PARALLEL MULTI-BLOCK ADMM WITH $o(1/k)$ CONVERGENCE

WEI DENG[*], MING-JUN LAI[†], ZHIMIN PENG[‡] , AND WOTAO YIN[‡]

Abstract. This paper introduces a parallel and distributed extension to the alternating direction method of multipliers (ADMM) for solving convex problem:

$$\begin{aligned} \text{minimize} \quad & f_1(\mathbf{x}_1) + \cdots + f_N(\mathbf{x}_N) \\ \text{subject to} \quad & A_1\mathbf{x}_1 \ + \cdots + A_N\mathbf{x}_N = c, \\ & \mathbf{x}_1 \in \mathcal{X}_1, \ \ldots, \ \mathbf{x}_N \in \mathcal{X}_N. \end{aligned}$$

The algorithm decomposes the original problem into N smaller subproblems and solves them in parallel at each iteration. This Jacobian-type algorithm is well suited for distributed computing and is particularly attractive for solving certain large-scale problems.

This paper introduces a few novel results. Firstly, it shows that extending ADMM straightforwardly from the classic Gauss-Seidel setting to the Jacobian setting, from 2 blocks to N blocks, will preserve convergence if matrices A_i are mutually near-orthogonal and have full column-rank. Secondly, for general matrices A_i, this paper proposes to add proximal terms of different kinds to the N subproblems so that the subproblems can be solved in flexible and efficient ways and the algorithm converges globally at a rate of $o(1/k)$. Thirdly, a simple technique is introduced to improve some existing convergence rates from $O(1/k)$ to $o(1/k)$.

In practice, some conditions in our convergence theorems are conservative. Therefore, we introduce a strategy for dynamically tuning the parameters in the algorithm, leading to substantial acceleration of the convergence in practice. Numerical results are presented to demonstrate the efficiency of the proposed method in comparison with several existing parallel algorithms.

We implemented our algorithm on Amazon EC2, an on-demand public computing cloud, and report its performance on very large-scale basis pursuit problems with distributed data.

Key words. alternating direction method of multipliers, ADMM, parallel and distributed computing, convergence rate

1. Introduction.

We consider the following convex optimization problem with N $(N \geq 2)$ blocks of variables:

$$\min_{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N} \ \sum_{i=1}^{N} f_i(\mathbf{x}_i) \quad \text{s.t.} \ \sum_{i=1}^{N} A_i\mathbf{x}_i = c, \tag{1.1}$$

[*]Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005. (wei.deng@rice.edu)

[†]Department of Mathematics, University of Georgia, Athens, GA 30602. (mjlai@math.uga.edu)

[‡]Department of Mathematics, University of California, Los Angeles, CA 90095. (zhimin.peng / wotaoyin@math.ucla.edu)

where $\mathbf{x}_i \in \mathbb{R}^{n_i}$, $A_i \in \mathbb{R}^{m\times n_i}$, $c \in \mathbb{R}^m$, and $f_i : \mathbb{R}^{n_i} \to (-\infty, +\infty]$ are closed proper convex functions, $i = 1, 2, \ldots, N$. If an individual block is subject to constraint $\mathbf{x}_i \in \mathcal{X}_i$, where $\mathcal{X}_i \subseteq \mathbb{R}^{n_i}$ is a nonempty closed convex set, it can be incorporated in the objective function f_i using the indicator function:

$$I_{\mathcal{X}_i}(\mathbf{x}_i) = \begin{cases} 0 & \text{if } \mathbf{x}_i \in \mathcal{X}_i, \\ +\infty & \text{otherwise.} \end{cases} \tag{1.2}$$

The problem (1.1) is also referred to as an *extended monotropic programming problem* [2]. The special case that each $\mathbf{x}_i$ is a scalar (i.e., $n_i = 1$) is called a *monotropic programming problem* [26]. Such optimization problems arise from a broad spectrum of applications including numerical partial differential equations, signal and image processing, compressive sensing, statistics and machine learning. See [11, 1, 3, 29, 4, 24, 22, 31, 23] and the references therein for a number of examples.

In this paper, we focus on *parallel and distributed* optimization algorithms for solving the problem (1.1). Since both of the objective function and constraints of (1.1) are summations of terms on individual $\mathbf{x}_i$'s (we call them *separable*), the problem can be decomposed into N smaller subproblems, which can be solved in a parallel and distributed manner.

1.1. Literature review. A simple distributed algorithm for solving (1.1) is *dual decomposition* [9], which is essentially a *dual ascent method* or *dual subgradient method* [28] as follows. Consider the Lagrangian for problem (1.1):

$$\mathcal{L}(\mathbf{x}_1, \ldots, \mathbf{x}_N, \lambda) = \sum_{i=1}^N f_i(\mathbf{x}_i) - \lambda^\top \left(\sum_{i=1}^N A_i \mathbf{x}_i - c \right) \tag{1.3}$$

where $\lambda \in \mathbb{R}^m$ is the Lagrangian multiplier or the dual variable. The method of dual decomposition iterates as follows: for $k \geq 1$,

$$\begin{cases} (\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}, \ldots, \mathbf{x}_N^{k+1}) = \arg\min_{\{\mathbf{x}_i\}} \ \mathcal{L}(\mathbf{x}_1, \ldots, \mathbf{x}_N, \lambda^k), \\ \lambda^{k+1} = \lambda^k - \alpha_k \left(\sum_{i=1}^N A_i \mathbf{x}_i^{k+1} - c \right), \end{cases} \tag{1.4}$$

where $\alpha_k > 0$ is a step-size. Since all the $\mathbf{x}_i$'s are separable in the Lagrangian function (1.3), the $\mathbf{x}$-update step reduces to solving N individual $\mathbf{x}_i$-subproblems:

$$\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} f_i(\mathbf{x}_i) - \langle \lambda^k, A_i \mathbf{x}_i \rangle, \ \text{for } i = 1, 2, \ldots, N, \tag{1.5}$$

and thus they can be carried out in parallel. With suitable choice of α_k and certain assumptions, dual decomposition is guaranteed to converge to an optimal solution [28]. However, the convergence of such subgradient method often tends to be slow in practice. Its convergence rate for general convex problems is $O(1/\sqrt{k})$.

Another effective distributed approach is based on the *alternating direction method of multipliers* (ADMM). ADMM was introduced in [10, 12] to solve the special case of problem (1.1) with two blocks of variables ($N = 2$). It utilizes the *augmented Lagrangian* for (1.1):

$$\mathcal{L}_\rho(\mathbf{x}_1, \ldots, \mathbf{x}_N, \lambda) = \sum_{i=1}^N f_i(\mathbf{x}_i) - \lambda^\top \left(\sum_{i=1}^N A_i \mathbf{x}_i - c \right) + \frac{\rho}{2} \left\| \sum_{i=1}^N A_i \mathbf{x}_i - c \right\|_2^2, \tag{1.6}$$

which incorporates a quadratic penalty of the constraints (with a parameter $\rho > 0$) into the Lagrangian. In each iteration, the augmented Lagrangian is minimized over $\mathbf{x}_1$ and $\mathbf{x}_2$ separately, one after the other, followed by a dual update for λ. The iterative scheme of ADMM is outlined below:

$$\begin{cases} \mathbf{x}_1^{k+1} = \arg\min_{\mathbf{x}_1} \mathcal{L}_\rho(\mathbf{x}_1, \mathbf{x}_2^k, \lambda^k), \\ \mathbf{x}_2^{k+1} = \arg\min_{\mathbf{x}_2} \mathcal{L}_\rho(\mathbf{x}_1^{k+1}, \mathbf{x}_2, \lambda^k), \\ \lambda^{k+1} = \lambda^k - \rho(A_1\mathbf{x}_1^{k+1} + A_2\mathbf{x}_2^{k+1} - c). \end{cases} \tag{1.7}$$

To solve the problem (1.1) with $N \geq 3$ using ADMM, one can first convert the multi-block problem into an equivalent two-block problem via variable splitting [1, 3, 30]:

$$\begin{aligned} \min_{\{\mathbf{x}_i\},\{\mathbf{z}_i\}} \quad & \sum_{i=1}^N f_i(\mathbf{x}_i) + I_{\mathcal{Z}}(\mathbf{z}_1, \ldots, \mathbf{z}_N) \\ \text{s.t.} \quad & A_i\mathbf{x}_i - \mathbf{z}_i = \frac{c}{N}, \ \forall i = 1, 2, \ldots, N, \end{aligned} \tag{1.8}$$

where $I_{\mathcal{Z}}$ is a indicator function defined by (1.2), and the convex set $\mathcal{Z}$ is given by

$$\mathcal{Z} = \left\{ (\mathbf{z}_1, \ldots, \mathbf{z}_N) : \sum_{i=1}^N \mathbf{z}_i = 0 \right\}.$$

The variables in (1.8) can be grouped into two blocks: $\mathbf{x} := (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and $\mathbf{z} := (\mathbf{z}_1, \ldots, \mathbf{z}_N)$, so that ADMM can directly apply. The augmented Lagrangian for

(1.8) is given by

(1.9)

$$\mathcal{L}_\rho(\mathbf{x},\mathbf{z},\lambda) = \sum_{i=1}^N f_i(\mathbf{x}_i) + I_{\mathcal{Z}}(\mathbf{z}) - \sum_{i=1}^N \lambda_i^\top \left(A_i\mathbf{x}_i - \mathbf{z}_i - \frac{c}{N}\right) + \frac{\rho}{2}\sum_{i=1}^N \left\|A_i\mathbf{x}_i - \mathbf{z}_i - \frac{c}{N}\right\|_2^2.$$

Since all the $\mathbf{x}_i$'s are now fully decoupled, the resulting $\mathbf{x}$-subproblem decomposes into N individual $\mathbf{x}_i$-subproblems, which can be carried out in parallel. The resulting $\mathbf{z}$-subproblem is a simple quadratic problem:

$$\mathbf{z}^{k+1} = \underset{\{\mathbf{z}:\sum_{i=1}^N \mathbf{z}_i = 0\}}{\arg\min} \sum_{i=1}^N \frac{\rho}{2}\left\|A_i\mathbf{x}_i - \mathbf{z}_i - \frac{c}{N} - \frac{\lambda_i^k}{\rho}\right\|_2^2, \tag{1.10}$$

which admits a closed-form solution. The details of the algorithm are summarized below:

Algorithm 1: Variable Splitting ADMM (VSADMM)

```
Initialize $\mathbf{x}^0,\ \lambda^0,\ \rho > 0$;
for $k = 0,\ 1, \ldots$ do
    Update $\mathbf{z}_i$ then $\mathbf{x}_i$ for $i = 1, \ldots, N$ in parallel by:
    $\mathbf{z}_i^{k+1} = \left(A_i\mathbf{x}_i^k - \frac{c}{N} - \frac{\lambda_i^k}{\rho}\right) - \frac{1}{N}\left\{\sum_{j=1}^N A_j\mathbf{x}_j^k - \frac{c}{N} - \frac{\lambda_j^k}{\rho}\right\}$;
    $\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \frac{\rho}{2}\left\|A_i\mathbf{x}_i - \mathbf{z}_i^{k+1} - \frac{c}{N} - \frac{\lambda_i^k}{\rho}\right\|_2^2$;
    Update $\lambda_i^{k+1} = \lambda_i^k - \rho\left(A_i\mathbf{x}_i^{k+1} - \mathbf{z}_i^{k+1} - \frac{c}{N}\right), \forall i = 1, \ldots, N$.
```

The distributed ADMM approach based on (1.8), by introducing splitting variables, *substantially increases the number of variables and constraints* in the problem, especially when N is large.

