Parallel Multi-Block ADMM with $o(1/k)$ Convergence

Wotao Yin (UCLA Math)

W. Deng, M.-J. Lai, Z. Peng, and W. Yin, Parallel Multi-Block ADMM with $o(1/k)$ Convergence, UCLA CAM 13-64, 2013.

Advanced Workshop at Shanghai U.


游人止步
NO ADM ITTANCE
北京市人民政府立

Linearly Constrained Separable Problem

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

- $f_i : \mathbb{R}^{n_i} \rightarrow (-\infty, +\infty]$ are convex functions. $N \geq 2$.
- a.k.a. *extended monotropic programming* [Bertsekas, 2008]
- Examples:
  - Linear programming
  - Multi-agent network optimization
  - Exchange problem
  - Regularization model

Parallel and Distributed Algorithms

Motivation:

- Data may be collected and stored in a distributed way
- Often difficult to minimize all the f_i's jointly

Strategy:

- Decompose the problem into N *simpler and smaller* subproblems
- Solve subproblems in parallel
- Coordinate by passing some information

Dual Decomposition

$$\begin{cases} (\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}, \ldots, \mathbf{x}_N^{k+1}) = \arg\min_{\{\mathbf{x}_i\}} \ \mathcal{L}(\mathbf{x}_1, \ldots, \mathbf{x}_N, \lambda^k), \\ \lambda^{k+1} = \lambda^k - \alpha_k \left(\sum_{i=1}^N A_i \mathbf{x}_i^{k+1} - c \right), \quad \alpha_k > 0. \end{cases}$$

- Lagrangian:

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

- $\mathbf{x}$-step has N decoupled $\mathbf{x}_i$-subproblems, *parallelizable*:

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

- Convergence rate: $O(1/\sqrt{k})$ (for general convex problems)
- Often slow convergence in practice

Distributed ADMM

[Bertsekas and Tsitsiklis, 1997, Boyd et al., 2010, Wang et al., 2013]

- Variable splitting:

$$\begin{aligned}\min_{\{\mathbf{x}_i\},\{\mathbf{z}_i\}} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i)\\ \text{s.t.} \quad & A_i\mathbf{x}_i - \mathbf{z}_i = \frac{c}{N},\ i = 1, 2, \ldots, N,\\ & \sum_{i=1}^{N} \mathbf{z}_i = 0.\end{aligned}$$

- Apply ADMM, alternatively update $\{\mathbf{x}_i\}$ and $\{\mathbf{z}_i\}$, then multipliers $\{\lambda_i\}$:

$$\mathbf{z}_i^{k+1} = \left(A_i\mathbf{x}_i^k - \frac{c}{N} - \frac{\lambda_i^k}{\rho}\right) - \frac{1}{N}\sum_{j=1}^{N}\left(A_j\mathbf{x}_j^k - \frac{c}{N} - \frac{\lambda_j^k}{\rho}\right), \forall i;$$

$$\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i}\ f_i(\mathbf{x}_i) + \frac{\rho}{2}\left\|A_i\mathbf{x}_i - \mathbf{z}_i^{k+1} - \frac{c}{N} - \frac{\lambda_i^k}{\rho}\right\|^2, \forall i;$$

$$\lambda_i^{k+1} = \lambda_i^k - \rho\left(A_i\mathbf{x}_i^{k+1} - \mathbf{z}_i^{k+1} - \frac{c}{N}\right), \forall i.$$

Jacobi ADMM

- Augmented Lagrangian:

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

- Do not introduce $\{\mathbf{z}_i\}$, directly apply Jacobi-type block minimization:

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

for $i = 1, \ldots, N$ **in parallel**;

$$\lambda^{k+1} = \lambda^k - \rho \left(\sum_{i=1}^N A_i \mathbf{x}_i^{k+1} - c \right).$$

- Not necessarily convergent (even if $N = 2$)
- Need *either* conditions *or* modifications to converge

A Sufficient Condition for Convergence

Theorem

Suppose that there exists $\delta > 0$ such that

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

Then Jacobi ADMM converges to a solution.

