}. \tag{10}$$

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

```
Algorithm MS-APG ( $\bar\rho, f, \mathbf{y}^{(0)}$ )
Step 0: Take $\bar{\mathbf{y}}^{(1)}=\mathbf{y}^{(0)}, t^{(1)}=1$
Step $\ell$: ($\ell\ge1$)
  1. $\mathbf{y}_i^{(\ell)}=\mathbf{prox}_{\rho_i/L_i}\left(\bar{\mathbf{y}}_i^{(\ell)}-\nabla_{y_i}f(\bar{\mathbf{y}}^{(\ell)})/L_i\right)\quad\forall i\in\mathcal{N}$
  2. $t^{(\ell+1)}=(1+\sqrt{1+4\,(t^{(\ell)})^2})/2$
  3. $\bar{\mathbf{y}}^{(\ell+1)}=\mathbf{y}^{(\ell)}+\frac{t^{(\ell)}-1}{t^{(\ell+1)}}\left(\mathbf{y}^{(\ell)}-\mathbf{y}^{(\ell-1)}\right)$
```

*Figure 2.* Multi Step - Accelerated Prox. Gradient (MS-APG) alg.

Consider the problem $\Phi^* = \min \Phi(\mathbf{y}) := \bar{\rho}(\mathbf{y}) + f(\mathbf{y})$ 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 $\frac{\max_{i\in\mathcal{N}} L_i}{\min_{i\in\mathcal{N}} L_i} \gg 1$, 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 follows.

**Lemma 5.** *The iterate sequence $\{\mathbf{y}^{(\ell)}\}_{\ell\in\mathbb{Z}_+}$ generated when we call **MS-APG**$\big(\lambda^{(k)}\bar{\rho}, f^{(k)}, \mathbf{x}^{(k-1)}\big)$ satisfies $P^{(k)}(\mathbf{y}^{(\ell)}) - P^{(k)}(\mathbf{x}_*^{(k)}) \leq \alpha^{(k)}$, for all $\ell \geq \sqrt{\frac{\sum_{i=1}^N 2L_i^{(k)}\|x_i^{(k-1)}-x_{*i}^{(k)}\|_2^2}{\alpha^{(k)}}} - 1$, 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 $\mathbf{x}^{(k)}$ satisfying (9) within $\mathcal{O}(1/\lambda^{(k)})$ **MS-APG** iterations.*

Theorem 2 and Lemma 5 together imply that **DFAL** can compute an $\epsilon$-feasible, and $\epsilon$-optimal solution to (6) within $\mathcal{O}(1/\epsilon)$ **MS-APG** iterations. Due to space considerations, we will only state and prove this result for the case where $\nabla\bar{\gamma}$ is bounded in $\mathbb{R}^{nN}$ since the bounds $B_\theta$ 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 $\|\nabla\gamma_i(x)\|_2 \leq G_i$ for all $x \in \mathbb{R}^n$ and for all $i \in \mathcal{N}$. Let $N^o_{\textbf{DFAL}}(\epsilon)$ and $N^f_{\textbf{DFAL}}(\epsilon)$ denote the number of **DFAL**-iterations to compute an $\epsilon$-optimal, and an $\epsilon$-feasible solutions to (6), respectively. Let $N^{(k)}$ denote **MS-APG** iteration number required to compute $\mathbf{x}^{(k)}$ satisfying at least one of the conditions in (9). Then*

$$\sum_{k=1}^{N^o_{\textbf{DFAL}}(\epsilon)} N^{(k)} = \mathcal{O}\left(\Theta^2\sigma_{\max}(A)\epsilon^{-1}\right),$$

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

*where $\Theta = \frac{\sigma_{\max}(A)}{\min_{i\in\mathcal{N}}\sigma_{\min}(A_i)}$.*

### 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 $C\mathbf{x} = 0$, where $C \in \mathbb{R}^{n|\mathcal{E}|\times nN}$ is a block matrix such that the block $C_{(i,j),l} \in \mathbb{R}^{n\times n}$ 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 $\mathbf{0}_n$ otherwise, where $I_n$ and $\mathbf{0}_n$ denote $n \times n$ identity and zero matrices, respectively. Let $\Omega \in \mathbb{R}^{N\times N}$ be the Laplacian of $\mathcal{G}$, i.e., for all $i \in \mathcal{N}$, $\Omega_{ii} = d_i$, 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 $i \in \mathcal{N}$. Then it follows that

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

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, $\mathbf{rank}(\Omega) = N - 1$, 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, $\mathbf{rank}(C) = n(N-1)$.