We prefer to extending the ADMM framework for solving (1.1) rather than first converting (1.1) to a two-block problem and then applying the classic ADMM. A natural extension is to simply replace the two-block alternating minimization scheme by a sweep of Gauss-Seidel update, namely, update $\mathbf{x}_i$ for $i = 1, 2, \ldots, N$ sequentially as follows:

$$\begin{aligned} \mathbf{x}_i^{k+1} &= \underset{\mathbf{x}_i}{\arg\min}\ \mathcal{L}_\rho(\mathbf{x}_1^{k+1}, \ldots, \mathbf{x}_{i-1}^{k+1}, \mathbf{x}_i, \mathbf{x}_{i+1}^k, \ldots, \mathbf{x}_N^k, \lambda^k) \\ &= \underset{\mathbf{x}_i}{\arg\min}\ f_i(\mathbf{x}_i) + \frac{\rho}{2}\left\|\sum_{j<i} A_j\mathbf{x}_j^{k+1} + A_i\mathbf{x}_i + \sum_{j>i} A_j\mathbf{x}_j^k - c - \frac{\lambda^k}{\rho}\right\|_2^2. \end{aligned} \tag{1.11}$$

Such *Gauss-Seidel ADMM* (Algorithm 2) has been considered lately, e.g., in [17, 20].

However, it has been shown that the algorithm may not converge for $N \geq 3$ [5]. Although lack of convergence guarantee, some empirical studies show that Algorithm 2 is still very effective at solving many practical problems (see, e.g., [24, 29, 30]).

Algorithm 2: Gauss-Seidel ADMM

```
Initialize $\mathbf{x}^0,\ \lambda^0,\ \rho > 0$;
for $k = 0,\ 1, \dots$ do
    Update $\mathbf{x}_i$ for $i = 1, \dots, N$ *sequentially* by:
    $\mathbf{x}_i^{k+1} = \min_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \frac{\rho}{2}\left\|\sum_{j<i} A_j\mathbf{x}_j^{k+1} + A_i\mathbf{x}_i + \sum_{j>i} A_j\mathbf{x}_j^k - c - \frac{\lambda^k}{\rho}\right\|_2^2$;
    Update $\lambda^{k+1} = \lambda^k - \rho\left(\sum_{i=1}^N A_i\mathbf{x}_i^{k+1} - c\right)$.
```

A disadvantage of Gauss-Seidel ADMM is that the blocks are updated one after another, which is not amenable for parallelization.

1.2. Jacobian scheme. To overcome this disadvantage, this paper considers using a Jacobi-type scheme that updates all the N blocks in parallel:

$$\begin{aligned}\mathbf{x}_i^{k+1} &= \arg\min_{\mathbf{x}_i}\ \mathcal{L}_\rho(\mathbf{x}_1^k, \dots, \mathbf{x}_{i-1}^k, \mathbf{x}_i, \mathbf{x}_{i+1}^k, \dots, \mathbf{x}_N^k, \lambda^k) \\ &= \arg\min_{\mathbf{x}_i}\ f_i(\mathbf{x}_i) + \frac{\rho}{2}\left\|A_i\mathbf{x}_i + \sum_{j\neq i} A_j\mathbf{x}_j^k - c - \frac{\lambda^k}{\rho}\right\|_2^2,\ \forall i = 1, \dots, N. \qquad (1.12)\end{aligned}$$

We refer to it as *Jacobian ADMM*; see Algorithm 3.

Algorithm 3: Jacobian ADMM

```
Initialize $\mathbf{x}^0,\ \lambda^0,\ \rho > 0$;
for $k = 0,\ 1, \dots$ do
    Update $\mathbf{x}_i$ for $i = 1, \dots, N$ *in parallel* by:
    $\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \frac{\rho}{2}\left\|A_i\mathbf{x}_i + \sum_{j\neq i} A_j\mathbf{x}_j^k - c - \frac{\lambda^k}{\rho}\right\|_2^2$;
    Update $\lambda^{k+1} = \lambda^k - \rho\left(\sum_{i=1}^N A_i\mathbf{x}_i^{k+1} - c\right)$.
```

The parallelization comes with a cost: this scheme is more likely to diverge than the Gauss-Seidel scheme for the same parameter ρ. In fact, it may diverge even in the two-block case; see [16] for such an example. In order to guarantee its convergence, either additional assumptions or modifications to the algorithm must be made.

In Section 4, we show that if matrices A_i are mutually near-orthogonal and have full column-rank, then Algorithm 3 converges globally. For general cases, a few variants of Jacobian ADMM have been proposed in [15, 16] by taking additional correction steps at every iteration.

In this paper, we propose *Proximal Jacobian ADMM*; see Algorithm 4). Compared with Algorithm 3, there are a proximal term $\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_i^k\|_{P_i}^2$ for each $\mathbf{x}_i$-subproblem and a damping parameter $\gamma > 0$ for the update of λ. Here $P_i \succeq 0$ is some symmetric and positive semi-definite matrix and we let $\|\mathbf{x}_i\|_{P_i}^2 := \mathbf{x}_i^\top P_i \mathbf{x}_i$.

Algorithm 4: Proximal Jacobian ADMM

```
Initialize: $\mathbf{x}_i^0$ $(i = 1, 2, \dots, N)$ and $\lambda^0$;
for $k = 0, 1, \dots$ do
    Update $\mathbf{x}_i$ for $i = 1, \dots, N$ in parallel by:
    $\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \frac{\rho}{2}\left\|A_i\mathbf{x}_i + \sum_{j\neq i} A_j\mathbf{x}_j^k - c - \frac{\lambda^k}{\rho}\right\|_2^2 + \frac{1}{2}\left\|\mathbf{x}_i - \mathbf{x}_i^k\right\|_{P_i}^2$;
    Update $\lambda^{k+1} = \lambda^k - \gamma\rho(\sum_{i=1}^N A_i\mathbf{x}_i^{k+1} - c)$.
```

The proposed algorithm has a few advantages. First of all, as we will show, it enjoys global convergence as well as an $o(1/k)$ convergence rate under conditions on P_i and γ. Secondly, when the $\mathbf{x}_i$-subproblem is not strictly convex, adding the proximal term can make the subproblem strictly or strongly convex, making it more stable. Thirdly, we provide multiple choices for matrices P_i with which the subproblems can be made easier to solve. Specifically, the $\mathbf{x}_i$-subproblem contains a quadratic term $\frac{\rho}{2}\mathbf{x}_i A_i^\top A_i \mathbf{x}_i$. When $A_i^\top A_i$ is ill-conditioned or computationally expensive to invert, one can let $P_i = D_i - \rho A_i^\top A_i$, which cancels the quadratic term $\frac{\rho}{2}\mathbf{x}_i A_i^\top A_i \mathbf{x}_i$ and adds $\frac{1}{2}\mathbf{x}_i D_i \mathbf{x}_i$. The matrix D_i can be chosen as some well-conditioned and simple matrix (e.g., a diagonal matrix), thereby leading to an easier subproblem.

Here we mention two commonly used choices of P_i:

- $P_i = \tau_i \mathbf{I}$ $(\tau_i > 0)$: This corresponds to the standard *proximal method.*
- $P_i = \tau_i \mathbf{I} - \rho A_i^\top A_i$ $(\tau_i > 0)$: This corresponds to the *prox-linear method* [6], which linearizes the quadratic penalty term of augmented Lagrangian at the current point $\mathbf{x}_i^k$ and adds a proximal term. More specifically, the prox-linear $\mathbf{x}_i$-subproblem is given by

  (1.13)
  $$\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} \; f_i(\mathbf{x}_i) + \left\langle \rho A_i^\top (A\mathbf{x}^k - c - \lambda^k/\rho), \mathbf{x}_i \right\rangle + \frac{\tau_i}{2}\left\|\mathbf{x}_i - \mathbf{x}_i^k\right\|^2.$$

  It essentially uses an identity matrix $\tau_i \mathbf{I}$ to approximate the Hessian matrix $\rho A_i^\top A_i$ of the quadratic penalty term.

More choices of P_i have also been discussed in [33, 8].

1.3. Summary of Contributions. This paper introduces a few novel results from different perspectives. Firstly, we propose *Proximal Jacobian ADMM*, which is suitable for parallel and distributed computing. The use of flexible proximal terms make it possible to solve its subproblems in different ways, important for easy coding and fast computation. We establish its convergence at a rate of $o(1/k)$. Our numerical results on the exchange problem and ℓ_1-minimization problem show that the proposed algorithm achieves competitive performance, in comparison with several existing parallel algorithms.

The second contribution is a sufficient condition of convergence for the extension of classic ADMM with a Jacobian update scheme.

The third contribution is the improvement of the established convergence rate of $O(1/k)$ for the standard ADMM to $o(1/k)$ by a simple proof. This technique can be also applied to various other algorithms, and some existing convergence rates of $O(1/k)$ can be slightly improved to $o(1/k)$ as well.

1.4. Preliminary, Notation, and Assumptions. To simplify the notation in this paper, we introduce

$$\mathbf{x} := \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{pmatrix} \in \mathbb{R}^n, \ A := \begin{pmatrix} A_1, \ldots, A_N \end{pmatrix} \in \mathbb{R}^{m\times n}, \ \mathbf{u} := \begin{pmatrix} \mathbf{x} \\ \lambda \end{pmatrix} \in \mathbb{R}^{n+m},$$

where

$$n = \sum_{i=1}^{N} n_i.$$

We let $\langle \cdot, \cdot \rangle$ and $\|\cdot\|$ denote the standard inner product and ℓ_2-norm $\|\cdot\|_2$, respectively, in the Euclidean space. For a matrix $M \in \mathbb{R}^{l\times l}$, $\|M\|$ denotes the spectral norm, i.e., the largest singular value of M. For a positive definite matrix $G \in \mathbb{R}^{l\times l}$, we define the *G-norm* as follows:

$$\|\mathbf{z}\|_G := \sqrt{\mathbf{z}^\top G\mathbf{z}}, \ \forall \mathbf{z} \in \mathbb{R}^l. \tag{1.14}$$

If the matrix G is positive semi-definite, then $\|\cdot\|_G$ is a semi-norm.

Throughout the paper, we make the following standard assumptions.

Assumption 1. *Functions $f_i : \mathbb{R}^{n_i} \to (-\infty, +\infty]$ $(i = 1, 2, \ldots, N)$ are closed proper convex.*

Assumption 2. *There exists a saddle point $\mathbf{u}^* = (\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_N^*, \lambda^*)$ to the* *problem* (1.1). *Namely,* $\mathbf{u}^*$ *satisfies the KKT conditions:*

$$A_i^\top \lambda^* \in \partial f_i(\mathbf{x}_i^*), \text{ for } i = 1, \ldots, N, \tag{1.15}$$

$$A\mathbf{x}^* = \sum_{i=1}^{N} A_i \mathbf{x}_i^* = c. \tag{1.16}$$

The optimality conditions (1.15) *and* (1.16) *can be written in a more compact form by the following variational inequality [16]:*

$$f(\mathbf{x}) - f(\mathbf{x}^*) + (\mathbf{u} - \mathbf{u}^*)^\top F(\mathbf{u}^*) \geq 0, \ \forall u, \tag{1.17}$$

where $f(\mathbf{x}) := \sum_i f_i(\mathbf{x}_i)$ *and*

$$F(\mathbf{u}) := \begin{pmatrix} -A_1^\top \lambda \\ \vdots \\ -A_N^\top \lambda \\ A\mathbf{x} - c \end{pmatrix}.$$

Let $\partial f_i(\mathbf{x}_i)$ denote the subdifferential of f_i at $\mathbf{x}_i$:

$$\partial f_i(\mathbf{x}_i) := \left\{ s_i \in \mathbb{R}^{n_i} : \ s_i^\top (\mathbf{y}_i - \mathbf{x}_i) \leq f_i(\mathbf{y}_i) - f_i(\mathbf{x}_i), \ \forall \mathbf{y}_i \in \mathrm{dom} f_i \right\}. \tag{1.18}$$

We recall the following basic property for convex functions, which will be used several times in later sections.