The assumption basically says:

- $\{A_i,\ i = 1, 2, \ldots, N\}$ are mutually "near-orthogonal"
- every A_i has full column rank and is sufficiently strong.

Proximal Jacobi ADMM

1. for $i = 1, \ldots, N$ in **parallel**,

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

2. $\lambda^{k+1} = \lambda^k - \gamma\rho \left(\sum_{i=1}^N A_i \mathbf{x}_i^{k+1} - b \right), \ \gamma > 0.$

- The added proximal term is critical to convergence.
- Some forms of $P_i \succeq 0$ make subproblems *easier to solve* and *more stable*.
- Global $o(1/k)$ convergence if P_i and γ are properly chosen.
- Suitable for parallel and distributed computing.

Little-o convergence

Lemma

If a sequence $\{a_k\} \subset \mathbb{R}$ *obeys:*

$$a_k \geq 0 \quad \textit{and} \quad \sum_{t=1}^{\infty} a_t < \infty,$$

then we have:

1. *(convergence)* $\lim_{k\to\infty} a_k = 0$;
2. *(ergodic convergence)* $\frac{1}{k}\sum_{t=1}^{k} a_t = O\left(\frac{1}{k}\right)$;
3. *(running best)* $\min_{t\leq k}\{a_t\} = o\left(\frac{1}{k}\right)$;
4. *(non-ergodic convergence) if* a_k *is monotonically nonincreasing, then* $a_k = o\left(\frac{1}{k}\right)$.

Convergence of Proximal Jacobi ADMM

Sufficient condition: there exist $\epsilon_i > 0,\ i = 1, 2, \ldots, N$ such that

$$\textbf{(C1)} \quad \begin{cases} P_i \succ \rho(\frac{1}{\epsilon_i} - 1)A_i^\top A_i,\ i = 1, 2, \ldots, N \\ \sum_{i=1}^N \epsilon_i < 2 - \gamma. \end{cases}$$

Simplification to (C1): set $\epsilon_i < \frac{2-\gamma}{N}$:

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

- $P_i = \tau_i \mathbf{I}$ (standard proximal method): $\tau_i > \rho \left(\frac{N}{2-\gamma} - 1 \right) \|A_i\|^2$
- $P_i = \tau_i \mathbf{I} - \rho A_i^\top A_i$ (prox-linear method): $\tau_i > \frac{\rho N}{2-\gamma} \|A_i\|^2$

$o(1/k)$ **Convergence Rate**

Notation:

$$G_x := \begin{pmatrix} P_1 + \rho A_1^\top A_1 & & \\ & \ddots & \\ & & P_N + \rho A_N^\top A_N \end{pmatrix}, \; G_x' := G_x - \rho A^\top A$$

Theorem

If $G_x' \succeq 0$ and the condition (C1) holds, then

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

Note: $(\mathbf{x}^{k+1}, \lambda^{k+1})$ is optimal if $\|\mathbf{x}^k - \mathbf{x}^{k+1}\|_{G_x'}^2 = 0,\ \|\lambda^k - \lambda^{k+1}\|^2 = 0$. The quantity $\|\mathbf{u}^k - \mathbf{u}^{k+1}\|_{G'}^2$ as a measure of the convergence rate. Proof is similar to He and Yuan [2012], He et al. [2013].

Adaptive Parameter Tuning

- Condition (C1) may be rather conservative.
- Adaptively adjusting the matrices $\{P_i\}$ with guaranteed convergence.

Initialize with small $P_i^0 \succeq 0$ $(i = 1, 2, \ldots, N)$ and a small $\eta > 0$;

for $k = 1, 2, \ldots$ **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: $P_i^{k+1} \leftarrow \alpha_i P_i^k + \beta_i Q_i \ (\alpha_i > 1, \ \beta_i \geq 0, \ Q_i \succ 0), \forall i$;

    Restart: $\mathbf{u}^k \leftarrow \mathbf{u}^{k-1}$;

Note: $h(\mathbf{u}^k, \mathbf{u}^{k+1})$ can be computed at little extra cost in the algorithm.

- Often yields much smaller paramters $\{P_i\}$ than those required by condition (C1), leading to substantially faster convergence in practice.