Let $C = U\Sigma V^\mathsf{T}$ denote the reduced singular value decomposition (SVD) of $C$, where $U \in \mathbb{R}^{n|\mathcal{E}|\times n(N-1)}$, $\Sigma = \operatorname{diag}(\sigma)$, $\sigma \in \mathbb{R}^{n(N-1)}_{++}$, and $V \in \mathbb{R}^{nN\times n(N-1)}$. Note that $\sigma^2_{\max}(C) = \psi_{\max}$, and $\sigma^2_{\min}(C) = \psi_{N-1}$. Define $A := \Sigma V^\mathsf{T}$. $A \in \mathbb{R}^{n(N-1)\times nN}$ has linearly independent rows; more importantly, $A^\mathsf{T} A = C^\mathsf{T} C = \Psi$; hence, $\sigma^2_{\max}(A) = \psi_{\max}$, and $\sigma^2_{\min}(A) = \psi_{N-1}$. We also have $\{\mathbf{x} \in \mathbb{R}^{nN} : A\mathbf{x} = 0\} = \{\mathbf{x} \in \mathbb{R}^{nN} : C\mathbf{x} = 0\}$. Hence, the general problem in (6) with $A := \Sigma V^\mathsf{T}$ and $b = \mathbf{0} \in \mathbb{R}^{n(N-1)}$ is equivalent to (4). Let $A_i \in \mathbb{R}^{n(N-1)\times n}$ and $C_i \in \mathbb{R}^{n|\mathcal{E}|\times n}$ 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 $A = \Sigma V^\mathsf{T}$ 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 $\theta^{(1)} = \mathbf{0}$, Step 2 in **DFAL** and (8) imply that $\theta^{(k+1)} = -\sum_{t=1}^k \frac{Ax^{(t)}}{\lambda^{(t)}}$, and $\nabla f^{(k)}(\mathbf{x}) = \lambda^{(k)}\nabla\bar{\gamma}(\mathbf{x}) + A^\mathsf{T}(A\mathbf{x} - \lambda^{(k)}\theta^{(k)}) = \lambda^{(k)}\nabla\bar{\gamma}(\mathbf{x}) + \Psi\left(\mathbf{x} + \lambda^{(k)}\sum_{t=1}^{k-1}\frac{1}{\lambda^{(t)}}\mathbf{x}^{(t)}\right)$. Moreover, from the definition of $\Psi$, it follows that

$$\nabla_{x_i} f^{(k)}(\mathbf{x}) = \lambda^{(k)}\nabla\gamma_i(x_i) + d_i\left(x_i + \bar{x}_i^{(k)}\right) - \sum_{j\in\mathcal{O}_i}\left(x_j + \bar{x}_j^{(k)}\right),$$

where $\bar{\mathbf{x}}^{(k)} := \sum_{t=1}^{k-1}\frac{\lambda^{(k)}}{\lambda^{(t)}}\mathbf{x}^{(t)}$, and $\mathcal{O}_i$ 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 $\{x_j^{(t)}\}_{j\in\mathcal{O}_i}$ is transmitted to $i$ at the end of Step 1 of the previous **DFAL** iterations $1 \leq t \leq k-1$. Hence, during the $\ell$-th iteration of **MS-APG**$\big(\lambda^{(k)}\bar{\rho}, f^{(k)}, \mathbf{x}^{(k-1)}\big)$ call, each node $i \in \mathcal{N}$ can compute $\nabla_{y_i} f^{(k)}(\bar{\mathbf{y}}^{\ell})$ locally if $\{\bar{y}_j^{(t)}\}_{j\in\mathcal{O}_i}$ 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., $\exists g_i^{(k)} \in \partial\rho_i(x_i)|_{x_i=x_i^{(k)}} + \nabla_{x_i} f^{(k)}(\mathbf{x}^{(k)})$ for all $i\in\mathcal{N}$ such that $\max_{i\in\mathcal{N}} \|g_i^{(k)}\|_2 \leq \frac{\xi^{(k)}}{\sqrt{N}}$. 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 $\ell_{\max}^{(k)} := B_x\sqrt{\frac{2\sum_{i\in\mathcal{N}} L_i^{(k)}}{\alpha^{(k)}}}$ **MS-APG** iterations, then Lemma 5 implies that (9)(a) must be true. Hence, all the nodes move to next **DFAL** iteration after $\ell_{\max}^{(k)}$ many **MS-APG** updates. For implementable version of **DFAL**, see Figure 3, where $B_x$ is the bound in Theorem 1, and $\mathcal{N}_i := \mathcal{O}_i \cup \{i\}$.

