A Distributed Proximal Method for Composite Convex Optimization

Garud Iyengar
Industrial Engineering and Operations Research
Columbia University

Joint work with Necdet Serhat Aybat and Zi Wang (Penn State)

NSF: DMS-1016571, CMMI-1235023, ONR: N000140310514
Distributed optimization

Compute the optimal solution for

\[
F^* := \min_{x \in \mathcal{X}} \sum_{i=1}^{N} F_i(x)
\]

Details

- \( \mathcal{N} = \{1, \ldots, N\} \) nodes of a connected undirected graph \( \mathcal{G} = (\mathcal{N}, \mathcal{E}) \)
  - Node \( i \) and \( j \) can communicate only if \( (i, j) \in \mathcal{E} \)

- Composite function \( F_i(x) = \rho_i(x) + \gamma_i(x) \) locally known at node \( i \)

- \( \mathcal{X} \) convex constraints of the form \( \{x : Ex - q \in \mathcal{K}\} \), \( \mathcal{K} \) convex cone.
Distributed optimization

Compute the optimal solution for

\[ F^* := \min_{x \in \mathcal{X}} \sum_{i=1}^{N} F_i(x) \]

Details

- \( \mathcal{N} = \{1, \ldots, N\} \) nodes of a connected undirected graph \( \mathcal{G} = (\mathcal{N}, \mathcal{E}) \)
  - Node \( i \) and \( j \) can communicate only if \( (i, j) \in \mathcal{E} \)

- Composite function \( F_i(x) = \rho_i(x) + \gamma_i(x) \) locally known at node \( i \)
  - \( \gamma_i \): convex + Lipschitz continuous gradient
  - \( \rho_i \): convex + possibly non-smooth but efficient prox-map \( \text{prox}_{t\rho}(x) \)

- For \( t > 0 \): \( \text{prox}_{t\rho}(x) = \arg\min_{y \in \mathbb{R}^n} \left\{ t\rho(y) + \frac{1}{2}\|y - x\|_2^2 \right\} \)

- \( \mathcal{X} \) convex constraints of the form \( \{x : Ex - q \in \mathcal{K}\} \), \( \mathcal{K} \) convex cone.
Canonical example

LASSO with locally known data

\[
\min_x \sum_{i=1}^{N} F_i(x) = \gamma_i(x) + \rho_i(x)
\]

\[= \frac{1}{2} \| A_i x - b_i \|_2^2 + \lambda_i \| x \|_1 \]

- \( \gamma_i \) has a Lipschitz continuous gradient: \( A_i^\top (A_i x - b_i) \)
- \( \rho_i \) has an efficient prox map: \( \text{prox}_{t \rho_i}(x) = \text{sgn}(x) \cdot (|x| - \lambda_i t)^+ \)

Other examples:

- SLOPE: \( \frac{1}{2} \| A x - b \|_2^2 + \sum_{i=1}^{n} \lambda_i x_{[i]} \)
- Nuclear norm minimization: \( \frac{1}{2} \| A(X) - b \|_2^2 + \| X \|_* \)
- Relaxations for optimal AC power flows on electricity grids
Trade-offs: centralized and distributed algorithms

Centralized algorithm

- **Expensive communication**: communicate all data to a central node
- **Expensive memory** requirements
- **Expensive computation**: full gradient evaluation
- **Violates privacy** constraints

Distributed algorithm

- Locally computing **partial gradients** more efficient (asynchronous)
- Each node updates **local** variables
- Have to ensure **consensus** for variables shared between nodes
- Side **constraints are harder** to impose.
- Potentially **slower convergence**
Consensus formulation

Distributed formulation

\[
\begin{align*}
\min_x & \quad \sum_{i=1}^{N} F_i(x_i) \\
\text{s. t.} & \quad x_i = x_j, \quad \forall (i, j) \in E \\
& \quad x_i \in X, \quad i \in N
\end{align*}
\]

- Create a local copy of variables for each node
- Impose consistency constraints to force the variables to be the same

Models a variety of important applications,
- distributed linear regression (Mateos et al., 2010),
- distributed control (Necoara and Suykens, 2008),
- machine learning (McDonald et al., 2010),
- estimation using sensor networks (Lesser et al., 2003).
Proposed solution