Lemma 1.1 (monotonicity of subdifferential). *Under Assumption 1, for any* $\mathbf{x}_i, \mathbf{y}_i \in \mathrm{dom}\ f_i$*, we have*

$$(s_i - t_i)^\top (\mathbf{x}_i - \mathbf{y}_i) \geq 0, \ \forall s_i \in \partial f_i(\mathbf{x}_i), \ t_i \in \partial f_i(\mathbf{y}_i), \tag{1.19}$$

for $i = 1, 2, \ldots, N$.

Proof. By definition, for any $s_i \in \partial f_i(\mathbf{x}_i)$ and $t_i \in \partial f_i(\mathbf{y}_i)$, we have

$$\begin{aligned} s_i^\top (\mathbf{y}_i - \mathbf{x}_i) &\leq f_i(\mathbf{y}_i) - f_i(\mathbf{x}_i), \\ t_i^\top (\mathbf{x}_i - \mathbf{y}_i) &\leq f_i(\mathbf{x}_i) - f_i(\mathbf{y}_i). \end{aligned}$$

Adding these two inequalities together yields (1.19). □

In addition, we shall use an elementary lemma to improve the convergence rate from $O(1/k)$ to $o(1/k)$. Intuitively, the harmonic sequence $1/k$ is not summable, so a summable, nonnegative, monotonic sequence shall converge faster than $1/k$.

Lemma 1.2. *If a sequence $\{a_k\} \subseteq \mathbb{R}$ obeys: (1) $a_k \geq 0$; (2) $\sum_{k=1}^{\infty} a_k < +\infty$; (3) a_k is monotonically non-increasing, then we have $a_k = o(1/k)$.*

Proof. By the assumptions, we have

$$k \cdot a_{2k} \leq a_{k+1} + a_{k+2} + \cdots + a_{2k} \to 0$$

as $k \to +\infty$. Therefore, $a_k = o(1/k)$. □

2. Convergence Analysis of the Proximal Jacobian ADMM.

In this section, we mainly study the convergence of Proximal Jacobian ADMM (Algorithm 4). We first show its convergence and then establish an $o(1/k)$ convergence rate in the same sense as in [18]. Furthermore, we discuss how to tune the parameter in order to make Proximal Jacobian ADMM more practical.

2.1. Convergence.

To simplify the notation, we let

$$G_x := \begin{pmatrix} P_1 + \rho A_1^\top A_1 & & \\ & \ddots & \\ & & P_N + \rho A_N^\top A_N \end{pmatrix}, \; G := \begin{pmatrix} G_x & \\ & \frac{1}{\gamma\rho}\mathbf{I} \end{pmatrix},$$

where $\mathbf{I}$ is the identity matrix of size $m \times m$. In the rest of the section, we let $\{\mathbf{u}^k\}$ denote the sequence generated by Proximal Jacobian ADMM from any initial point. The analysis is based on bounding the error $\|\mathbf{u}^k - \mathbf{u}^*\|_G^2$ and estimating its decrease, motivated by the works [19, 8, 16].

Lemma 2.1. *For $k \geq 1$, we have*

$$\|\mathbf{u}^k - \mathbf{u}^*\|_G^2 - \|\mathbf{u}^{k+1} - \mathbf{u}^*\|_G^2 \geq h(\mathbf{u}^k,\ \mathbf{u}^{k+1}), \tag{2.1}$$

where

$$h(\mathbf{u}^k,\ \mathbf{u}^{k+1}) := \|\mathbf{x}^k - \mathbf{x}^{k+1}\|_{G_x}^2 + \frac{2-\gamma}{\rho\gamma^2}\|\lambda^k - \lambda^{k+1}\|^2 + \frac{2}{\gamma}(\lambda^k - \lambda^{k+1})^\top A(\mathbf{x}^k - \mathbf{x}^{k+1}). \tag{2.2}$$

Proof. Recall that in Algorithm 4, we solve the following $\mathbf{x}_i$-subproblem:

$$\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \frac{\rho}{2}\left\| A_i\mathbf{x}_i + \sum_{j\neq i} A_j\mathbf{x}_j^k - c - \frac{\lambda^k}{\rho}\right\|^2 + \frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_i^k\|_{P_i}^2.$$

Its optimality condition is given by

$$A_i^\top\left(\lambda^k - \rho(A_i\mathbf{x}_i^{k+1} + \sum_{j\neq i} A_j\mathbf{x}_j^k - c)\right) + P_i(\mathbf{x}_i^k - \mathbf{x}_i^{k+1}) \in \partial f_i(\mathbf{x}_i^{k+1}). \tag{2.3}$$

For convenience, we introduce $\hat{\lambda} := \lambda^k - \rho(A\mathbf{x}^{k+1} - c)$. Then (2.3) can be rewritten as

$$A_i^\top\left(\hat{\lambda} - \rho\sum_{j\neq i} A_j(\mathbf{x}_j^k - \mathbf{x}_j^{k+1})\right) + P_i(\mathbf{x}_i^k - \mathbf{x}_i^{k+1}) \in \partial f_i(\mathbf{x}_i^{k+1}). \tag{2.4}$$

By Lemma 1.1, it follows from (1.15) and (2.4) that

$$\left\langle A_i(\mathbf{x}_i^{k+1} - \mathbf{x}_i^*),\ \hat{\lambda} - \lambda^* - \rho\sum_{j\neq i} A_j(\mathbf{x}_j^k - \mathbf{x}_j^{k+1})\right\rangle + (\mathbf{x}_i^{k+1} - \mathbf{x}_i^*)^\top P_i(\mathbf{x}_i^k - \mathbf{x}_i^{k+1}) \geq 0.$$

Summing the above inequality over all i and using the following equality for each i:

$$\sum_{j\neq i} A_j(\mathbf{x}_j^k - \mathbf{x}_j^{k+1}) = A(\mathbf{x}^k - \mathbf{x}^{k+1}) - A_i(\mathbf{x}_i^k - \mathbf{x}_i^{k+1}),$$

we obtain

$$\begin{aligned}&\langle A(\mathbf{x}^{k+1} - \mathbf{x}^*), \hat{\lambda} - \lambda^*\rangle + \sum_{i=1}^N (\mathbf{x}_i^{k+1} - \mathbf{x}_i^*)^\top (P_i + \rho A_i^\top A_i)(\mathbf{x}_i^k - \mathbf{x}_i^{k+1})\\ &\geq \rho\langle A(\mathbf{x}^{k+1} - \mathbf{x}^*), A(\mathbf{x}^k - \mathbf{x}^{k+1})\rangle.\end{aligned} \tag{2.5}$$

Note that

$$A(\mathbf{x}^{k+1} - \mathbf{x}^*) = \frac{1}{\gamma\rho}(\lambda^k - \lambda^{k+1}),$$

and

$$\hat{\lambda} - \lambda^* = (\hat{\lambda} - \lambda^{k+1}) + (\lambda^{k+1} - \lambda^*) = \frac{\gamma - 1}{\gamma}(\lambda^k - \lambda^{k+1}) + (\lambda^{k+1} - \lambda^*).$$

With the above two equations, the inequality (2.5) can be rewritten as

$$\begin{aligned} &\langle \frac{1}{\gamma\rho}(\lambda^k-\lambda^{k+1}), \lambda^{k+1}-\lambda^*\rangle + \sum_{i=1}^N (\mathbf{x}_i^{k+1}-\mathbf{x}_i^*)^\top (P_i+\rho A_i^\top A_i)(\mathbf{x}_i^k-\mathbf{x}_i^{k+1}) \\ &\geq \frac{1-\gamma}{\gamma^2\rho}\|\lambda^k-\lambda^{k+1}\|^2 + \frac{1}{\gamma}(\lambda^k-\lambda^{k+1})^\top A(\mathbf{x}^k-\mathbf{x}^{k+1}), \end{aligned} \tag{2.6}$$

i.e.,

$$(\mathbf{u}^k-\mathbf{u}^{k+1})^\top G(\mathbf{u}^{k+1}-\mathbf{u}^*) \geq \frac{1-\gamma}{\gamma^2\rho}\|\lambda^k-\lambda^{k+1}\|^2+\frac{1}{\gamma}(\lambda^k-\lambda^{k+1})^\top A(\mathbf{x}^k-\mathbf{x}^{k+1}). \tag{2.7}$$

Since $\|\mathbf{u}^k-\mathbf{u}^*\|_G^2 - \|\mathbf{u}^{k+1}-\mathbf{u}^*\|_G^2 = 2(\mathbf{u}^k-\mathbf{u}^{k+1})^\top G(\mathbf{u}^{k+1}-\mathbf{u}^*) + \|\mathbf{u}^k-\mathbf{u}^{k+1}\|_G^2$, using the above inequality (2.7) yields (2.1) immediately. $\square$

LEMMA 2.2. *Suppose the parameters ρ, γ and P_i $(i=1,2,\ldots,N)$ satisfy the following condition:*

$$\begin{cases} P_i \succ \rho(\frac{1}{\epsilon_i}-1)A_i^\top A_i, \;\; i=1,2,\ldots,N \\ \sum_{i=1}^N \epsilon_i < 2-\gamma, \end{cases} \tag{2.8}$$

for some $\epsilon_i > 0$, $i=1,2,\ldots,N$. Then there exists some $\eta>0$ such that

$$h(\mathbf{u}^k,\mathbf{u}^{k+1}) \geq \eta\cdot\|\mathbf{u}^k-\mathbf{u}^{k+1}\|_G^2. \tag{2.9}$$

Therefore,

$$\|\mathbf{u}^k-\mathbf{u}^*\|_G^2 - \|\mathbf{u}^{k+1}-\mathbf{u}^*\|_G^2 \geq \eta\cdot\|\mathbf{u}^k-\mathbf{u}^{k+1}\|_G^2. \tag{2.10}$$

Condition (2.8) can be reduced to

$$P_i \succ \rho\left(\frac{N}{2-\gamma}-1\right)A_i^\top A_i, \;\; i=1,2,\ldots,N, \tag{2.11}$$

by letting each $\epsilon_i < \frac{2-\gamma}{N}$. In particular, for the following choices:

- $P_i = \tau_i\mathbf{I}$ *(standard proximal), condition (2.11) becomes $\tau_i > \rho\left(\frac{N}{2-\gamma}-1\right)\|A_i\|^2$;*
- $P_i = \tau_i\mathbf{I} - \rho A_i^\top A_i$ *(prox-linear), condition (2.11) becomes $\tau_i > \frac{\rho N}{2-\gamma}\|A_i\|^2$.*

Proof. By the Cauchy-Schwarz inequality,

$$\begin{aligned}\frac{2}{\gamma}(\lambda^k-\lambda^{k+1})^\top A(\mathbf{x}^k-\mathbf{x}^{k+1}) &= \sum_{i=1}^N \frac{2}{\gamma}(\lambda^k-\lambda^{k+1})^\top A_i(\mathbf{x}_i^k-\mathbf{x}_i^{k+1}) \\ &\geq -\sum_{i=1}^N\left(\frac{\epsilon_i}{\rho\gamma^2}\|\lambda^k-\lambda^{k+1}\|^2+\frac{\rho}{\epsilon_i}\|A_i(\mathbf{x}_i^k-\mathbf{x}_i^{k+1})\|^2\right) \qquad (2.12)\end{aligned}$$

Then we have

$$h(\mathbf{u}^k,\mathbf{u}^{k+1}) \geq \sum_{i=1}^N \|\mathbf{x}_i^k-\mathbf{x}_i^{k+1}\|^2_{P_i+\rho A_i^\top A_i-\frac{\rho}{\epsilon_i}A_i^\top A_i} + \frac{2-\gamma-\sum_{i=1}^N\epsilon_i}{\rho\gamma^2}\|\lambda^k-\lambda^{k+1}\|^2. \qquad (2.13)$$

The condition (2.8) guarantees that $P_i+\rho A_i^\top A_i-\frac{\rho}{\epsilon_i}A_i^\top A_i \succ 0$ and $2-\gamma-\sum_{i=1}^N\epsilon_i>0$. Therefore, we must have (2.9) for some $\eta>0$. By Lemma 2.1, (2.10) follows immediately. □

Lemma 2.2 shows that the iterative sequence $\{\mathbf{u}^k\}$ is *strictly contractive*. It follows that the error $\|\mathbf{u}^k-\mathbf{u}^*\|_G^2$ is monotonically non-increasing and thus converging, as well as $\|\mathbf{u}^k-\mathbf{u}^{k+1}\|_G^2\to 0$. The convergence of the algorithm follows immediately from the standard analysis for contraction methods (see, e.g., [14]). We omit the details of the proof for the sake of brevity.