**Algorithm DFAL** $(\mathbf{x}^{(0)}, \lambda^{(1)}, \alpha^{(1)}, \xi^{(1)}, B_x, \psi_{\max})$

```
1:  $k \leftarrow 1, \quad \bar{x}_i^{(1)} \leftarrow \mathbf{0}, \quad \forall i \in \mathcal{N}$
2:  while $k \geq 1$ do
3:    $\ell \leftarrow 1, \quad t^{(1)} \leftarrow 1, \quad \text{STOP} \leftarrow \textbf{false}$
4:    $y_i^{(0)} \leftarrow x_i^{(k-1)}, \quad \bar{y}_i^{(1)} \leftarrow x_i^{(k-1)}, \quad \forall i \in \mathcal{N}$
5:    $L_i^{(k)} \leftarrow \lambda^{(k)} L_{\gamma_i} + \psi_{\max}, \forall i \in \mathcal{N}$
6:    $\ell_{\max}^{(k)} \leftarrow B_x\sqrt{\frac{2\sum_{i\in\mathcal{N}} L_i^{(k)}}{\alpha_k}}$
7:    while STOP $=$ false do
8:      for $i \in N$ do
9:        $q_i^{(\ell)} \leftarrow \lambda^{(k)}\nabla\gamma_i\left(\bar{y}_i^{(\ell)}\right) + \sum_{j\in\mathcal{N}_i}\Omega_{ij}\left(\bar{y}_j^{(\ell)} + \bar{x}_j^{(k)}\right)$
10:       $y_i^{(\ell)} \leftarrow \mathbf{prox}_{\lambda^{(k)}\rho_i/L_i^{(k)}}\left(\bar{\mathbf{y}}_i^{(\ell)} - q_i^{(\ell)}/L_i^{(k)}\right)$
11:     end for
12:     if $\exists g_i \in q_i^{(\ell)} + \lambda^{(k)}\partial\rho_i\left(\bar{y}_i^{(\ell)}\right)$ s.t. $\max_{i\in\mathcal{N}} \|g_i\|_2 \leq \frac{\xi^{(k)}}{\sqrt{N}}$
        then
13:       STOP $\leftarrow \mathbf{true}, \quad x_i^{(k)} \leftarrow \bar{y}_i^{(\ell)}, \forall i \in \mathcal{N}$
14:     else if $\ell = \ell_{\max}^{(k)}$ then
15:       STOP $\leftarrow \mathbf{true}, \quad x_i^{(k)} \leftarrow y_i^{(\ell)}, \forall i \in \mathcal{N}$
16:     end if
17:     $t^{(\ell+1)} \leftarrow (1 + \sqrt{1 + 4\,(t^{(\ell)})^2})/2$
18:     $\bar{y}_i^{(\ell+1)} \leftarrow y_i^{(\ell)} + \frac{t^{(\ell)}-1}{t^{(\ell+1)}}\left(y_i^{(\ell)} - y_i^{(\ell-1)}\right), \forall i \in \mathcal{N}$
19:     $\ell \leftarrow \ell + 1$
20:   end while
21:   $\lambda^{(k+1)} \leftarrow c\lambda^{(k)}, \ \alpha^{(k+1)} \leftarrow c^2\alpha^{(k)}, \ \xi^{(k+1)} \leftarrow c^2\xi^{(k)}$
22:   $\bar{x}_i^{(k+1)} \leftarrow \frac{\lambda^{(k+1)}}{\lambda^{(k)}}\left(\bar{x}_i^{(k)} + x_i^{(k)}\right), \quad \forall i \in \mathcal{N}$
23:   $k \leftarrow k + 1$
24: end while
```

*Figure 3.* Dist. First-order Aug. Lagrangian (DFAL) alg.

### 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 $\mathcal{O}\left(\frac{1}{\epsilon^2}\log\left(\frac{1}{p}\right)\right)$ **RBCD** iterations. In Section 4.9 of the supplementary file, we discuss how to improve this rate to $\mathcal{O}\left(\frac{1}{\epsilon}\log\left(\frac{1}{p}\right)\right)$ using an *accelerated* **RBCD**.

```
Algorithm RBCD ( $\bar{\rho}, f, \mathbf{y}^{(0)}$ )
Step $\ell$: ($\ell \geq 0$)
  1. $i \in \mathcal{N}$ is realized with probability $\frac{1}{N}$
  2. $\mathbf{y}_i^{(\ell+1)} = \mathbf{prox}_{\rho_i/L_i}\left(y_i^{(\ell)} - \nabla_{y_i} f(\mathbf{y}^{(\ell)})/L_i\right)$
  3. $\mathbf{y}_{-i}^{(\ell+1)} = \mathbf{y}_{-i}^{(\ell)}$
```