A distributed first-order augmented Lagrangian (DFAL) algorithm

- uses only local computations
- circumvents privacy, communication and memory issues
- can be easily extended to handle global side constraint $Ex - q \in \mathcal{K}$
- works in asynchronous setting

Need some additional constraints on $\gamma$ and $\rho$

- $\gamma \approx$ loss function
- $\rho \approx$ norm or combinations of norms
## Previous work

<table>
<thead>
<tr>
<th>Reference</th>
<th>$F_i$</th>
<th>operation/iter</th>
<th>iter # for $\epsilon$-feas</th>
<th>iter # for $\epsilon$-opt</th>
<th>comm steps</th>
<th>asynch</th>
<th>Can handle constraints?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duchi et al. (2012)</td>
<td>convex, Lipschitz cont</td>
<td>subgrad, projection</td>
<td>unknown</td>
<td>$O(1/\epsilon^2)$</td>
<td>$O(1/\epsilon^2)$</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Nedic and Ozdaglar (2009)</td>
<td>convex</td>
<td>subgrad.</td>
<td>$O(1)$</td>
<td>$O(1/\epsilon^2)$</td>
<td>$O(1/\epsilon^2)$</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Jakovetic et al. (2014)</td>
<td>smooth convex bounded $\nabla F_i$</td>
<td>$\nabla F_i$</td>
<td>$O(1/\sqrt{\epsilon})$</td>
<td>$O(1/\sqrt{\epsilon})$</td>
<td>$O(1/\sqrt{\epsilon})$</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Chen and Ozdaglar (2012)</td>
<td>composite convex $F_i = \rho + \gamma_i$ bounded $\nabla \gamma_i$</td>
<td>prox$_\rho$, $\nabla \gamma_i$</td>
<td>$O(1/\sqrt{\epsilon})$</td>
<td>$O(1/\sqrt{\epsilon})$</td>
<td>$O(1/\epsilon)$</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Wei and Ozdaglar (2012)</td>
<td>strictly convex</td>
<td>prox$_{F_i}$</td>
<td>unknown</td>
<td>$O(1/\epsilon)$</td>
<td>$O(1/\epsilon)$</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Makhdoumi and Ozdaglar (2014)</td>
<td>convex</td>
<td>prox$_{F_i}$</td>
<td>$O(1/\epsilon)$</td>
<td>$O(1/\epsilon)$</td>
<td>$O(1/\epsilon)$</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Wei and Ozdaglar (2013)</td>
<td>convex</td>
<td>prox$_{F_i}$</td>
<td>$O(1/\epsilon)$</td>
<td>$O(1/\epsilon)$</td>
<td>$O(1/\epsilon)$</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td><strong>DFAL</strong></td>
<td>composite convex $F_i = \rho_i + \gamma_i$</td>
<td>prox$_{\rho_i}$, $\nabla \gamma_i$</td>
<td>$O(1/\epsilon)$</td>
<td>$O(1/\epsilon)$</td>
<td>$O(1/\epsilon)$</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
## Previous work: competitive algorithms