THEOREM 2.3. *Suppose the parameters in Algorithm 4 satisfy the condition* (2.8). *Then the sequence* $\{\mathbf{u}^k\}$ *generated by Algorithm 4 converges to a solution* $\mathbf{u}^*$ *to the problem* (1.1).

2.2. Rate of Convergence. Next, we shall establish the $o(1/k)$ convergence rate of Proximal Jacobian ADMM. We use the quantity $\|\mathbf{u}^k-\mathbf{u}^{k+1}\|^2_{G'}$ as a measure of the convergence rate motivated by [18, 16]. Here, we define the matrix G' by

$$G' := \begin{pmatrix} G'_x & \\ & \frac{1}{\gamma\rho}\mathbf{I}\end{pmatrix} \quad\text{and}\quad G'_x := G_x-\rho A^\top A.$$

THEOREM 2.4. *If* $G'_x\succeq 0$ *and* (2.8) *holds, then*

$$\|\mathbf{u}^k-\mathbf{u}^{k+1}\|^2_{G'}=o(1/k),$$

and, thus,

$$\|\mathbf{x}^k-\mathbf{x}^{k+1}\|^2_{G'_x}=o(1/k) \quad \textit{and} \quad \|\lambda^k-\lambda^{k+1}\|^2=o(1/k).$$

We need the following monotonic property of the iterations:

LEMMA 2.5. *If $G'_x \succeq 0$ and $0 < \gamma < 2$, then*

$$\|\mathbf{u}^k - \mathbf{u}^{k+1}\|^2_{G'} \leq \|\mathbf{u}^{k-1} - \mathbf{u}^k\|^2_{G'}. \tag{2.14}$$

Proof. Let $\Delta\mathbf{x}_i^{k+1} = \mathbf{x}_i^k - \mathbf{x}_i^{k+1}, i = 1, \ldots, N$, $\Delta\mathbf{x}^{k+1} = \mathbf{x}^k - \mathbf{x}^{k+1}$, and $\Delta\lambda^{k+1} = \lambda^k - \lambda^{k+1}$. By Lemma 1.1, the optimality conditions (2.4) at k-th and $(k+1)$-th iterations yield

$$\langle A_i\Delta\mathbf{x}_i^{k+1}, \Delta\lambda^k - \rho A\Delta\mathbf{x}^{k+1} - \rho\sum_{j\neq i} A_j(\Delta\mathbf{x}_j^k - \Delta\mathbf{x}_j^{k+1})\rangle + (\Delta\mathbf{x}_i^{k+1})^\top P_i(\Delta\mathbf{x}_i^k - \Delta\mathbf{x}_i^{k+1}) \geq 0. \tag{2.15}$$

Summing up over all i and rearranging the terms, we have

$$\langle A\Delta\mathbf{x}^{k+1}, \Delta\lambda^k\rangle \geq \|\Delta\mathbf{x}^{k+1}\|^2_{G_x} - (\Delta\mathbf{x}^k)^\top(G_x - \rho A^\top A)\Delta\mathbf{x}^{k+1}. \tag{2.16}$$

Since $G'_x := G_x - \rho A^\top A \succeq 0$, we have

$$2(\Delta\mathbf{x}^k)^\top(G_x - \rho A^\top A)\Delta\mathbf{x}^{k+1} \leq \|\Delta\mathbf{x}^k\|^2_{G'_x} + \|\Delta\mathbf{x}^k\|^2_{G'_x}, \tag{2.17}$$

and thus

$$\begin{aligned} 2\langle A\Delta\mathbf{x}^{k+1}, \Delta\lambda^k\rangle &\geq \|\Delta\mathbf{x}^{k+1}\|^2_{2G_x - G'_x} - \|\Delta\mathbf{x}^k\|^2_{G'_x} \\ &= \|\Delta\mathbf{x}^{k+1}\|^2_{G_x + \rho A^\top A} - \|\Delta\mathbf{x}^k\|^2_{G'_x}. \end{aligned} \tag{2.18}$$

Note that $\Delta\lambda^{k+1} = \Delta\lambda^k - \gamma\rho A\Delta\mathbf{x}^{k+1}$. It follows that

$$\begin{aligned} \frac{1}{\gamma\rho}\|\Delta\lambda^k\|^2 - \frac{1}{\gamma\rho}\|\Delta\lambda^{k+1}\|^2 &= 2\langle A\Delta\mathbf{x}^{k+1}, \Delta\lambda^k\rangle - \gamma\rho\|A\Delta\mathbf{x}^{k+1}\|^2 \\ &\geq \|\Delta\mathbf{x}^{k+1}\|^2_{G_x + (1-\gamma)\rho A^\top A} - \|\Delta\mathbf{x}^k\|^2_{G'_x}, \end{aligned} \tag{2.19}$$

i.e.,

$$\begin{aligned} &(\|\Delta\mathbf{x}^k\|^2_{G'_x} + \frac{1}{\gamma\rho}\|\Delta\lambda^k\|^2) - (\|\Delta\mathbf{x}^{k+1}\|_{G'_x} + \frac{1}{\gamma\rho}\|\Delta\lambda^{k+1}\|^2) \\ &\geq \|\Delta\mathbf{x}^{k+1}\|^2_{(2-\gamma)\rho A^\top A} \geq 0, \end{aligned} \tag{2.20}$$

which completes the proof. □

Proof. [Proof of Theorem 2.4] By Lemma 2.2, we have

$$\|\mathbf{u}^k-\mathbf{u}^*\|_G^2-\|\mathbf{u}^{k+1}-\mathbf{u}^*\|_G^2 \ge \eta\|\mathbf{u}^k-\mathbf{u}^{k+1}\|_G^2 \ge \eta\|\mathbf{u}^k-\mathbf{u}^{k+1}\|_{G'}^2. \tag{2.21}$$

Summing (2.21) over k gives

$$\sum_{k=1}^{\infty}\|\mathbf{u}^k-\mathbf{u}^{k+1}\|_{G'}^2<\infty. \tag{2.22}$$

On the other hand, Lemma 2.5 implies the monotone non-increasing of $\|\mathbf{u}^k-\mathbf{u}^{k+1}\|_{G'}^2$. By Lemma 1.2, we have $\|\mathbf{u}^k-\mathbf{u}^{k+1}\|_{G'}^2=o(1/k)$, which completes the proof. □

2.3. Adaptive Parameter Tuning. The parameters satisfying the condition (2.8) may be rather conservative because the Cauchy-Schwarz inequality (2.12) for bounding $h(\mathbf{u}^k,\mathbf{u}^{k+1})$ is usually very loose. In practice, we can exactly compute $h(\mathbf{u}^k,\mathbf{u}^{k+1})$ at very little extra cost, instead of using the bound in (2.13). Based on the value of $h(\mathbf{u}^k,\mathbf{u}^{k+1})$, we thereby propose a practical strategy for adaptively adjusting the matrices $\{P_i\}$:

```
Initialize with small $P_i^0 \succeq 0$ $(i=1,2,\dots,N)$ and a small $\eta>0$;
for $k=1,2,\dots$ do
    if $h(\mathbf{u}^{k-1},\mathbf{u}^k) > \eta\cdot\|\mathbf{u}^{k-1}-\mathbf{u}^k\|_G^2$ then
        $P_i^{k+1} \leftarrow P_i^k,\ \forall i$;
    else
        Increase $P_i$: $P_i^{k+1} \leftarrow \alpha_i P_i^k+\beta_i Q_i$ $(\alpha_i>1,\ \beta_i\ge 0,\ Q_i\succ 0), \forall i$;
        Restart: $\mathbf{u}^k \leftarrow \mathbf{u}^{k-1}$;
```

The above strategy starts with relatively small proximal terms and gradually increase them. By Lemma 2.2, we know that when the parameters $\{P_i\}$ are large enough for (2.8) to hold, the condition (2.9) will be satisfied (for sufficiently small η). Therefore, the adjustment of $\{P_i\}$ cannot occur infinite times. After a finite number of iterations, $\{P_i\}$ will remain constant and the contraction property (2.10) of the iterations will hold. Therefore, the convergence of such an adaptive parameter tuning scheme follows immediately from our previous analysis.

Theorem 2.6. *Suppose the matrices P_i $(i=1,2,\dots,N)$ in Algorithm 4 are adaptively adjusted using the above scheme. Then the algorithm converges to a solution to the problem* (1.1).

Empirical evidence shows that the paramters $\{P_i\}$ typically adjust themselves only during the first few iterations and then remain constant afterwards. Alternatively, one may also decrease the parameters after every few iterations or after they have not been updated for a certain number of iterations. But the total times of decrease should be bounded to guarantee convergence. By using this adaptive strategy, the resulting paramters $\{P_i\}$ are usually much smaller than those required by the condition (2.8), thereby leading to substantially faster convergence in practice.

3. Numerical Experiments.

In this section, we present numerical results to compare the performance of the following parallel splitting algorithms:

- **Prox-JADMM**: proposed Proximal Jacobian ADMM (Algorithm 4);
- **VSADMM**: Variable Splitting ADMM (Algorithm 1);
- **Corr-JADMM**: Jacobian ADMM with correction steps [16]. At every iteration, it first generates a "predictor" $\tilde{\mathbf{u}}^{k+1}$ by an iteration of Jacobian ADMM (Algorithm 3) and then corrects $\tilde{\mathbf{u}}^{k+1}$ to generate the new iterate by:

  $$\mathbf{u}^{k+1} = \mathbf{u}^k - \alpha_k(\mathbf{u}^k - \tilde{\mathbf{u}}^{k+1}), \tag{3.1}$$

  where $\alpha_k > 0$ is a step size. In our experiments, we adopt the dynamically updated step size α_k according to [16], which is shown to converge significantly faster than using a constant step size, though updating the step size requires extra computation.
- **YALL1**: one of the state-of-the-art solvers for the ℓ_1-minimization problem.

In Section 3.1 and 3.2, all of the numerical experiments are run in MATLAB (R2011b) on a workstation with an Intel Core i5-3570 CPUs (3.40GHz) and 32 GB of RAM. Section 3.3 gives two very large instances that are solved by a C/MPI implementation on Amazon Elastic Compute Cloud (EC2).

3.1. Exchange Problem.

Consider a network of N agents that exchange n commodities. Let $\mathbf{x}_i \in \mathbb{R}^n$ $(i = 1, 2, \ldots, N)$ denote the amount of commodities that are exchanged among the N agents. Each agent i has a certain cost function $f_i : \mathbb{R}^n \to \mathbb{R}$. The exchange problem (see, e.g., [3] for a review) is given by

$$\min_{\{\mathbf{x}_i\}} \sum_{i=1}^{N} f_i(\mathbf{x}_i) \quad \text{s.t.} \sum_{i=1}^{N} \mathbf{x}_i = 0, \tag{3.2}$$

which minimizes the total cost among N agents subject to an equilibrium constraint on the commodities. This is a special case of (1.1) where $A_i = \mathbf{I}$ and $c = 0$.