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

Nesterov (2012) proposed an RBCD method for solving $\min_{\mathbf{y}\in\mathbb{R}^{nN}} f(\mathbf{y})$, where $f$ is convex with *block Lipschitz continuous gradient*, i.e., $\nabla_{y_i} f(y_i;\mathbf{y}_{-i})$ is Lipschitz continuous in $y_i$ with constant $L_i$ for all $i$. Later, Richtárik & Takáč (2012) extended the convergence rate results to $\min_{\mathbf{y}\in\mathbb{R}^{nN}} \Phi(\mathbf{y}) := \sum_{i=1}^N \rho_i(y_i) + f(\mathbf{y})$, such that $\mathbf{prox}_{t\rho_i}$ can be computed efficiently for all $t > 0$ 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 $\{\mathbf{y}^{(\ell)}\}$ computed by **RBCD** displayed in Fig. 4 satisfies

$$\mathbb{P}(\Phi(\mathbf{y}^{(\ell)}) - \Phi^* \leq \alpha) \geq 1 - p, \qquad (11)$$

where $C := \max\{\mathcal{R}_L^2(\mathbf{y}^{(0)}), \Phi(\mathbf{y}^{(0)}) - \Phi^*\}$, $\mathcal{R}_L^2(\mathbf{y}^{(0)}) := \max_{\mathbf{y},\mathbf{y}^*}\left\{\sum_{i=1}^N L_i\|y_i - y_i^*\|_2^2 : \ \Phi(\mathbf{y}) \leq \Phi(\mathbf{y}^{(0)}), \ \mathbf{y}^* \in \mathcal{Y}^*\right\}$, 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 $\nabla_{y_i} f(\mathbf{y})$ can be computed more efficiently as compared to the full gradient $\nabla f(\mathbf{y})$. The **RBCD** algorithm can be implemented for the distributed minimization problem when the nodes in $\mathcal{G}$ work *asynchronously*. Assume that for any $\mathbf{y} = (y_i)_{i\in\mathcal{N}} \in \mathbb{R}^{nN}$, each node $i$ is *equally likely* to be the first to complete computing $\mathbf{prox}_{\rho_i/L_i}(y_i - \nabla_{y_i} f(\mathbf{y})/L_i)$, 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

$$\mathbb{P}\left(P^{(k)}\big(\mathbf{x}^{(k)}\big) - P^{(k)}\big(\mathbf{x}_*^{(k)}\big) \leq \alpha^{(k)}\right) \geq (1-p)^{\frac{1}{N(\epsilon)}}, \qquad (12)$$

where $N(\epsilon) = \log_{\frac{1}{c}}\left(\frac{\bar{C}}{\epsilon}\right)$ defined in Corollary 1. Since $\left(1-p\right)^{\frac{1}{N(\epsilon)}} \leq 1 - \frac{p}{N(\epsilon)}$ for $p \in (0,1)$, the total number of **RBCD** iterarions for the $k$-th subproblem is bounded: $N^{(k)} \leq \mathcal{O}\big(\frac{1}{\alpha^{(k)}} \log\big(\frac{N(\epsilon)}{p}\big)\big) = \mathcal{O}\big(\frac{1}{\alpha^{(k)}}\big(\log\big(\frac{1}{p}\big) + \log\log\big(\frac{1}{\epsilon}\big)\big)\big)$. 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 $\mathcal{O}\left(\frac{1}{\epsilon^2}\log\left(\frac{1}{p}\right)\right)$ **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:

$$\min_{x\in\mathbb{R}^n} \sum_{i=1}^{N} \beta_1\|x\|_1 + \beta_2\|x\|_{G_i} + \sum_{j=1}^{m_i} h_\delta\left(\mathbf{a}_i^\mathsf{T}(j)x - b_i\right), \quad (13)$$

where $\beta_1, \beta_2 > 0$, $A_i \in \mathbb{R}^{m_i \times n}$, $b_i \in \mathbb{R}^{m_i}$, $\mathbf{a}_i^\mathsf{T}(j)$ 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 $\|x\|_{G_i} := \sum_{k=1}^{K}\|x_{g_i(k)}\|_2$ denotes the group norm with respect to the partition $G_i$ of $[1,n] := \{1,\cdots,n\}$ 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 $g_i(j) \cap g_i(k) = \emptyset$ for all $j \neq k$. In this case, $\gamma_i(x) := \sum_{j=1}^{m_i} h_\delta\left(\mathbf{a}_i^\mathsf{T}(j)x - b_i\right)$ 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 efficent variant, **SADMM**, for (13).

```
Algorithm SADMM ( $c, \mathbf{x}^{(0)}$ )
Initialization: $\mathbf{y}^{(0)} = \mathbf{x}^{(0)}$, $p_i^{(k)} = \tilde{p}_i^{(k)} = \mathbf{0}$,  $i \in \mathcal{N}$
Step $\ell$: ($\ell \geq 0$) For $i \in \mathcal{N}$ compute
  1. $x_i^{(k+1)} = \mathbf{prox}_{\frac{1}{c(d_i^2+d_i+1)}\rho_i}\left(\tilde{x}_i^{(k)}\right)$
  2. $y_i^{(k+1)} = \mathbf{prox}_{\frac{1}{c(d_i^2+d_i+1)}\gamma_i}\left(\tilde{y}_i^{(k)}\right)$
  3. $s_i^{(k+1)} = \sum_{j\in\mathcal{N}_i} \Omega_{ij} x_j^{(k+1)}/(d_i+1)$
  4. $p_i^{(k+1)} = p_i^{(k)} + s_i^{(k+1)}$
  5. $\tilde{s}_i^{(k+1)} = \sum_{j\in\mathcal{N}_i} \Omega_{ij} y_j^{(k+1)}/(d_i+1)$
  6. $\tilde{p}_i^{(k+1)} = \tilde{p}_i^{(k)} + \tilde{s}_i^{(k+1)}$
```