<table>
<thead>
<tr>
<th>Reference</th>
<th>$F_i$</th>
<th>operation/iter</th>
<th>iter # for $\epsilon$-feas</th>
<th>iter # for $\epsilon$-opt</th>
<th>comm steps</th>
<th>asynch</th>
<th>Can handle constraints?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jakovetic et al. (2014)</td>
<td>smooth convex bounded $\nabla F_i$</td>
<td>$\nabla F_i$</td>
<td>$O(1/\sqrt{\epsilon})$</td>
<td>$O(1/\sqrt{\epsilon})$</td>
<td>$O(1/\sqrt{\epsilon})$</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Chen and Ozdaglar (2012)</td>
<td>composite convex $F_i = \rho + \gamma_i$</td>
<td>$\text{prox}_\rho, \nabla \gamma_i$</td>
<td>$O(1/\sqrt{\epsilon})$</td>
<td>$O(1/\sqrt{\epsilon})$</td>
<td>$O(1/\epsilon)$</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Wei and Ozdaglar (2012)</td>
<td>strictly convex</td>
<td>$\text{prox}_{F_i}$</td>
<td>unknown</td>
<td>$O(1/\epsilon)$</td>
<td>$O(1/\epsilon)$</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Makhdoumi and Ozdaglar (2014)</td>
<td>convex</td>
<td>$\text{prox}_{F_i}$</td>
<td>$O(1/\epsilon)$</td>
<td>$O(1/\epsilon)$</td>
<td>$O(1/\epsilon)$</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Wei and Ozdaglar (2013)</td>
<td>convex</td>
<td>$\text{prox}_{F_i}$</td>
<td>$O(1/\epsilon)$</td>
<td>$O(1/\epsilon)$</td>
<td>$O(1/\epsilon)$</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td><strong>DFAL</strong></td>
<td>composite convex $F_i = \rho_i + \gamma_i$</td>
<td>$\text{prox}_{\rho_i}, \nabla \gamma_i$</td>
<td>$O(1/\epsilon)$</td>
<td>$O(1/\epsilon)$</td>
<td>$O(1/\epsilon)$</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Highlights of DFAL
- Imposes much milder conditions on $\rho$ and $\gamma$
  - Does not require $\nabla \gamma$ to be bounded
  - Does not require that non-smooth $\rho$ to be the same at all nodes
- Works in an asynchronous setting
- **Only** one that allows side constraints
DFAL: Augmented Lagrangian formulation

General distributed formulation: \( \mathbf{x} = (x_1, \ldots, x_N)^\top \)

\[
\begin{align*}
\min_{\mathbf{x}} & \quad \sum_{i \in \mathcal{N}} \rho_i(x_i) + \sum_{i \in \mathcal{N}} \gamma_i(x_i) \\
\text{s.t.} & \quad A\mathbf{x} = b \quad (A \in \mathbb{R}^{m \times nN}, \ \text{rank}(A) = m)
\end{align*}
\]

Augmented Lagrangian formulation: dual \( \theta^{(k)} \) and penalty \( 1/\lambda^{(k)} \)

\[
P^{(k)}(\mathbf{x}) = \lambda^{(k)} \left( \bar{\rho}(\mathbf{x}) + \bar{\gamma}(\mathbf{x}) - (\theta^{(k)})^\top (A\mathbf{x} - b) \right) + \frac{1}{2} \| A\mathbf{x} - b \|_2^2
\]

\[
= \lambda^{(k)} \bar{\rho}(\mathbf{x}) + \lambda^{(k)} \bar{\gamma}(\mathbf{x}) + \frac{1}{2} \| A\mathbf{x} - b - \lambda^{(k)}\theta^{(k)} \|_2^2
\]

\[
= \lambda^{(k)} \bar{\rho}(\mathbf{x}) + f^{(k)}(\mathbf{x})
\]

where \( \nabla f^{(k)} \) is Lipschitz continuous with constant \( \lambda^{(k)} \bar{L} + \sigma_{\text{max}}^2(A) \)
DFAL: Inexactly solve the augmented Lagrangian

Two stopping conditions:

\[(a) \quad P^{(k)}(x^{(k)}) - P^{(k)}(x_\star^{(k)}) \leq \alpha^{(k)}, \quad x_\star^{(k)} \in \text{argmin}_{x \in \mathbb{R}^n} P^{(k)}(x)\]

\[(b) \quad \exists g_i^{(k)} \in \partial x_i P^{(k)}(x) \big| x = x^{(k)} \text{ s.t.} \quad \max_{i \in \mathcal{N}} \|g_i^{(k)}\|_2 \leq \frac{\xi^{(k)}}{\sqrt{N}}\]  

Prox-gradient step

\[\text{prox}_{t_k \rho}(x) = \text{argmin}_{y \in \mathbb{R}^n} \left\{ \rho(y) + \frac{1}{2t_k} \|y - (x - t_k \nabla f_k(x))\|^2 \right\}\]

Algorithm DFAL \( \left( \lambda^{(1)}, \alpha^{(1)}, \xi^{(1)} \right) \)

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

Step \( k: \ (k \geq 1) \)