We consider quadratic cost functions $f_i(\mathbf{x}_i) := \frac{1}{2}\|C_i\mathbf{x}_i - d_i\|^2$, where $C_i \in \mathbb{R}^{p\times n}$ and $d_i \in \mathbb{R}^p$. Then all the compared algorithms solve the following type of subproblems at every iteration:

$$\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} \ \frac{1}{2}\|C_i\mathbf{x}_i - d_i\|^2 + \frac{\rho}{2}\|\mathbf{x}_i - b_i^k\|^2, \ \forall i = 1,2,\dots,N, \tag{3.3}$$

except that Prox-JADMM also adds a proximal term $\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_i^k\|_{P_i}^2$. Here $b_i^k \in \mathbb{R}^m$ is a vector independent of $\mathbf{x}_i$ and takes different forms in different algorithms. For Prox-JADMM, we simply set $P_i = \tau_i\mathbf{I}$ ($\tau_i > 0$). Clearly, each $\mathbf{x}_i$-subproblem is a quadratic program that can be computed efficiently using various methods.

In our experiment, we randomly generate $\mathbf{x}_i^*$, $i = 1,2,\dots,N-1$, following the standard Gaussian distribution, and let $\mathbf{x}_N^* = -\sum_{i=1}^{N-1}\mathbf{x}_i^*$. Matrices C_i are random Gaussian matrices, and vectors d_i are computed by $d_i = C_i\mathbf{x}_i^*$. Apparently, $\mathbf{x}^*$ is a solution (not necessarily unique) to (3.2), and the optimal objective value is 0.

The penalty parameter ρ is set to be 0.01, 1 and 0.01 for Prox-JADMM, VSADMM and Corr-JADMM, respectively. They are nearly optimal for each algorithm, picked out of a number of different values. Note that the parameter for VSADMM is quite different from the other two algorithms because it has different constraints due to the variable splitting. For Prox-JADMM, the proximal parameters are initialized by $\tau_i = 0.1(N-1)\rho$ and adaptively updated by the strategy in Subsection 2.3; the parameter γ is set to be 1.

The size of the test problem is set to be $n = 100$, $N = 100$, $p = 80$. Letting all the algorithms run 200 iterations, we plot their objective value $\sum_{i=1}^N f_i(\mathbf{x}_i)$ and residual $\|\sum_{i=1}^N \mathbf{x}_i\|_2$. Note that the per-iteration cost (in terms of both computation and communication) is roughly the same for all the compared algorithms. Figure 3.1 shows the comparison result, which is averaged over 100 random trials. We can see that Prox-JADMM is clearly the fastest one among the compared algorithm.

3.2. ℓ_1-minimization. We consider the ℓ_1-minimization problem for finding sparse solutions of an underdetermined linear system:

$$\min_{\mathbf{x}} \ \|\mathbf{x}\|_1 \quad \text{s.t. } A\mathbf{x} = c, \tag{3.4}$$

where $\mathbf{x} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$ and $c \in \mathbb{R}^m$ ($m < n$). It is also known as the *basis pursuit* problem, which has been widely used in compressive sensing, signal and image pro-

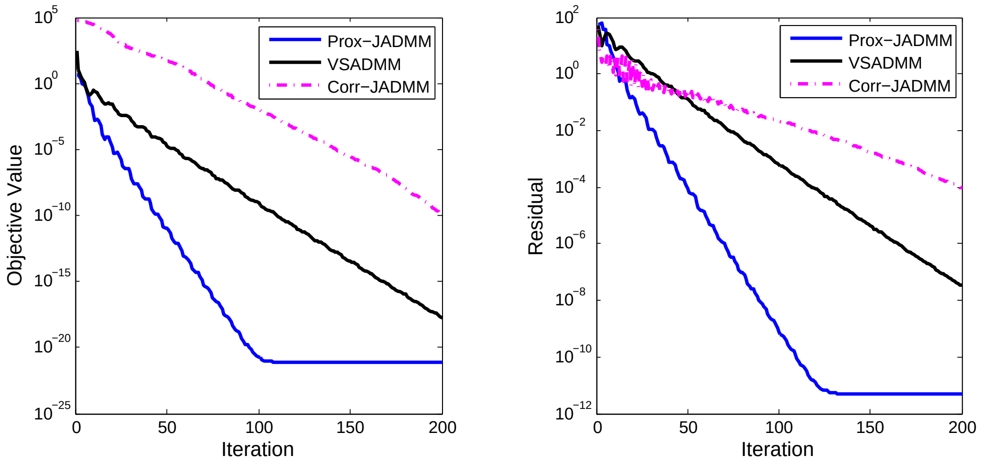


FIG. 3.1. *Exchange problem* ($n = 100,\ N = 100,\ p = 80$).

cessing, statistics, and machine learning. Suppose that the data is partitioned into N blocks: $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$ and $A = [A_1, A_2, \ldots, A_N]$. Then the problem (3.4) can be written in the form of (1.1) with $f_i(\mathbf{x}_i) = \|\mathbf{x}_i\|_1$.

In our experiment, a sparse solution $\mathbf{x}^*$ is randomly generated with k ($k \ll n$) nonzeros drawn from the standard Gaussian distribution. Matrix A is also randomly generated from the standard Gaussian distribution, and it is partitioned evenly into N blocks. The vector c is then computed by $c = A\mathbf{x}^* + \eta$, where $\eta \sim \mathcal{N}(0,\ \sigma^2\mathbf{I})$ is Gaussian noise with standard deviation σ.

Prox-JADMM solves the $\mathbf{x}_i$-subproblems with $P_i = \tau_i \mathbf{I} - \rho A_i^\top A_i$ ($i = 1, 2, \ldots, N$) as follows:

$$\begin{aligned} \mathbf{x}_i^{k+1} &= \arg\min_{\mathbf{x}_i}\ \|\mathbf{x}_i\|_1 + \frac{\rho}{2}\left\| A_i\mathbf{x}_i + \sum_{j\neq i} A_j\mathbf{x}_j^k - c - \frac{\lambda^k}{\rho}\right\|^2 + \frac{1}{2}\left\|\mathbf{x}_i - \mathbf{x}_i^k\right\|_{P_i}^2 \\ &= \arg\min_{\mathbf{x}_i}\ \|\mathbf{x}_i\|_1 + \left\langle \rho A_i^\top \left(A\mathbf{x}^k - c - \frac{\lambda^k}{\rho}\right), \mathbf{x}_i \right\rangle + \frac{\tau_i}{2}\left\|\mathbf{x}_i - \mathbf{x}_i^k\right\|^2 . \end{aligned} \tag{3.5}$$

Here, we choose the prox-linear P_i's to linearize the original subproblems, and thus (3.5) admits a simple closed-form solution by the *shrinkage* (or *soft-thresholding*) formula. The proximal parameters are initialized as $\tau_i = 0.1N\rho$ and are adaptively updated by the strategy discussed in Section 2.3.

Recall that VSADMM needs to solve the following $\mathbf{x}_i$-subproblems:

$$\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} \ \|\mathbf{x}_i\|_1 + \frac{\rho}{2}\left\|A_i\mathbf{x}_i - \mathbf{z}_i^{k+1} - \frac{c}{N} - \frac{\lambda_i^k}{\rho}\right\|^2. \tag{3.6}$$

Such subproblems are not easily computable, unless $\mathbf{x}_i$ is a scalar (i.e., $n_i = 1$) or $A_i^\top A_i$ is a diagonal matrix. Instead, we solve the subproblems approximately using the prox-linear approach:

$$\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} \ \|\mathbf{x}_i\|_1 + \left\langle \rho A_i^\top \left(A_i\mathbf{x}_i^k - \mathbf{z}_i^{k+1} - \frac{c}{N} - \frac{\lambda_i^k}{\rho}\right), \mathbf{x}_i - \mathbf{x}_i^k \right\rangle + \frac{\tau_i}{2}\left\|\mathbf{x}_i - \mathbf{x}_i^k\right\|^2, \tag{3.7}$$

which can be easily computed by the shrinkage operator. We set $\tau_i = 1.01\rho\|A_i\|^2$ in order to guarantee the convergence, as suggested in [30].

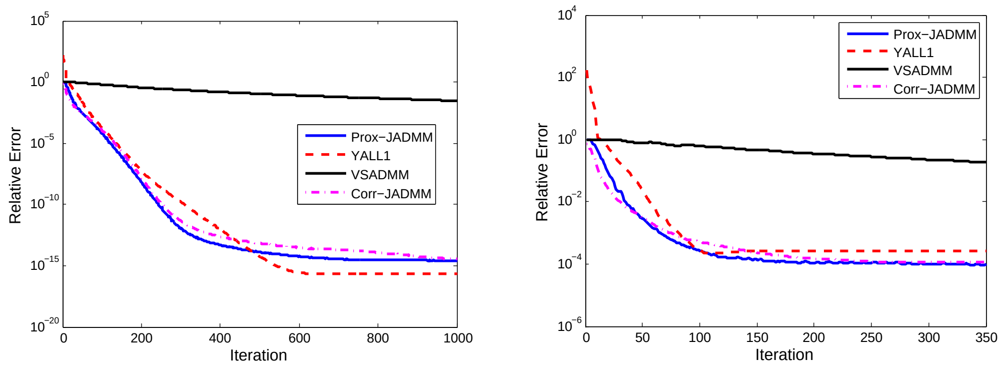
Corr-JADMM solves the following $\mathbf{x}_i$-subproblems in the "prediction" step:

$$\tilde{\mathbf{x}}_i^{k+1} = \arg\min_{\mathbf{x}_i} \ \|\mathbf{x}_i\|_1 + \frac{\rho}{2}\left\|A_i\mathbf{x}_i + \sum_{j\neq i} A_j\mathbf{x}_j^k - c - \frac{\lambda^k}{\rho}\right\|^2. \tag{3.8}$$

Because the correction step in [16] is based on exact minimization of the subproblems, we do not apply the prox-linear approach to solve the subproblems approximately. Instead, we always partition $\mathbf{x}$ into scalar components (i.e., $N = n$) so that the subproblems (3.8) can still be computed exactly. The same penalty parameter $\rho = 10/\|c\|_1$ is used for the three algorithms. It is nearly optimal for each algorithm, selected out of a number of different values.

In addition, we also include the YALL1 package [32] in the experiment, which is one of the state-of-the-art solvers for ℓ_1 minimization. Though YALL1 is not implemented in parallel, the major computation of its iteration is matrix-vector multiplication by A and $A^\top$, which can be easily parallelized (see [25]). Since all the compared algorithms have roughly the same amount of per-iteration cost (in terms of both computation and communication), we simply let all the algorithms run for a fixed number of iterations and plot their relative error $\frac{\|\mathbf{x}^k - \mathbf{x}^*\|_2}{\|\mathbf{x}^*\|_2}$.

Figure 3.2 shows the comparison result where $n = 1000$, $m = 300$, $k = 60$ and the standard deviation of noise σ is set to be 0 and 10^{-3}, respectively. For Prox-JADMM and VSADMM, we set $N = 100$; for Corr-JADMM, we set $N = 1000$. The results are average of 100 random trials. We can see that Prox-JADMM and Corr-JADMM achieve very close performance and are the fastest ones among



(a) Noise-free ($\sigma = 0$) (b) Noise added ($\sigma = 10^{-3}$)

FIG. 3.2. *ℓ_1-problem* ($n = 1000,\ m = 300,\ k = 60$).

the compared algorithms. YALL1 also shows competitive performance. However, VSADMM is far slower than the others, probably due to inexact minimization of the subproblems and the conservative proximal parameters.