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

$$\begin{aligned} \min_{x_i\in\mathbb{R}^n, z_i\in\mathcal{Z}_i} \quad & \textstyle\sum_{i=1}^{N} F_i(x_i) := \rho_i(x_i) + \gamma_i(x_i) \\ \text{s.t.} \quad & \Omega_{ij}x_j = z_{ij}, \quad i \in \mathcal{N},\ j \in \mathcal{N}_i. \end{aligned} \quad (14)$$

Only (14) is penalized when forming the augmented Lagrangian, which is alternatingly minimized in $\mathbf{x} \in \mathbb{R}^{nN}$, $\mathbf{z}^\mathsf{T} = [z_1^\mathsf{T}, \ldots, z_N^\mathsf{T}]$, where $\mathcal{Z}_i \ni z_i = [z_{ij}]_{j\in\mathcal{N}_i} \in \mathbb{R}^{d_i+1}$. Makhdoumi & Ozdaglar (2014) establish that suboptimality and consensus violation converge to 0 with a rate $\mathcal{O}(1/k)$, 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 $\mathbf{prox}_{F_i}$ for each $i \in \mathcal{N}$ is the computational bottleneck in each iteration of **ADMM**. Note that computing $\mathbf{prox}_{F_i}$ for (13) is almost as *hard* as solving the problem. To deal with this issue, we considered the following reformulation:

$$\begin{aligned} \min_{\substack{x_i, y_i \in\mathbb{R}^n,\\ z_i, \tilde{z}_i \in \mathcal{Z}_i}} \quad & \textstyle\sum_{i\in\mathcal{N}} \rho_i(x_i) + \gamma_i(y_i) \\ \text{s.t.} \quad & \Omega_{ij}x_j = z_{ij}, \quad i \in \mathcal{N},\ j \in \mathcal{N}_i \\ & \Omega_{ij}y_j = \tilde{z}_{ij}, \quad i \in \mathcal{N},\ j \in \mathcal{N}_i \\ & x_i = q_i,\ y_i = q_i, \quad i \in \mathcal{N}. \end{aligned}$$

ADMM algorithm for this formulation is displayed in Fig. 5, where $c > 0$ denotes the penalty parameter. Steps of **SADMM** can be derived by minimizing the augmented Lagrangian alternatingly in $(\mathbf{x}, \mathbf{y})$, and in $(\mathbf{z}, \tilde{\mathbf{z}}, \mathbf{q})$ while fixing the other. As in (Makhdoumi & Ozdaglar, 2014), computing $(\mathbf{z}, \tilde{\mathbf{z}}, \mathbf{q})$ can be avoided by exploiting the structure of optimality conditions. Prox centers in **SADMM** are

$$\begin{aligned} \tilde{x}_i^{(k)} &= x_i^{(k)} - \tfrac{\sum_{j\in\mathcal{N}_i}\Omega_{ji}(s_j^{(k)}+p_j^{(k)})+r_i^{(k)}+(x_i^{(k)}-y_i^{(k)})/2}{d_i^2+d_i+1}, \\ \tilde{y}_i^{(k)} &= y_i^{(k)} - \tfrac{\sum_{j\in\mathcal{N}_i}\Omega_{ji}(\tilde{s}_j^{(k)}+\tilde{p}_j^{(k)})-r_i^{(k)}-(x_i^{(k)}-y_i^{(k)})/2}{d_i^2+d_i+1}, \end{aligned}$$

respectively; and $r_i^{(k+1)} = r_i^{(k)} + (x_i^{(k+1)} - y_i^{(k+1)})/2$.

### 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 $x \in \mathbb{R}$, define $\mathrm{sgn}(x)$ as -1, 0 and 1 when $x < 0$, $x = 0$, and $x > 0$, respectively; and for $x \in \mathbb{R}^n$, define $\mathrm{sgn}(x) = [\mathrm{sgn}(x_1), \mathrm{sgn}(x_2), \ldots, \mathrm{sgn}(x_n)]^\mathsf{T}$.