1. Do prox-gradient steps until \( x^{(k)} \) satisfies (1)(a) or (1)(b)

2. \( \theta^{(k+1)} = \theta^{(k)} - \frac{Ax^{(k)} - b}{\lambda^{(k)}} \)

3. \( \lambda^{(k+1)} = c \lambda^{(k)}, \quad \alpha^{(k+1)} = c^2 \alpha^{(k)}, \quad \xi^{(k+1)} = c^2 \xi^{(k)} \)
**Theorem:** There exists explicitly defined \((\lambda^{(1)}, \alpha^{(1)}, \xi^{(1)}, c)\) such that for all \(\epsilon > 0\), DFAL iterates \(x^{(k)}\) are

- \(\epsilon\)-feasible, i.e. \(\|Ax^{(k)} - b\|_2 \leq \epsilon\), and
- \(\epsilon\)-optimal, i.e. \(|\bar{F}(x^{(k)}) - \bar{F}^*| \leq \epsilon\)

for all \(k \geq K(\epsilon) := \log \frac{1}{\epsilon} (\frac{\bar{C}'}{\epsilon})\) for some \(\bar{C}' > 0\) which requires \(O\left(\frac{1}{\epsilon}\right)\) communication steps along the edges \(E\).

**Main ingredients of the proof**

- New multi-step accelerated proximal gradient method
- Prove duals \(\theta^{(k)}\) are bounded: Assumptions on \(\gamma\) and \(\rho\)
- Prove iterates \(x^{(k)}\) are bounded: Assumptions on \(\gamma\) and \(\rho\)
- Induction establishes the result
DFAL: Assumptions on $\gamma$ and $\rho$

**Smooth function $\gamma$**

- lower bounded: $\exists \gamma \in \mathbb{R}$ s.t. $\gamma(x) \geq \gamma$ for all $x \in \mathbb{R}^n$.

Examples: quadratic-loss $\|Ax - b\|_2^2$, Huber-loss $\sum_{i=1}^{m} h_{\delta}( a_i^T x - b_i )$

logistic-loss $\sum_{i=1}^{m} \log \left(1 + e^{-b_i a_i^T x}\right)$

**Non-smooth function $\rho$**

- bounded sub-gradients: $\exists B > 0$ s.t. $\|q\|_2 \leq B \ \forall q \in \partial \rho(x), \forall x \in \mathbb{R}^n$
- lower bounded by a norm: $\exists \tau > 0$ s.t. $\tau \|x\|_2 \leq \rho(x) \ \forall x \in \mathbb{R}^n$

Examples: $\|\cdot\|_\alpha$ with $\alpha \in \{1, 2, \infty\}$, group norm, nuclear norm, weighted sum of norms, e.g. sparse group norm
DFAL: Consensus problem

Main ingredient in **DFAL** is the gradient $\nabla_{x_i} f^{(k)}(x)$: Suppose $\theta^{(0)} \equiv 0$

\[
\nabla_{x_i} f^{(k)}(x) = \lambda^{(k)} \nabla \gamma_i(x_i) + d_i \left( x_i + \bar{x}_i^{(k)} \right) - \sum_{j \in O_i} \left( x_j + \bar{x}_j^{(k)} \right)
\]

where $\bar{x}^{(k)} := \lambda^{(k)} \sum_{t=1}^{k-1} \frac{1}{\lambda(t)} x^{(t)}$, $O_i = \{ j \in \mathcal{N} : (i, j) \in \mathcal{E} \}$

Note
- Gradient can be computed **without** explicitly computing the duals $\theta^{(k)}$.
- Neighbors only communicate the current iterate $x_j$
- Only need local graph structure
Theorem: DFAL iterates are $\varepsilon$-optimal and $\varepsilon$-feasible within $K(\varepsilon) = \mathcal{O}(\log(\frac{1}{\varepsilon}))$ iterations, requiring at most $\mathcal{O}(\frac{\psi_1^{1.5}}{d_{\text{min}} \frac{1}{\varepsilon}})$ communication steps.