3.3. Distributed Large-Scale ℓ_1-Minimization. In previous subsections, we described the numerical simulation of a distributed implementation of Proximal Jacobian ADMM that was carried out in Matlab. We now turn to realistic distributed examples and solve two very large instances of the ℓ_1-minimization problem (3.4) using a C code with MPI for inter-process communication and the GNU Scientific Library (GSL) for BLAS operations. The experiments are carried out on Amazon's Elastic Compute Cloud (EC2).

We generate two test instances as shown in Table 3.1. Specifically, a sparse solution $\mathbf{x}^*$ is randomly generated with k nonzeros drawn from the standard Gaussian distribution. Matrix A is also randomly generated from the standard Gaussian distribution with m rows and n columns, and it is partitioned evenly into $N = 80$ blocks. Vector c is then computed by $c = A\mathbf{x}^*$. Note that A is dense and has double precision. For Test 1 it requires over 150 GB of RAM and has 20 billion nonzero entries, and for Test 2 it requires over 337GB of RAM. Those two tests are far too large to process on a single PC or workstation. We want to point out that we cannot find a dataset of similar or larger size in the public domain. We are willing to test our a larger problem per reader's request.

We solve the problem using a cluster of 10 machines, where each machine is a "memory-optimized instance" with 68 GB RAM and 1 eight-core Intel Xeon E5-2665 CPU. Those instances run Ubuntu 12.04 and are connected with 10 Gigabit

Table 3.1
Two large datasets

	m	n	k	RAM
dataset 1	1.0×10^5	2.0×10^5	2.0×10^3	150GB
dataset 2	1.5×10^5	3.0×10^5	3.0×10^3	337GB

ethernet network. Since each has 8 cores, we run the code with 80 processes so that each process runs on its own core. Such a setup is charged for under \$17 per hour.

We solve the large-scale ℓ_1 minimization problems with a C implementation that matches the Matlab implementation in the previous section. The implementation consists of a single file of C code of about 300 lines, which is available for download on our personal website.

Table 3.2
Time results for large scale ℓ_1 minimization examples

	150GB Test			337GB Test		
	Itr	Time(s)	Cost(\$)	Itr	Time(s)	Cost(\$)
Data generation	–	44.4	0.21	–	99.5	0.5
CPU per iteration	–	1.32	–	–	2.85	–
Comm. per iteration	–	0.07	–	–	0.15	–
Reach 10^{-1}	23	30.4	0.14	27	79.08	0.37
Reach 10^{-2}	30	39.4	0.18	39	113.68	0.53
Reach 10^{-3}	86	112.7	0.53	84	244.49	1.15
Reach 10^{-4}	234	307.9	1.45	89	259.24	1.22

The breakdown of the wall-clock time is summarized in Table 3.2. We can observe that Jacobian ADMM is very efficient in obtaining a relative low accuracy, which is usually sufficient for large-scale problems. We want to point out that the basic BLAS operations in our implantation can be further improved by using other libraries such as hardware-optimized BLAS libraries produced by ATLAS, Armadillo, etc. Those libraries might lead to several times of speedup[1]. We use GSL due to its ease of use, so the code can be easily adapted for solving similar problems.

4. A Sufficient Condition for Convergence of Jacobian ADMM. In this section, we provide a sufficient condition to guarantee the convergence of Jacobian

[1]http://nghiaho.com/?p=1726

ADMM (Algorithm 3), which does not use either proximal terms or correction steps. The condition only depends on the coefficient matrices A_i, without imposing further assumptions on the objective functions f_i or the penalty parameter ρ. For the Gauss-Seidel ADMM (Algorithm 2), a sufficient condition for convergence is provided in [5] for the special case $N = 3$, assuming two of the three coefficient matrices are orthogonal. Our condition does not require exact orthogonality. Instead, we mainly assume that the matrices $A_i,\ i = 1, 2, \ldots, N$ are mutually "near-orthogonal" and have full column-rank.

Theorem 4.1. *Suppose that there exists $\delta > 0$ such that*

$$\|A_i^\top A_j\| \le \delta,\ \forall\ i \ne j,\ \text{and}\ \lambda_{min}(A_i^\top A_i) > 3(N-1)\delta,\ \forall\ i, \tag{4.1}$$

where $\lambda_{min}(A_i^\top A_i)$ denotes the smallest eigenvalue of $A_i^\top A_i$. Then the sequence $\{\mathbf{u}^k\}$ generated by Algorithm 3 converges to a solution $\mathbf{u}^*$ to the problem (1.1).

The proof technique is motivated by the contraction analysis of the sequence $\{\mathbf{u}^k\}$ under some G-norm (e.g., see [19, 8, 16]). To prove the theorem, we first need the following lemma:

Lemma 4.2. *Let*

$$G_0 := \begin{pmatrix} \rho A_1^\top A_1 & & & \\ & \ddots & & \\ & & \rho A_N^\top A_N & \\ & & & \frac{1}{\rho}\mathbf{I} \end{pmatrix},$$

where $\mathbf{I}$ is the identity matrix of size $m \times m$. For $k \ge 1$, we have

$$\|\mathbf{u}^k - \mathbf{u}^*\|_{G_0}^2 - \|\mathbf{u}^{k+1} - \mathbf{u}^*\|_{G_0}^2 \ge h_0(\mathbf{u}^k,\ \mathbf{u}^{k+1}), \tag{4.2}$$

where

$$h_0(\mathbf{u}^k,\ \mathbf{u}^{k+1}) := \|\mathbf{u}^k - \mathbf{u}^{k+1}\|_{G_0}^2 + 2(\lambda^k - \lambda^{k+1})^\top A(\mathbf{x}^k - \mathbf{x}^{k+1}). \tag{4.3}$$

This lemma follows directly from Lemma 2.1 since it is a special case with $\gamma = 1$ and $P_i = 0,\ \forall i$. Now we are ready to prove the theorem.

Proof. [Proof of Theorem 4.1] By the assumption $\|A_i^\top A_j\| \leq \delta,\ i \neq j$, we have

$$\left|\sum_{i\neq j}\langle A_i\mathbf{a}_i, A_j\mathbf{b}_j\rangle\right| \leq \sum_{i\neq j}\delta\|\mathbf{a}_i\|\|\mathbf{b}_j\| \leq \frac{\delta}{2}(N-1)(\|\mathbf{a}\|^2+\|\mathbf{b}\|^2),\ \forall\ \mathbf{a},\mathbf{b} \tag{4.4}$$

To simplify the notation, we let

$$\mathbf{a}_i^k := \mathbf{x}_i^k - \mathbf{x}_i^*,\ i = 1,2,\dots,N. \tag{4.5}$$

Note that

$$\lambda^k - \lambda^{k+1} = \rho A\mathbf{a}^{k+1},\ \mathbf{x}^k - \mathbf{x}^{k+1} = \mathbf{a}^k - \mathbf{a}^{k+1}.$$

Then, we can rewrite (4.3) as

$$\frac{1}{\rho}h_0(\mathbf{u}^k - \mathbf{u}^{k+1}) = \sum_i \|A_i(\mathbf{a}_i^k - \mathbf{a}_i^{k+1})\|^2 + \|A\mathbf{a}^{k+1}\|^2 + 2\langle A\mathbf{a}^{k+1},\ A(\mathbf{a}^k - \mathbf{a}^{k+1})\rangle \tag{4.6}$$

$$= \sum_i \|A_i\mathbf{a}_i^k\|^2 + 2\sum_{i\neq j}\langle A_i\mathbf{a}_i^{k+1}, A_j\mathbf{a}_j^k\rangle - \sum_{i\neq j}\langle A_i\mathbf{a}_i^{k+1}, A_j\mathbf{a}_j^{k+1}\rangle \tag{4.7}$$

$$\geq \sum_i \|A_i\mathbf{a}_i^k\|^2 - (N-1)\delta(\|\mathbf{a}^{k+1}\|^2 + \|\mathbf{a}^k\|^2) - (N-1)\delta\|\mathbf{a}^{k+1}\|^2 \tag{4.8}$$

$$= \sum_i \|A_i\mathbf{a}_i^k\|^2 - (N-1)\delta\|\mathbf{a}^k\|^2 - 2(N-1)\delta\|\mathbf{a}^{k+1}\|^2, \tag{4.9}$$

where the inequality (4.8) comes from (4.4). By Lemma 4.2, we have

$$\begin{aligned}&\|\mathbf{u}^k - \mathbf{u}^*\|_{G_0}^2 - 2(N-1)\delta\rho\|\mathbf{a}^k\|^2 \\ \geq&\|\mathbf{u}^{k+1} - \mathbf{u}^*\|_{G_0}^2 - 2(N-1)\delta\rho\|\mathbf{a}^{k+1}\|^2 + \rho\sum_i\|A_i\mathbf{a}_i^k\|^2 - 3(N-1)\delta\rho\|\mathbf{a}^k\|^2.\end{aligned} \tag{4.10}$$

We further simplify (4.10) as

$$\mathbf{b}^k - \mathbf{b}^{k+1} \geq \mathbf{d}^k, \tag{4.11}$$

where the sequences $\{\mathbf{b}^k\}$ and $\{\mathbf{d}^k\}$ are defined by

$$\mathbf{b}^k := \|\mathbf{u}^k - \mathbf{u}^*\|_{G_0}^2 - 2(N-1)\delta\rho\|\mathbf{a}^k\|^2, \tag{4.12}$$

$$\mathbf{d}^k := \rho\sum_i\|A_i\mathbf{a}_i^k\|^2 - 3(N-1)\delta\rho\|\mathbf{a}^k\|^2. \tag{4.13}$$

By the definition of G_0, we have

$$\mathbf{b}^k = \rho \sum_i \|A_i \mathbf{a}_i^k\|^2 - 2(N-1)\delta\rho\|\mathbf{a}_i^k\|^2 + \frac{1}{\rho}\|\lambda^k - \lambda^*\|^2. \tag{4.14}$$

Since we assume $\lambda_{\min}(A_i^\top A_i) > 3(N-1)\delta$, it follows that

$$\|A_i \mathbf{a}_i^k\|^2 \geq 3(N-1)\delta\|\mathbf{a}_i^k\|^2, \ \forall i. \tag{4.15}$$

Then it is easy to see that $\mathbf{b}^k \geq 0$ and $\mathbf{d}^k \geq 0$. By (4.11), the nonnegative sequence $\{\mathbf{b}^k\}$ is monotonically non-increasing. Hence, $\{\mathbf{b}^k\}$ converges to some $\mathbf{b}^* \geq 0$. By (4.11), it also follows that $\mathbf{d}^k \to 0$. Therefore, $\mathbf{a}^k \to 0$, i.e., $\mathbf{x}^k \to \mathbf{x}^*$.

Next we show $\lambda^k \to \lambda^*$. By taking limit of (4.14) and using $\mathbf{a}^k \to 0$, we have

$$\mathbf{b}^* = \lim_{k\to\infty} \mathbf{b}^k = \lim_{k\to\infty} \frac{1}{\rho}\|\lambda^k - \lambda^*\|^2. \tag{4.16}$$

To show $\lambda^k \to \lambda^*$, it thus suffices to show $\mathbf{b}^* = 0$.