**Lemma 6.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function, $G = \{g(k)\}_{k=1}^{K}$ be a partition of $[1,n]$. For $\beta_1, \beta_2 > 0$, define $P = \lambda\rho + f$, where $\rho(x) := \beta_1\|x\|_1 + \beta_2\|x\|_G$. Then, for all $\bar{x} \in \mathbb{R}^n$ and $\xi > 0$, there exists $\nu \in \partial P(x)|_{x=\bar{x}}$ such that $\|\nu\|_2 \leq \xi$ if and only if $\|\pi^* + \omega^* + \nabla f(\bar{x})\|_2 \leq \xi$ for $\pi^*, \omega^*$ such that for each $k$, if $\bar{x}_{g(k)} \neq \mathbf{0}$, then $\pi^*_{g(k)} = \lambda\beta_1\,\mathrm{sgn}(\bar{x}_{g(k)}) + \big(\mathbf{1} - \mathrm{sgn}(|\bar{x}_{g(k)}|)\big) \odot \eta_{g(k)}$, and $\omega^*_{g(k)} = \lambda\beta_2\,\frac{\bar{x}_{g(k)}}{\|\bar{x}_{g(k)}\|_2}$; otherwise, if $\bar{x}_{g(k)} = \mathbf{0}$, then $\pi^*_{g(k)} = \eta_{g(k)}$, and $\omega^*_{g(k)}$ equals*

$$-\left(\pi^*_{g(k)} + \nabla_{x_{g(k)}} f(\bar{x})\right)\min\left\{1,\ \frac{\lambda\beta_2}{\|\pi^*_{g(k)} + \nabla_{x_{g(k)}} f(\bar{x})\|_2}\right\},$$

*where $\odot$ denotes componentwise multiplication, and $\eta_{g(k)} = -\,\mathrm{sgn}\left(\nabla_{x_{g(k)}} f(\bar{x})\right) \odot \min\left\{|\nabla_{x_{g(k)}} f(\bar{x})|,\ \lambda\beta_1\right\}$.*

*Table 2.* Comparison of **DFAL**, **AFAL** (Asynchronous **DFAL**), **ADMM**, and **SADMM**. (Termination time T=1800 sec.)