Numerical Experiments

Compare several parallel splitting algorithms:

- **Prox-JADMM**: Proximal Jacobi ADMM [this work]
- **VSADMM**: distributed ADMM, **variable splitting** [Bertsekas, 2008]
- **Corr-JADMM**: Jacobian ADMM with correction steps [He et al., 2013]

They have roughly the same per-iteration cost (in terms of both computation and communication).

Exchange Problem

Consider a network of N agents that exchange n commodities.

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

- $\mathbf{x}_i \in \mathbb{R}^n$ $(i = 1, 2, \ldots, N)$: quantities of commodities that are exchanged by agents i.
- $f_i : \mathbb{R}^n \to \mathbb{R}$: cost function for agent i.

Numerical Result

Let $f_i(\mathbf{x}_i) := \frac{1}{2}\|C_i\mathbf{x}_i - d_i\|^2$, $C_i \in \mathbb{R}^{p\times n}$ and $d_i \in \mathbb{R}^p$.

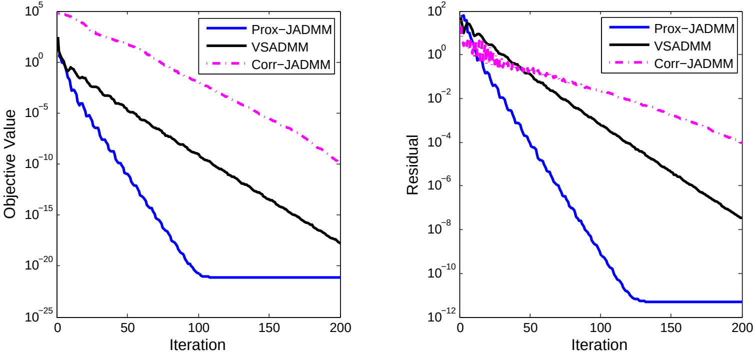


Figure: Exchange problem $(n = 100,\ N = 100,\ p = 80)$.

Basis Pursuit

Finding sparse solutions of an under-determined linear system:

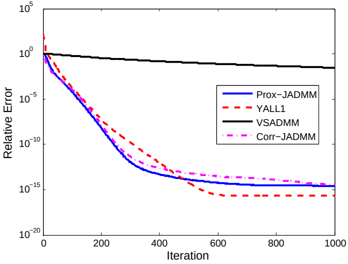
$$\min_{\mathbf{x}} \ \|\mathbf{x}\|_1 \quad \text{s.t.} \ \ A\mathbf{x} = c$$

- $\mathbf{x} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ $(m < n)$
- Partition data into N blocks:

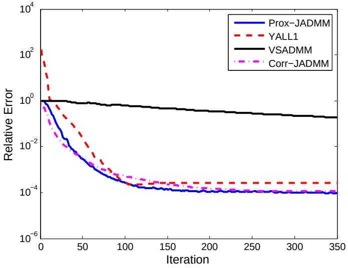
$$\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N], \ A = [A_1, A_2, \dots, A_N], \ f_i(\mathbf{x}_i) = \|\mathbf{x}_i\|_1$$

- **YALL1**: a dual-ADMM solver for the basis pursuit problem.

Numerical Result



(a) Noise-free $(\sigma = 0)$



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

Figure: ℓ_1-problem $(n = 1000,\ m = 300,\ k = 60)$.

Amazon EC2

Tested solve two basis pursuit problems.

	m	n	k	Size
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

Environment:

- C code uses GSL and MPI, about 300 lines
- 10 instances from Amazon, each with 8 cores and 68GB RAM
- price: $17 each hour

	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

Summary

- It is feasible to extend ADMM from 2 blocks to **3 or more blocks**
- Jacobi ADMM is good for problems with large and distributed data
- Gauss-Seidel ADMM is good for 3 or a few more blocks,
  Jacobi ADMM is good for many blocks
- Asynchronous subproblems become a real need (talk to Dong Qian)

References

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

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

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(1):1–122, 2010.

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.

B. S. He and X. M. Yuan. On non-ergodic convergence rate of Douglas-Rachford alternating direction method of multipliers. 2012.

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