Remarks:

1. $d_{\text{min}}$ denotes the degree of smallest degree node in $G$
2. $\psi_1 \geq \psi_2 \geq \ldots \psi_{N-1} > \psi_N = 0$: eigenvalues of the graph Laplacian
3. Since $\psi_{N-1} \leq d_{\text{min}}$, the main theorem implies the number of communication steps can be bounded above by $\mathcal{O}(\frac{\psi_1^{1.5}}{\psi_{N-1} \frac{1}{\varepsilon}})$. 
DFAL: Global Constraints

Global constraints $Ex - q \in \mathcal{K}$ where projection $\Pi_{\mathcal{K}}$ is easy

$$
\min \sum_{i \in \mathcal{N}} \rho_i(x) + \gamma_i(x) \quad \s.t. \quad Ex - q \in \mathcal{K}
$$

$$
\min \bar{\rho}(x) + \bar{\gamma}(x) \quad \s.t. \quad Cx = 0
$$

Augmented Lagrangian: duals $\theta^{(k)}$, $\{\mu_i^{(k)}\}_{i \in \mathcal{N}}$, and penalty $1/\lambda^{(k)}$

$$
P^{(k)}(x) = \lambda^{(k)}(\bar{\rho}(x) + \bar{\gamma}(x)) + \frac{1}{2} \|Cx - \lambda^{(k)}\theta^{(k)}\|_2^2
$$

$$
+ \sum_{i \in \mathcal{N}} \frac{1}{2} d_{\mathcal{K}} \left( E x_i - q - \lambda^{(k)}\mu_i^{(k)} \right)^2
$$

$$
\Delta \equiv p(x)
$$

- $\nabla p(x) = E^T (Ex - q - \Pi_{\mathcal{K}}(Ex - q)) = -E^T \Pi_{\mathcal{K}^*}(q - Ex)$
- $\nabla p$ is Lipschitz continuous with constant $\sigma_{\text{max}}^2(E)$
Algorithm DFAL \( \left( \lambda^{(1)}, \alpha^{(1)}, \xi^{(1)} \right) \)

Step 0: Set \( \theta^{(1)} = 0, \mu_{i}^{(1)} = 0, i \in \mathcal{N}, k = 1 \)

Step \( k: (k \geq 1) \)

1. **Do prox-gradient steps until** \( x^{(k)} \) **satisfies** (1)(a) or (1)(b)

2. \( \theta^{(k+1)} = \theta^{(k)} - \frac{C x^{(k)}}{\lambda^{(k)}} \) (do not compute this explicitly)

3. \( \mu_{i}^{(k+1)} = \frac{1}{\lambda^{(k)}} \Pi_{\mathcal{K}^*} \left( \lambda^{(k)} \mu_{i}^{(k)} + q - E x_{i}^{(k)} \right) \in \mathcal{K}^* \) for all \( i \in \mathcal{N} \)

4. \( \lambda^{(k+1)} = c \lambda^{(k)}, \quad \alpha^{(k+1)} = c^2 \alpha^{(k)}, \quad \xi^{(k+1)} = c^2 \xi^{(k)} \)

Two stopping conditions

- (1)(a): \( \alpha^{(k)} \)-optimality
- (1)(b): \( \xi^{(k)} \)-stationarity

Rate result: \( \epsilon \)-feasible and \( \epsilon \)-optimal in \( \mathcal{O} \left( \frac{1}{\epsilon} \right) \) communication steps
DFAL vs ADMM

ADMM ≡ Alternating direction method of multipliers

- Competing algorithms are ADMM-type algorithms

**DFAL** algorithm

- Does not need an order over nodes
- Side constraints are not a problem
- Can convert into an asynchronous algorithm
DFAL: Problems with synchronous algorithm

**DFAL** is a **synchronous** algorithm

- All nodes have to complete computation before an update
- Rate is dominated by the slowest node
- Serious issue if computational resources at nodes are very variable

Asynchronous **DFAL**

- Computation terminates at node $i$.
- Node $i$ signals to all its neighbors to terminate, and
- Shares its local variable with them.
DFAL: Problems with synchronous algorithm

**DFAL** is a synchronous algorithm
- All nodes have to complete computation before an update
- Rate is dominated by the slowest node
- Serious issue if computational resources at nodes are very variable

Asynchronous **DFAL**
- Computation terminates at node $i$.
- Node $i$ signals to all its neighbors to terminate, and
- Shares its local variable with them.

