On the $O(1/k)$ Convergence of Asynchronous Distributed Alternating Direction Method of Multipliers (ADMM)

Ermin Wei   Asu Ozdaglar

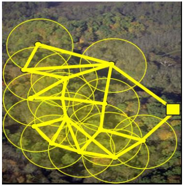
Laboratory for Information and Decision Systems
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology

Big Data Workshop
Simons Institute, Berkeley, CA
October 2013

Motivation

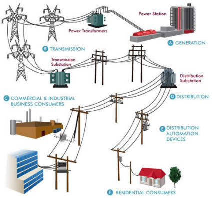
- Many networks are large-scale and comprise of agents with local information and heterogeneous preferences.
- This motivated much interest in developing distributed schemes for control and optimization of multi-agent networked systems.

Routing and congestion control in wireline and wireless networks



Parameter estimation in sensor networks

Multi-agent cooperative control and coordination



Smart grid systems

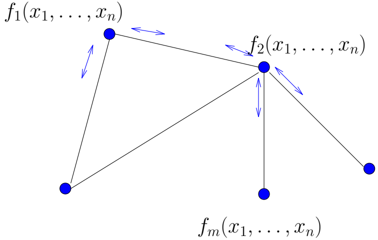
Distributed Multi-agent Optimization

- Many of these problems can be represented within the general formulation:
- A set of agents (nodes) $\{1, \ldots, N\}$ connected through a network.

- The goal is to cooperatively solve

$$\begin{aligned} \min_{x} \quad & \sum_{i=1}^{N} f_i(x) \\ \text{s.t.} \quad & x \in \mathbb{R}^n, \end{aligned}$$

$f_i(x) : \mathbb{R}^n \to \mathbb{R}$ is a convex (possibly nonsmooth) function, known only to agent i.



- Since such systems often lack a centralized processing unit, algorithms for this problem should involve each agent performing computations locally and communicating this information according to the underlying network.

Machine Learning Example

- A network of 3 sensors, supervised passive learning.
- Data is collected at different sensors: temperature t, electricity demand d.

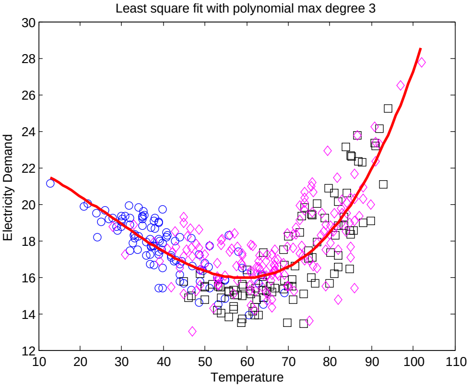
- System goal: learn a degree 3 polynomial electricity demand model:

$$d(t) = x_3 t^3 + x_2 t^2 + x_1 t + x_0.$$

- System objective:

$$\min_{x} \quad \sum_{i=1}^{3} ||A_i' x - d_i||_2^2 .$$

where $A_i = [1, t_i, t_i^2, t_i^3]'$ at input data t_i.



Machine Learning General Set-up

- A network of agents $i = 1, \ldots, N$.
- Each agent i has access to local feature vectors A_i and output b_i.
- System objective: train weight vector x to

$$\min_x \quad \sum_{i=1}^{N-1} L(A_i' x - b_i) + p(x),$$

for some loss function L (on the prediction error) and penalty function p (on the complexity of the model).

  - Example: Least-Absolute Shrinkage and Selection Operator (LASSO):

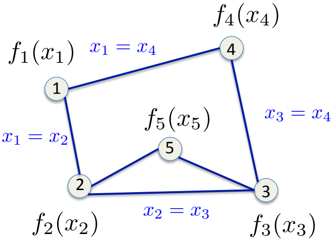
$$\min_x \quad \sum_{i=1}^{N-1} ||A_i' x - b_i||_2^2 + \lambda \, ||x||_1 \, .$$

  - Other examples from ML estimation, low rank matrix completion, image recovery [Schizas, Ribeiro, Giannakis 08], [Recht, Fazel, Parrilo 10], [Steidl, Teuber, 10]

Existing Distributed Algorithms

- Given an undirected connected graph $G = \{V, E\}$ with M nodes, we reformulate the problem as

$$\min_{x} \sum_{i=1}^{M} f_i(x_i)$$
$$s.t. \quad x_i = x_j, \quad \text{for} \quad (i,j) \in E,$$



- Distributed gradient/subgradient methods for solving these problems:
  - Each agent maintains an local estimate, updates it by taking a (sub)gradient step and averaging with neighbors' estimates.
  - Best known convergence rate: $O(1/\sqrt{k})$.[Nedic, Ozdaglar 08], [Lobel, Ozdaglar 09], [Duchi, Agarwal, Wainwright 12].