| Size | Alg. | Rel. Suboptimality | | Consensus Violation (CV) | | CPU Time (sec.) | | Iterations | |
|---|---|---|---|---|---|---|---|---|---|
| | | Case 1 | Case 2 | Case 1 | Case 2 | Case 1 | Case 2 | Case 1 | Case 2 |
| $n_g = 100$ $N = 5$ | SDPT3 (C) | 0 | 0 | 0 | 0 | 28 | 85 | 24 | 22 |
| | APG (C) | 1E-3 | N/A | 0 | N/A | 10 | N/A | 2173 | N/A |
| | DFAL (D) | 6E-4, 7E-4 | 4E-4, 6E-4 | 5E-5, 2E-5 | 5E-5, 3E-5 | **5, 5** | **5, 5** | 1103, 1022 | 1105, 1108 |
| | AFAL (D) | 3E-4, 8E-4 | 3E-4, 6E-4 | 3E-6, 5E-6 | 2E-6, 5E-6 | **16, 11** | **17, 11** | 9232, 9083 | 9676, 9844 |
| | ADMM (D) | 6E-5, 5E-5 | 7E-5, 7E-5 | 1E-4, 1E-4 | 1E-4, 1E-4 | 1125, 808 | 1090, 771 | 353, 253 | 363, 261 |
| | SADMM (D) | 1E-4, 3E-4 | 1E-4, 3E-4 | 1E-4, 1E-4 | 1E-4, 1E-4 | 771, 784 | 772, 804 | 592, 606 | 593, 623 |
| $n_g = 100$ $N = 10$ | SDPT3 (C) | 0 | 0 | 0 | 0 | 28 | 89 | 24 | 22 |
| | APG (C) | 1E-3 | N/A | 0 | N/A | 10 | N/A | 2173 | N/A |
| | DFAL (D) | 4E-4, 7E-4 | 4E-4, 3E-4 | 6E-5, 1E-5 | 6E-5, 2E-5 | **14, 12** | **14, 13** | 1794, 1439 | 1812, 1560 |
| | AFAL (D) | 4E-4, 8E-4 | 2E-4, 8E-4 | 7E-6, 2E-6 | 8E-6, 2E-6 | **48, 65** | **49, 62** | 20711, 41125 | 21519, 41494 |
| | ADMM (D) | 9E-3, 2E-2 | 8E-3, 2E-2 | 9E-4, 5E-4 | 8E-4, 4E-4 | T, T | T, T | 354, 353 | 373, 372 |
| | SADMM (D) | 3E-4, 9E-3 | 4E-4, 1E-2 | 2E-4, 1E-3 | 2E-4, 1E-3 | T, T | T, T | 867, 883 | 865, 879 |
| $n_g = 300$ $N = 5$ | SDPT3 (C) | 0 | 0 | 0 | 0 | 806 | 1653 | 26 | 29 |
| | APG (C) | 1E-3 | N/A | 0 | N/A | 253 | N/A | 8663 | N/A |
| | DFAL (D) | 2E-4, 5E-4 | 2E-4, 4E-4 | 6E-5, 4E-5 | 5E-5, 5E-5 | **77, 64** | **80, 65** | 1818, 1511 | 1897, 1535 |
| | AFAL (D) | 1E-4, 6E-4 | 2E-5, 6E-4 | 5E-7, 2E-5 | 6E-8, 2E-5 | **164, 99** | **273, 99** | 21747, 8760 | 37212, 8736 |
| | ADMM (D) | 5E-2, 1E-3 | 5E-2, 1E-3 | 5E-3, 1E-3 | 5E-3, 1E-3 | T, T | T, T | 109, 118 | 109, 118 |
| | SADMM (D) | 2E-2, 7E-2 | 2E-2, 8E-2 | 2E-3, 3E-3 | 2E-3, 3E-3 | T, T | T, T | 269, 274 | 268, 273 |
| $n_g = 300$ $N = 10$ | SDPT3 (C) | 0 | 0 | 0 | 0 | 806 | 1641 | 26 | 29 |
| | APG (C) | 1E-3 | N/A | 0 | N/A | 253 | N/A | 8663 | N/A |
| | DFAL (D) | 1E-4, 6E-4 | 6E-4, 1E-3 | 7E-5, 4E-5 | 9E-5, 5E-5 | **130, 80** | **122, 82** | 2942, 1721 | 2794, 1769 |
| | AFAL (D) | 2E-4, 7E-4 | 6E-4, 1E-3 | 8E-7, 1E-5 | 3E-7, 1E-5 | **350, 294** | **437, 288** | 48214, 29946 | 63110, 30371 |
| | ADMM (D) | 5E-2, 8E-2 | 5E-2, 8E-2 | 7E-3, 9E-3 | 7E-3, 9E-3 | T, T | T, T | 114, 124 | 113, 123 |
| | SADMM (D) | 3E-1, 3E+0 | 3E-1, 3E+0 | 4E-3, 2E-2 | 4E-3, 2E-2 | T, T | T, T | 255, 269 | 256, 268 |

Both **DFAL** and **SADMM** call for $\mathbf{prox}_{\rho_i}$. In Lemma 7, we show that it can be computed in closed form. On the other hand, when **ADMM**, and **SADMM** are implemented on (13), one needs to compute $\mathbf{prox}_{F_i}$ and $\mathbf{prox}_{\gamma_i}$, 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* $\bar{x} \in \mathbb{R}^n$, $x^p = \mathbf{prox}_{t\rho}(\bar{x})$ *is given by* $x^p_{g(k)} = \eta'_{g(k)} \max\left\{1 - \frac{t\beta_2}{\|\eta_{g(k)}\|_2}, 0\right\}$, *for* $1 \leq k \leq K$, *where* $\eta' = \mathrm{sgn}(\bar{x}) \odot \max\{|\bar{x}| - t\beta_1, 0\}$.

In our experiments, the network was either a *star tree* or a *clique* with either 5 or 10 nodes. The remaining problem parameters defining $\{\rho_i, \gamma_i\}_{i\in\mathcal{N}}$ were set as follows. We set $\beta_1 = \beta_2 = \frac{1}{N}$, $\delta = 1$, and $K = 10$. Let $n = Kn_g$ for $n_g \in \{100, 300\}$, i.e., $n \in \{1000, 3000\}$. We generated partitions $\{G_i\}_{i\in\mathcal{N}}$ in two different ways. For test problems in **CASE 1**, we created a single partition $G = \{g(k)\}_{k=1}^K$ by generating $K$ groups uniformly at random such that $|g(k)| = n_g$ for all $k$; and set $G_i = G$ for all $i \in \mathcal{N}$, i.e., $\rho_i(x) = \rho(x) := \beta_1\|x\|_1 + \beta_2\|x\|_G$ for all $i \in \mathcal{N}$. 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 $i \in \mathcal{N}$, $m_i = \frac{n}{2N}$, and the elements of $A_i \in \mathbb{R}^{m_i\times n}$ are i.i.d. with standard Gaussian, and we set $b_i = A_i\bar{x}$ for $\bar{x}_j = (-1)^j e^{-(j-1)/n_g}$ for $j \in [1, n]$.