**Theorem**: The Asynchronous **DFAL** algorithm iterates are $\epsilon$-optimal and $\epsilon$-feasible with probability at least $1 - p$ in at most $O\left(\frac{\psi_1^{1.5}}{d_{\min}} \frac{1}{\epsilon} \frac{1}{p}\right)$ communication steps.
- A variant of the algorithm in Fercoq and Richtárik (2013)
Numerical study: Sparse group LASSO

Loss function: Huber loss  
\[ h_\delta(y) = \max \{ u^\top y - \frac{1}{2} \| u \|_2^2 : u_i \in [-\delta, \delta] \} \]

\[ \gamma(x) = \sum_{i=1}^{N} h_\delta(A_i x - b_i) \]

Non-smooth regularizer:

- Sparse solution and
- Few non-zero groups: \( G_i = \{ g_{ik} : g_{ik} \subseteq \{1, \ldots, n\} \} \) partition of \( \{1, \ldots, n\} \)

\[ \rho(x) = \beta_1 \sum_{i=1}^{N} \| x \|_1 + \beta_2 \sum_{i=1}^{n} \sum_{k=1}^{m_i} \| x_{g_{ik}} \|_2 \]

Optimization problem

\[ \min_{x \in \mathbb{R}^n} \left\{ \rho(x) + \sum_{i=1}^{N} h_\delta(A_i x - b_i) \right\} \]
Problem classes

Graph $G$: either a star or a clique on $N = 5$ or $10$ nodes

Rate result: clique should converge faster than star

Partition: $K = 10$ and $n_g \in \{100, 300\}$, i.e. $n = Kn_g \in \{1000, 3000\}$
  - Case I: randomly generate 1 partition $G$. Set $G_i \equiv G$.
  - Case II: randomly generate a partition for each node

Data matrices: For each node $m = \frac{n}{4N}$,
  - $A_{ij} \sim \mathcal{N}(0, 1)$ for $1 \leq i \leq m$ and $1 \leq j \leq n$
  - $b_i = A_i \bar{x}$ where $\bar{x}_j = (-1)^j e^{-(j-1)/n_g}$

Machine with 4 (2.75GHz) cores and 64 GB of RAM.
**Distributed ADMM Algorithm**

**ADMM:** Makhdoumi and Ozdaglar (2014) implemented ADMM on

\[ \min_{x,z} \sum_{i=1}^{N} \left( \rho_i(x_i) + \gamma_i(x_i) \right) \]

\[ F_i(x_i) \]

s.t. \[ \Omega_{ij} x_j = z_{ij}, \quad i \in \mathcal{N}, \ j \in \mathcal{N}_i := \mathcal{O}_i \cup \{i\}. \]

- \( \mathcal{O}(\epsilon^{-1}) \) communication steps and \( \mathcal{O}(\epsilon^{-1}) \) \( \text{prox}_{F_i} \)-computations
- Computing \( \text{prox}_{F_i} \) is almost as hard as solving the problem.

**SADMM:** To overcome the \( \text{prox}_{F_i} \) bottleneck, we propose “splitting” \( x_i \)

\[ \min_{x,z,\tilde{z}} \sum_{i \in \mathcal{N}} \rho_i(x_i) + \gamma_i(y_i) \]

s.t. \[ \Omega_{ij} x_j = z_{ij}, \quad i \in \mathcal{N}, \ j \in \mathcal{N}_i \]
\[ \Omega_{ij} y_j = \tilde{z}_{ij}, \quad i \in \mathcal{N}, \ j \in \mathcal{N}_i \]
\[ x_i = q_i, \quad y_i = q_i, \quad i \in \mathcal{N} \]
### Group sparse LASSO: Results for $n_g = 100$