Faster ADMM-based Distributed Algorithms

- Classical Augmented Lagrangian/Method of Multipliers and Alternating Direction Method of Multipliers (ADMM) methods: fast and parallel [Glowinski, Marrocco 75], [Eckstein, Bertsekas 92], [Boyd et al. 10]:
- Known convergence rates for synchronous ADMM type algorithm:
  - [He, Yuan 11] General convex $O(1/k)$.
  - [Goldfarb et al. 10] Lipschitz gradient $O(1/k^2)$.
  - [Deng, Yin 12] Lipschitz gradient, strong convexity linear rate.
  - [Hong, Luo 12] Strong convexity linear rate.
- Highly decentralized nature of the problem calls for an asynchronous algorithm. Almost all known distributed algorithms are synchronous.[1]
- In this talk, we present asynchronous ADMM-type algorithms for general convex problems and show that it converges at the best known rate of $O(1/k)$ [Wei, Ozdaglar 13].

[1] Exceptions: [Ram, Nedic, Veeravalli 09], [Iutzeler, Bianchi, Ciblat, and Hachem 13] without any rate results.

Standard ADMM

- Standard ADMM solves a separable problem, where decision variable decomposes into two (linearly coupled) variables:

$$\begin{aligned} \min_{x,y} \quad & f(x) + g(y) \\ \text{s.t.} \quad & Ax + By = c. \end{aligned} \tag{1}$$

- Consider an Augmented Lagrangian function:

$$L_\beta(x, y, p) = f(x) + g(y) - p'(Ax + By - c) + \frac{\beta}{2} ||Ax + By - c||_2^2 .$$

- ADMM: approximate version of classical Augmented Lagrangian method.
  - Primal variables: approximately minimize augmented Lagrangian through a single-pass coordinate descent (in a Gauss-Seidel manner).
  - Dual variable: updated through gradient ascent.

Standard ADMM

More specifically, updates are as follows:

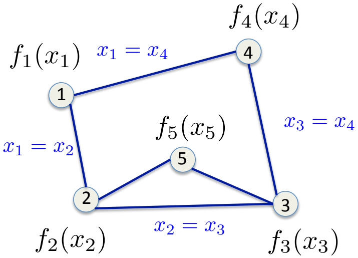
$$
\begin{aligned}
x^{k+1} &= \text{argmin}_x \quad L_\beta(x, y^k, p^k), \\
y^{k+1} &= \text{argmin}_y \quad L_\beta(x^{k+1}, y, p^k), \\
p^{k+1} &= p^k - \beta(Ax^{k+1} - By^{k+1} - c).
\end{aligned}
$$

- Each minimization involves (quadratic perturbations of) functions f and g separately.
    - In many applications, these minimizations are easy (quadratic minimization, l_1 minimization, which arises in Huber fitting, basis pursuit, LASSO, total variation denoising). [Boyd et al. 10]

ADMM for Multi-agent Optimization Problem

- Multi-agent optimization problem can be reformulated in the ADMM framework:
- Consider a set of agents $V = \{1, \ldots, N\}$ connected through an undirected connected graph $G = \{V, E\}$.
- We introduce a local copy x_i for each of the agents and impose $x_i = x_j$ for all $(i, j) \in E$.

$$\min_{x} \quad \sum_{i=1}^{N} f_i(x_i)$$
$$s.t. \quad x_i = x_j, \quad \text{for} \quad (i, j) \in E,$$

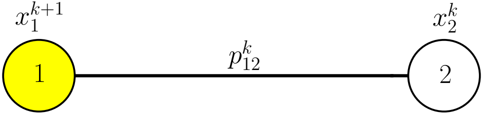


Special Case Study: 2-agent Optimization Problem

- Multi-agent optimization problem with two agents: special case of problem (1):

$$\begin{array}{rl} \min_{x_1,x_2} & f_1(x_1) + f_2(x_2) \\ \text{s.t.} & x_1 = x_2. \end{array}$$

- ADMM applied to this problem yields:



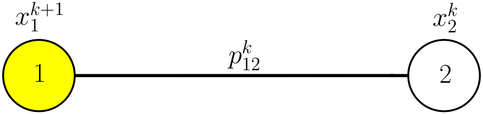
- $x_1^{k+1} = \text{argmin}_{x_1} \quad f_1(x_1) + f_2(x_2^k) - (p_{12}^k)'(x_1 - x_2^k) + \frac{\beta}{2} \left|\left|x_1 - x_2^k\right|\right|_2^2$

Special Case Study: 2-agent Optimization Problem

- Multi-agent optimization problem with two agents: special case of problem (1):

$$\begin{array}{rl} \min_{x_1,x_2} & f_1(x_1) + f_2(x_2) \\ \text{s.t.} & x_1 = x_2. \end{array}$$

- ADMM applied to this problem yields:



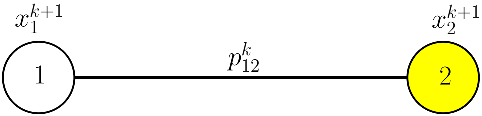
- $x_1^{k+1} = \text{argmin}_{x_1} \quad f_1(x_1) - (p_{12}^k)' x_1 + \frac{\beta}{2} \left|\left| x_1 - x_2^k \right|\right|_2^2$