By (4.16), $\{\lambda^k\}$ is bounded and must have a convergent subsequence $\lambda^{k_j} \to \bar{\lambda}$. Recall the optimality conditions for the $\mathbf{x}_i$-subproblems (1.12):

$$A_i^\top \left(\lambda^k - \rho(A_i \mathbf{x}_i^{k+1} + \sum_{j\neq i} A_j \mathbf{x}_j^k - c)\right) \in \partial f_i(\mathbf{x}_i^{k+1}). \tag{4.17}$$

By Theorem 24.4 of [27], taking limit over the subsequence $\{k_j\}$ on both sides of (4.17) yields:

$$A_i^\top \bar{\lambda} \in \partial f_i(\mathbf{x}_i^*), \ \forall\ i. \tag{4.18}$$

Therefore, $(\mathbf{x}^*, \bar{\lambda})$ satisfies the KKT conditions of the problem (1.1). Since $(\mathbf{x}^*, \lambda^*)$ is any KKT point, now we let $\lambda^* = \bar{\lambda}$. By (4.16) and $\|\lambda^{k_j} - \lambda^*\|^2 \to 0$, we must have $\mathbf{b}^* = 0$, thereby completing the proof. □

Under the similar near-orthogonality assumption on the matrices A_i, $i = 1, 2, \ldots, N$, we have the following convergence result for Proximal Jacobian ADMM:

THEOREM 4.3. *Suppose $\|A_i^\top A_j\| \leq \delta$ for all $i \neq j$, and the parameters in Algorithm 4 satisfy the following condition: for some $\alpha, \beta > 0$,*

$$\begin{cases} P_i \succ \rho(\frac{1}{\alpha} - 1)A_i^\top A_i + \frac{\rho}{\beta}\delta(N-1)\mathbf{I} \\ \lambda_{\min}(A_i^\top A_i) > \frac{2-\gamma+\beta}{2-\gamma-\alpha}\delta(N-1) \end{cases} \textit{for } i = 1, \ldots, N. \tag{4.19}$$

Then Algorithm 4 converges to a solution to the problem (1.1)*.*

Proof. Let

$$H := \begin{pmatrix} A_1^\top A_1 & & \\ & \ddots & \\ & & A_N^\top A_N \end{pmatrix}.$$

If $\|A_i^\top A_j\| \leq \delta$ for all $i \neq j$, then it is easy to show the following: for any $\mathbf{x}$ and $\mathbf{y}$,

$$\begin{aligned} \|A\mathbf{x}\|^2 &= \sum_{i=1}^N \|A_i\mathbf{x}_i\|^2 + \sum_{i\neq j} \mathbf{x}_i^\top A_i^\top A_j \mathbf{x}_j \geq \sum_{i=1}^N \|A_i\mathbf{x}_i\|^2 - \delta \sum_{i \neq j} \|\mathbf{x}_i\|\|\mathbf{x}_j\| \\ &\geq \sum_{i=1}^N \|A_i\mathbf{x}_i\|^2 - \delta(N-1)\|\mathbf{x}\|^2 = \|\mathbf{x}\|^2_{[H-\delta(N-1)\mathbf{I}]}, \end{aligned} \tag{4.20}$$

and

$$\begin{aligned} 2\mathbf{x}^\top A^\top A\mathbf{y} &= 2\sum_{i=1}^N \mathbf{x}_i^\top A_i^\top A_j \mathbf{y}_j + 2\sum_{i\neq j} \mathbf{x}_i^\top A_i^\top A_j \mathbf{y}_j \geq 2\sum_{i=1}^N \mathbf{x}_i^\top A_i^\top A_j \mathbf{y}_j - 2\delta \sum_{i\neq j} \|\mathbf{x}_i\|\|\mathbf{y}_j\| \\ &\geq -\sum_{i=1}^N \alpha\|A_i\mathbf{x}_i\|^2 - \beta\delta(N-1)\|\mathbf{x}\|^2 - \sum_{i=1}^N \frac{1}{\alpha}\|A_i\mathbf{y}_i\|^2 - \frac{1}{\beta}\delta(N-1)\|\mathbf{y}\|^2 \\ &= -\|\mathbf{x}\|^2_{[\alpha H + \beta\delta(N-1)\mathbf{I}]} - \|\mathbf{y}\|^2_{[\frac{1}{\alpha}H + \frac{1}{\beta}\delta(N-1)\mathbf{I}]}, \ \forall \alpha, \beta > 0, \end{aligned} \tag{4.21}$$

Using the above inequalities, we have

$$\begin{aligned} &\frac{2}{\gamma}(\lambda^k - \lambda^{k+1})^\top A(\mathbf{x}^k - \mathbf{x}^{k+1}) = 2\rho(\mathbf{x}^{k+1} - \mathbf{x}^*)A^\top A(\mathbf{x}^k - \mathbf{x}^{k+1}) \\ &\geq -\rho\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2_{[\alpha H + \beta\delta(N-1)\mathbf{I}]} - \rho\|\mathbf{x}^k - \mathbf{x}^{k+1}\|^2_{[\frac{1}{\alpha}H + \frac{1}{\beta}\delta(N-1)\mathbf{I}]}, \end{aligned} \tag{4.22}$$

and

$$\|\lambda^k - \lambda^{k+1}\|^2 = \gamma^2\rho^2\|A(\mathbf{x}^{k+1} - \mathbf{x}^*)\|^2 \geq \gamma^2\rho^2\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2_{[H-\delta(N-1)\mathbf{I}]}. \tag{4.23}$$

Therefore,

$$\begin{aligned} h(\mathbf{u}^k, \mathbf{u}^{k+1}) \geq &\|\mathbf{x}^k - \mathbf{x}^{k+1}\|^2_{G_x} + (2-\gamma)\rho\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2_{[H-\delta(N-1)\mathbf{I}]} \\ &- \rho\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2_{[\alpha H + \beta\delta(N-1)\mathbf{I}]} - \rho\|\mathbf{x}^k - \mathbf{x}^{k+1}\|^2_{[\frac{1}{\alpha}H + \frac{1}{\beta}\delta(N-1)\mathbf{I}]}. \end{aligned} \tag{4.24}$$

As long as the following holds:

$$\begin{cases} G_x \succ \frac{\rho}{\alpha} H + \frac{\rho}{\beta}\delta(N-1)\mathbf{I}, \\ (2-\gamma)\rho[H - \delta(N-1)\mathbf{I}] \succ \rho[\alpha H + \beta\delta(N-1)\mathbf{I}], \end{cases} \tag{4.25}$$

which is equivalent to the condition (4.19), there must exist some $\eta > 0$ such that (2.9) and (2.10) hold. Then the convergence of Algorithm 4 follows immediately from the standard analysis of contraction methods [14]. □

5. On $o(1/k)$ Convergence Rate of ADMM. The convergence of the standard two-block ADMM has been long established in the literature [10, 12]. Its convergence rate has been actively studied; see [21, 19, 18, 13, 8, 20] and the references therein. In the following, we briefly review the convergence analysis for ADMM ($N = 2$) and then improve the $O(1/k)$ convergence rate established in [18] slightly to $o(1/k)$ by using the same technique as in Subsection 2.2.

As suggested in [18], the quantity $\|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2$ can be used to measure the optimality of the iterations of ADMM , where

$$\mathbf{w} := \begin{pmatrix} \mathbf{x}_2 \\ \lambda \end{pmatrix}, \ H := \begin{pmatrix} \rho A_2^\top A_2 & \\ & \frac{1}{\rho}\mathbf{I} \end{pmatrix},$$

and $\mathbf{I}$ is the identity matrix of size $m \times m$. Note that $\mathbf{x}_1$ is not part of $\mathbf{w}$ because $\mathbf{x}_1$ can be regarded as an intermediate variable in the iterations of ADMM, whereas $(\mathbf{x}_2, \lambda)$ are the essential variables [3]. In fact, if $\|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2 = 0$ then $\mathbf{w}^{k+1}$ is optimal. The reasons are as follows. Recall the subproblems of ADMM:

$$\mathbf{x}_1^{k+1} = \arg\min_{\mathbf{x}_1} \ f_1(\mathbf{x}_1) + \frac{\rho}{2}\|A_1\mathbf{x}_1 + A_2\mathbf{x}_2^k - \lambda^k/\rho\|^2, \tag{5.1}$$

$$\mathbf{x}_2^{k+1} = \arg\min_{\mathbf{x}_2} \ f_2(\mathbf{x}_2) + \frac{\rho}{2}\|A_1\mathbf{x}_1^{k+1} + A_2\mathbf{x}_2 - \lambda^k/\rho\|^2. \tag{5.2}$$

By the formula for λ^{k+1}, their optimality conditions can be written as:

$$A_1^\top\lambda^{k+1} - \rho A_1^\top A_2(\mathbf{x}_2^k - \mathbf{x}_2^{k+1}) \in \partial f(\mathbf{x}_1^{k+1}), \tag{5.3}$$

$$A_2^\top\lambda^{k+1} \in \partial f_2(\mathbf{x}_2^{k+1}). \tag{5.4}$$

In comparison with the KKT conditions (1.15) and (1.16), we can see that $\mathbf{u}^{k+1} =$

$\left(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}, \lambda^{k+1}\right)$ is a solution of (1.1) if and only if the following holds:

$$\mathbf{r}_p^{k+1} := A_1\mathbf{x}_1^{k+1} + A_2\mathbf{x}_2^{k+1} - c = 0 \quad \text{(primal feasibility)}, \tag{5.5}$$

$$\mathbf{r}_d^{k+1} := \rho A_1^\top A_2(\mathbf{x}_2^k - \mathbf{x}_2^{k+1}) = 0 \qquad \text{(dual feasibility)}. \tag{5.6}$$

By the update formula for λ^{k+1}, we can write $\mathbf{r}_p$ equivalently as

$$\mathbf{r}_p^{k+1} = \frac{1}{\rho}(\lambda^k - \lambda^{k+1}). \tag{5.7}$$

Clearly, if $\|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2 = 0$ then the optimality conditions (5.5) and (5.6) are satisfied, so $\mathbf{w}^{k+1}$ is a solution. On the other hand, if $\|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2$ is large, then $\mathbf{w}^{k+1}$ is likely to be far away from being a solution. Therefore, the quantity $\|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2$ can be viewed as a measure of the distance between the iteration $\mathbf{w}^{k+1}$ and the solution set. Furthermore, based on the variational inequality (1.17) and the variational characterization of the iterations of ADMM, it is reasonable to use the quadratic term $\|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2$ rather than $\|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H$ to measure the convergence rate of ADMM (see [18] for more details).

The work [18] proves that $\|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2$ converges to zero at a rate of $O(1/k)$. The key steps of the proof are to establish the following properties:

- the sequence $\{\mathbf{w}^k\}$ is contractive:

$$\|\mathbf{w}^k - \mathbf{w}^*\|_H^2 - \|\mathbf{w}^{k+1} - \mathbf{w}^*\|_H^2 \geq \|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2, \tag{5.8}$$

- the sequence $\|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2$ is monotonically non-increasing:

$$\|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2 \leq \|\mathbf{w}^{k-1} - \mathbf{w}^k\|_H^2. \tag{5.9}$$

The contraction property (5.8) has been long established and its proof dates back to [10, 12]. Inspired by [18], we provide a much shorter proof for (5.9) than the one in [18].