<table>
<thead>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDPT3 (C)</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>1E-3</td>
<td>N/A</td>
<td>10</td>
<td>2173</td>
</tr>
<tr>
<td>APG (C)</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>N/A</td>
<td>10</td>
<td>2173</td>
</tr>
<tr>
<td>DFAL (D)</td>
<td>6E-4, 7E-4</td>
<td>4E-4, 6E-4</td>
<td>5E-5, 2E-5</td>
<td>5, 5</td>
</tr>
<tr>
<td></td>
<td>3E-6, 5E-6</td>
<td>2E-6, 5E-6</td>
<td>16, 11</td>
<td>9232, 9083</td>
</tr>
<tr>
<td>ADMM (D)</td>
<td>6E-4, 7E-5</td>
<td>1E-4, 1E-4</td>
<td>1125, 808</td>
<td>353, 253</td>
</tr>
<tr>
<td>SADMM (D)</td>
<td>1E-4, 3E-4</td>
<td>1E-4, 3E-4</td>
<td>771, 784</td>
<td>592, 606</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$N = 5, n = 1000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDPT3 (C)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>APG (C)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>DFAL (D)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>ADMM (D)</td>
</tr>
<tr>
<td>SADMM (D)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$N = 10, n = 1000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDPT3 (C)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>APG (C)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>DFAL (D)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>ADMM (D)</td>
</tr>
<tr>
<td>SADMM (D)</td>
</tr>
</tbody>
</table>

**Termination time $T = 3600s$ (1 hour)**
### Group sparse LASSO: Results for $n_g = 300$

<table>
<thead>
<tr>
<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>
</tr>
<tr>
<td>SDPT3 (C)</td>
<td>0</td>
<td>0</td>
<td>806</td>
<td>1653</td>
</tr>
<tr>
<td>APG (C)</td>
<td>1E-3</td>
<td>N/A</td>
<td>253</td>
<td>N/A</td>
</tr>
<tr>
<td>DFAL (D)</td>
<td>2E-4, 5E-4</td>
<td>2E-4, 4E-4</td>
<td>77, 64</td>
<td>80, 65</td>
</tr>
<tr>
<td>AFAL (D)</td>
<td>1E-4, 6E-4</td>
<td>2E-5, 6E-4</td>
<td>164, 99</td>
<td>273, 99</td>
</tr>
<tr>
<td>ADMM (D)</td>
<td>5E-2, 1E-3</td>
<td>5E-2, 1E-3</td>
<td>5E-3, 1E-3</td>
<td>7E-3, 1E-3</td>
</tr>
<tr>
<td>SADMM (D)</td>
<td>2E-2, 7E-2</td>
<td>2E-2, 8E-2</td>
<td>2E-3, 3E-3</td>
<td>2E-3, 3E-3</td>
</tr>
</tbody>
</table>

$N = 5, n = 3000$

<table>
<thead>
<tr>
<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>
</tr>
<tr>
<td>SDPT3 (C)</td>
<td>0</td>
<td>0</td>
<td>806</td>
<td>1641</td>
</tr>
<tr>
<td>APG (C)</td>
<td>1E-3</td>
<td>N/A</td>
<td>253</td>
<td>N/A</td>
</tr>
<tr>
<td>DFAL (D)</td>
<td>1E-4, 6E-4</td>
<td>6E-4, 1E-3</td>
<td>130, 80</td>
<td>122, 82</td>
</tr>
<tr>
<td>AFAL (D)</td>
<td>2E-4, 7E-4</td>
<td>6E-4, 1E-3</td>
<td>350, 294</td>
<td>437, 288</td>
</tr>
<tr>
<td>ADMM (D)</td>
<td>5E-2, 8E-2</td>
<td>5E-2, 8E-2</td>
<td>7E-3, 9E-3</td>
<td>7E-3, 9E-3</td>
</tr>
<tr>
<td>SADMM (D)</td>
<td>3E-1, 3E+0</td>
<td>3E-1, 3E+0</td>
<td>4E-3, 2E-2</td>
<td>4E-3, 2E-2</td>
</tr>
</tbody>
</table>

$N = 10, n = 3000$

**Termination time $T = 3600s$ (1 hour)**
Summary

- Propose a distributed first-order Augmented Lagrangian algorithm
- Guarantees a bound on the consensus error
- Can handle global constraints