We solved the distributed optimization problem (4) using **DFAL**, **AFAL** (*asynchronous* version of **DFAL** with *accelerated* **RBCD** -see the supplementary file for details), **ADMM**, and **SADMM** 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**, $\sum_{i\in\mathcal{N}} \rho_i(x) = \|x\|_1 + \|x\|_G$ and its prox mapping can be computed efficiently, while for **Case 2**, $\sum_{i\in\mathcal{N}} \rho_i(x)$ 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 then $10^{-3}$, and consensus violation, $\mathrm{CV}^{(k)}$, is less than $10^{-4}$, where $F^{(k)}$ equals to $\sum_{i\in\mathcal{N}} F_i(x_i^{(k)})$ for **DFAL** and **ADMM**, and to $\sum_{i\in\mathcal{N}} F_i\left(\frac{x_i^{(k)}+y_i^{(k)}}{2}\right)$ for **SADMM**; $\mathrm{CV}^{(k)}$ equals to $\max_{(ij)\in\mathcal{E}} \|x_i^{(k)} - x_j^{(k)}\|_2/\sqrt{n}$ for **DFAL**, and **ADMM**, and to $\max\{\max_{(ij)\in\mathcal{E}} \|x_i^{(k)} - x_j^{(k)}\|_2,\ \max_{i\in\mathcal{N}} \|x_i^{(k)} - y_i^{(k)}\|_2\}/\sqrt{n}$ for **SADMM**. 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 separated 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.

## References

Aybat, N. S. and Iyengar, G. A first-order augmented lagrangian method for compressed sensing. *SIAM Journal on Optimization*, 22(2):429–459, 2012.

Aybat, Necdet Serhat and Iyengar, Garud. An augmented lagrangian method for conic convex programming. *arXiv preprint arXiv:1302.6322*, 2013.

Aybat, Necdet Serhat and Iyengar, Garud. A unified approach for minimizing composite norms. *Mathematical Programming*, 144(1-2):181–226, 2014.

Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, 2(1):183–202, March 2009. ISSN 1936-4954.

Blatt, D., Hero, A., and Gauchman, H. A convergent incremental gradient method with a constant step size. *SIAM Journal on Optimization*, 18(1):29–51, 2007.

Chen, Annie I and Ozdaglar, Asuman. A fast distributed proximal-gradient method. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pp. 601–608. IEEE, 2012.

Duchi, J. C., Agarwal, A., and Wainwright, M. J. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Trans. Automat. Contr.*, 57(3):592–606, 2012.

Fercoq, Olivier and Richtárik, Peter. Accelerated, parallel and proximal coordinate descent. *arXiv preprint arXiv:1312.5799*, 2013.

Grone, R. and Merris, R. The laplacian spectrum of a graph. ii. *SIAM J. Discrete Math.*, 7(2):221–229, 1994.

Jakovetic, Dusan, Xavier, Joao, and Moura, J. Fast distributed gradient methods. 2011.

Lesser, Victor, Ortiz Jr, Charles L, and Tambe, Milind. *Distributed sensor networks: A multiagent perspective*, volume 9. Springer, 2003.

Makhdoumi, Ali and Ozdaglar, Asuman. Broadcast-based distributed alternating direction method of multipliers. In *Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on*, pp. 270–277. IEEE, 2014.

Mateos, Gonzalo, Bazerque, Juan Andrés, and Giannakis, Georgios B. Distributed sparse linear regression. *Signal Processing, IEEE Transactions on*, 58(10):5262–5276, 2010.

McDonald, Ryan, Hall, Keith, and Mann, Gideon. Distributed training strategies for the structured perceptron. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 456–464. Association for Computational Linguistics, 2010.

Necoara, Ion and Suykens, Johan AK. Application of a smoothing technique to decomposition in convex optimization. *Automatic Control, IEEE Transactions on*, 53 (11):2674–2679, 2008.

Nedic, Angelia and Ozdaglar, Asuman. Distributed subgradient methods for multi-agent optimization. *Automatic Control, IEEE Transactions on*, 54(1):48–61, 2009.

Nesterov, Y. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Qu, Zheng and Richtárik, Peter. Coordinate descent with arbitrary sampling i: Algorithms and complexity. *arXiv preprint arXiv:1412.8060*, 2014.

Richtárik, P. and Takáč, M. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *forthcoming, Mathematical Programming, Series A*, 2012. DOI: 10.1007/s10107-012-0614-z.

Tseng, Paul. On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM Journal on Optimization*, 2008.

Wei, Ermin and Ozdaglar, Asuman. Distributed alternating direction method of multipliers. 2012.

Wei, Ermin and Ozdaglar, Asuman. On the o (1/k) convergence of asynchronous distributed alternating direction method of multipliers. *arXiv preprint arXiv:1307.8254*, 2013.