Proof. [Proof of (5.9)] Let $\Delta\mathbf{x}_i^{k+1} = \mathbf{x}_i^k - \mathbf{x}_i^{k+1}$ and $\Delta\lambda^{k+1} = \lambda^k - \lambda^{k+1}$. By Lemma 1.1, i.e., (1.19), the optimality condition (5.3) at the k-th and $(k+1)$-th iterations yields:

$$\langle \Delta\mathbf{x}_1^{k+1},\ A_1^\top \Delta\lambda^{k+1} - \rho A_1^\top A_2(\Delta\mathbf{x}_2^k - \Delta\mathbf{x}_2^{k+1})\rangle \geq 0. \tag{5.10}$$

Similarly for (5.4), we obtain

$$\langle \Delta \mathbf{x}_2^{k+1},\ A_2^\top \Delta\lambda^{k+1}\rangle \geq 0. \tag{5.11}$$

Adding the above two inequalities together, we have

$$(A_1\Delta\mathbf{x}_1^{k+1} + A_2\Delta\mathbf{x}_2^{k+1})^\top \Delta\lambda^{k+1} - \rho(A_1\Delta\mathbf{x}_1^{k+1})^\top A_2(\Delta\mathbf{x}_2^k - \Delta\mathbf{x}_2^{k+1}) \geq 0. \tag{5.12}$$

Using the equality according to (5.7):

$$A_1\Delta\mathbf{x}_1^{k+1} + A_2\Delta\mathbf{x}_2^{k+1} = \frac{1}{\rho}(\Delta\lambda^k - \Delta\lambda^{k+1}), \tag{5.13}$$

(5.12) becomes

$$\frac{1}{\rho}(\Delta\lambda^k - \Delta\lambda^{k+1})^\top\Delta\lambda^{k+1} - (\Delta\lambda^k - \Delta\lambda^{k+1} - \rho A_2\Delta\mathbf{x}_2^{k+1})^\top A_2(\Delta\mathbf{x}_2^k - \Delta\mathbf{x}_2^{k+1}) \geq 0. \tag{5.14}$$

After rearranging the terms, we get

$$\begin{aligned}&(\sqrt{\rho}A_2\Delta\mathbf{x}_2^k + \frac{1}{\sqrt{\rho}}\Delta\lambda^k)^\top(\sqrt{\rho}A_2\Delta\mathbf{x}_2^{k+1} + \frac{1}{\sqrt{\rho}}\Delta\lambda^{k+1}) - (A_2\Delta\mathbf{x}_2^k)^\top\Delta\lambda^k - (A_2\Delta\mathbf{x}_2^{k+1})^\top\Delta\lambda^{k+1}\\ &\geq \frac{1}{\rho}\|\Delta\lambda^{k+1}\|^2 + \rho\|A_2\Delta\mathbf{x}_2^{k+1}\|^2 = \|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2.\end{aligned} \tag{5.15}$$

By the Cauchy-Schwarz inequality, we have

$$(a_1+b_1)^\top(a_2+b_2) \leq (\|a_1+b_1\|^2 + \|a_2+b_2\|^2)/2, \tag{5.16}$$

or equivalently,

$$(a_1+b_1)^\top(a_2+b_2) - a_1^\top b_1 - a_2^\top b_2 \leq (\|a_1\|^2 + \|b_1\|^2 + \|a_2\|^2 + \|b_2\|^2)/2. \tag{5.17}$$

Applying the above inequality to the left-hand side of (5.15), we have

$$\begin{aligned}\|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2 &\leq \left(\rho\|A_2\Delta\mathbf{x}_2^k\|^2 + \frac{1}{\rho}\|\Delta\lambda^k\|^2 + \rho\|A_2\Delta\mathbf{x}_2^{k+1}\|^2 + \frac{1}{\rho}\|\Delta\lambda^{k+1}\|^2\right)/2\\ &= \left(\|\mathbf{w}^{k-1} - \mathbf{w}^k\|_H^2 + \|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2\right)/2,\end{aligned} \tag{5.18}$$

and thus (5.9) follows immediately. □

We are now ready to improve the convergence rate from $O(1/k)$ to $o(1/k)$.

THEOREM 5.1. *The sequence $\{\mathbf{w}^k\}$ generated by Algorithm 2 (for $N = 2$) converges to a solution $\mathbf{w}^*$ of problem* (1.1) *in the H-norm, i.e., $\|\mathbf{w}^k - \mathbf{w}^*\|_H^2 \to 0$, and $\|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2 = o(1/k)$. Therefore,*

$$\|A_1\mathbf{x}_1^k - A_1\mathbf{x}_1^{k+1}\|^2 + \|A_2\mathbf{x}_2^k - A_2\mathbf{x}_2^{k+1}\|^2 + \|\lambda^k - \lambda^{k+1}\|^2 = o(1/k), \tag{5.19}$$

for $k \to \infty$.

Proof. Using the contractive property of the sequence $\{\mathbf{w}^k\}$ (5.8) along with the optimality conditions, the convergence of $\|\mathbf{w}^k - \mathbf{w}^*\|_H^2 \to 0$ follows from the standard analysis for contraction methods [14].

By (5.8), we have

$$\sum_{k=1}^{n} \|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2 \leq \|\mathbf{w}^1 - \mathbf{w}^*\|_H^2 - \|\mathbf{w}^{n+1} - \mathbf{w}^*\|_H^2, \ \forall n. \tag{5.20}$$

Therefore, $\sum_{k=1}^{\infty} \|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2 < \infty$. By (5.9), $\|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2$ is monotonically non-increasing and nonnegative. So Lemma 1.2 indicates that $\|\mathbf{w}^k - \mathbf{w}^{k+1}\|_H^2 = o(1/k)$, which further implies that $\|A_2\mathbf{x}_2^k - A_2\mathbf{x}_2^{k+1}\|^2 = o(1/k)$ and $\|\lambda^k - \lambda^{k+1}\|^2 = o(1/k)$. By (5.13), we also have $\|A_1\mathbf{x}_1^k - A_1\mathbf{x}_1^{k+1}\|^2 = o(1/k)$. Thus (5.19) follows immediately. □

REMARK 5.1. *The proof technique based on Lemma 1.2 can be applied to improve some other existing convergence rates of $O(1/k)$ (e.g., [16, 7]) to $o(1/k)$ as well.*

6. Conclusion.

Due to the dramatically increasing demand for dealing with big data, parallel and distributed computational methods are highly desirable. ADMM, as a versatile algorithmic tool, has proven to be very effective at solving many large-scale problems and well suited for distributed computing. Yet, its parallelization still needs further investigation and improvement. This paper proposes a simpler parallel and distributed ADMM for solving problems with separable structures. The algorithm framework introduces more flexibility for computing the subproblems due to the use of proximal terms $\|\mathbf{x}_i - \mathbf{x}_i^k\|_{P_i}^2$ with wisely chosen P_i. Its theoretical properties such as global convergence and an $o(1/k)$ rate are established. Our numerical results demonstrate the efficiency of the proposed method in comparison with several existing parallel algorithms. The code will be available online for further studies. In addition, we provide a simple sufficient condition to ensure the convergence of

Jacobian ADMM and demonstrate a simple technique to improve the convergence rate of ADMM from $O(1/k)$ to $o(1/k)$.

Acknowledgements. Wei Deng is supported by NSF grant ECCS-1028790. Ming-Jun Lai is partially supported by a Simon collaboration grant for 2013–2018. Wotao Yin is partially supported by NSF grants DMS-0748839 and DMS-1317602, and ARO/ARL MURI grant FA9550-10-1-0567.

REFERENCES

[1] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods, Second Edition*, Athena Scientific, 1997.

[2] D. P. Bertsekas, *Extended monotropic programming and duality*, Journal of optimization theory and applications, 139 (2008), pp. 209–225.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Machine Learning, 3 (2010), pp. 1–122.

[4] V. Chandrasekaran, P. A Parrilo, and A. S Willsky, *Latent variable graphical model selection via convex optimization*, Annals of Statistics, 40 (2012), pp. 1935–1967.

[5] C. H. Chen, B. S. He, Y. Y. Ye, and X. M. Yuan, *The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent*, preprint, (2013).

[6] G. Chen and M. Teboulle, *A proximal-based decomposition method for convex minimization problems*, Mathematical Programming, 64 (1994), pp. 81–101.

[7] E. Corman and X. M. Yuan, *A generalized proximal point algorithm and its convergence rate*, (2013).

[8] W. Deng and W. Yin, *On the global and linear convergence of the generalized alternating direction method of multipliers*, Rice University CAAM Technical Report TR12-14, (2012).

[9] H. Everett, *Generalized lagrange multiplier method for solving problems of optimum allocation of resources*, Operations research, 11 (1963), pp. 399–417.

[10] D. Gabay and B. Mercier, *A dual algorithm for the solution of nonlinear variational problems via finite element approximation*, Computers & Mathematics with Applications, 2 (1976), pp. 17–40.

[11] R. Glowinski, *Numerical methods for nonlinear variational problems*, Springer Series in Computational Physics, 1984.

[12] R. Glowinski and A. Marrocco, *Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité, d'une classe de problèmes de Dirichlet non linéaires*, Laboria, 1975.

[13] T. Goldstein, B. O'Donoghue, and S. Setzer, *Fast alternating direction optimization methods*, CAM report 12-35, UCLA, (May 2012).

[14] B. S. He, *A class of projection and contraction methods for monotone variational inequalities*, Applied Mathematics and Optimization, 35 (1997), pp. 69–76.

[15] ———, *Parallel splitting augmented lagrangian methods for monotone structured variational inequalities*, Computational Optimization and Applications, 42 (2009), pp. 195–212.

[16] B. S. He, L. S. Hou, and X. M. Yuan, *On full Jacobian decomposition of the augmented lagrangian method for separable convex programming*, (2013).

[17] B. S. He, M. Tao, and X. M. Yuan, *Alternating direction method with gaussian back substitution for separable convex programming*, SIAM Journal on Optimization, 22 (2012), pp. 313–340.

[18] B. S. He and X. M. Yuan, *On non-ergodic convergence rate of Douglas-Rachford alternating direction method of multipliers*, (2012).

[19] ———, *On the $O(1/n)$ convergence rate of the Douglas-Rachford alternating direction method*, SIAM Journal on Numerical Analysis, 50 (2012), pp. 700–709.

[20] M. Hong and Z.-Q. Luo, *On the linear convergence of the alternating direction method of multipliers*, arXiv preprint arXiv:1208.3922, (2012).

[21] P.-L. Lions and B. Mercier, *Splitting algorithms for the sum of two nonlinear operators*, SIAM Journal on Numerical Analysis, 16 (1979), pp. 964–979.

[22] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Puschel, *D-ADMM: A communication-efficient distributed algorithm for separable optimization*, IEEE Transactions on Signal Processing, 61 (2013), pp. 2718–2723.

[23] N. Parikh and S. Boyd, *Block splitting for distributed optimization*, Mathematical Programming Computation, (2013), pp. 1–26.

[24] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma, *RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2012), pp. 2233–2246.

[25] Z. Peng, M. Yan, and W. Yin, *Parallel and distributed sparse optimization*, IEEE Asilomar Conference on Signals Systems and Computers, (2013).

[26] R. T. Rockafellar, *Monotropic programming: descent algorithms and duality*, Nonlinear programming, 4 (1981), pp. 327–366.

[27] ———, *Convex analysis*, vol. 28, Princeton University Press, 1997.

[28] N. Z. Shor, K. C Kiwiel, and A. Ruszcayski, *Minimization methods for non-differentiable functions*, Springer-Verlag New York, Inc., 1985.

[29] M. Tao and X. M. Yuan, *Recovering low-rank and sparse components of matrices from incomplete and noisy observations*, SIAM Journal on Optimization, 21 (2011), pp. 57–81.

[30] X. F. Wang, M. Y. Hong, S. Q. Ma, and Z.-Q. Luo, *Solving multiple-block separable convex minimization problems using two-block alternating direction method of multipliers*, arXiv preprint arXiv:1308.5294, (2013).

[31] E. Wei and A. Ozdaglar, *On the $O(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers*, arXiv preprint arXiv:1307.8254, (2013).

[32] J. F. Yang and Y. Zhang, *Alternating direction algorithms for ℓ_1-problems in compressive sensing*, SIAM journal on scientific computing, 33 (2011), pp. 250–278.

[33] X. Zhang, M. Burger, and S. Osher, *A unified primal-dual algorithm framework based on Bregman iteration*, Journal of Scientific Computing, 46 (2011), pp. 20–46.