Special Case Study: 2-agent Optimization Problem

- Multi-agent optimization problem with two agents: special case of problem (1):

$$\begin{array}{rl} \min_{x_1,x_2} & f_1(x_1) + f_2(x_2) \\ \text{s.t.} & x_1 = x_2. \end{array}$$

- ADMM applied to this problem yields:



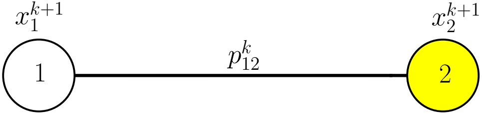
- $x_2^{k+1} =$
$$\text{argmin}_{x_2} \quad f_1(x_1^{k+1}) + f_2(x_2) - (p_{12}^k)'(x_1^{k+1} - x_2) + \frac{\beta}{2}\left|\left|x_1^{k+1} - x_2\right|\right|_2^2$$

Special Case Study: 2-agent Optimization Problem

- Multi-agent optimization problem with two agents: special case of problem (1):

$$\begin{array}{rl} \min_{x_1,x_2} & f_1(x_1) + f_2(x_2) \\ \text{s.t.} & x_1 = x_2. \end{array}$$

- ADMM applied to this problem yields:



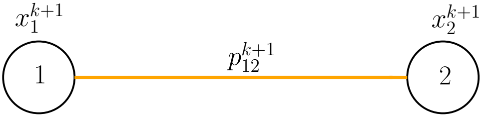
- $x_2^{k+1} = \text{argmin}_{x_2} \quad f_2(x_2) + (p_{12}^k)' x_2 + \frac{\beta}{2} \left|\left| x_1^{k+1} - x_2 \right|\right|_2^2$

Special Case Study: 2-agent Optimization Problem

- Multi-agent optimization problem with two agents: special case of problem (1):

$$\begin{array}{rl} \min_{x_1,x_2} & f_1(x_1) + f_2(x_2) \\ \text{s.t.} & x_1 = x_2. \end{array}$$

- ADMM applied to this problem yields:



- $p^{k+1} = p^k - \beta(x_1^{k+1} - x_2^{k+1})$.

Multi-agent Asynchronous ADMM - Problem Formulation

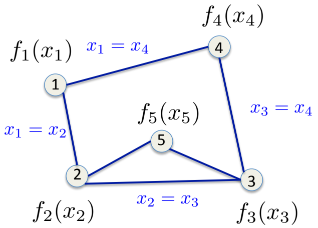
$$\min_{x} \quad \sum_{i=1}^{N} f_i(x_i)$$
$$s.t. \quad x_i = x_j, \quad \text{for} \quad (i,j) \in E.$$

- Reformulate to decouple x_i and x_j by introducing the auxiliary z variable [Bertsekas, Tsitsiklis 89], which allows us to simultaneously update x_i and potentially improves performance.

- Each constraint $x_i - x_j = 0$ for edge $e = (i,j)$ becomes

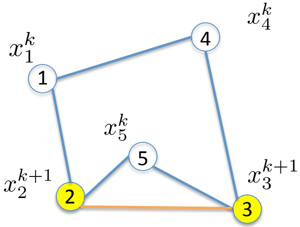
$$x_i = z_{ei}, \quad -x_j = z_{ej},$$

$$z_{ei} + z_{ej} = 0.$$



Multi-agent Asynchronous ADMM - Algorithm

$$\min_{x,z} \quad \sum_{i=1}^{N} f_i(x_i)$$
$$\text{s.t.} \quad x_i = z_{ei}, -x_j = z_{ej} \quad \text{for } (i,j) \in E,$$
$$x \in X, \quad i = 1, \ldots, N,$$
$$z \in Z.$$



- Set $Z = \{z \mid z_{ei} + z_{ej} = 0 \text{ for all } e = (i,j)\}$.
- Write constraint as $Dx = z$, set $E(i)$: the set of edges incident to node i.
- We associate an independent Poisson local clock with each edge.
- At iteration k, if the clock corresponding to edge (i,j) ticks:
  - The constraint $x_i = z_{ei}$, $-x_j = z_{ej}$ (subject to $z_{ei} + z_{ej} = 0$) is active.
  - The agents i and j are active.
  - The dual variables p_{ei} and p_{ej} associated with edge (i,j) are active.

Asynchronous ADMM Algorithm

A Initialization: choose some arbitrary x^0 in X, z^0 in Z and $p^0 = 0$.

B At time step k, an edge $e = (i, j)$ and its end points become active.

   a The active primal variables x_q for $q = i, j$ are updated as

$$x_q^{k+1} \in \operatorname*{argmin}_{x_q \in X_q} f_q(x_q) - \sum_{e \in E(q)} (p_{eq}^k)' D_{eq} x_q + \frac{\beta}{2} \sum_{e \in E(q)} \left|\left| D_{eq} x_q - z_{eq}^k \right|\right|^2.$$

   with $x_w^{k+1} = x_w^k$ for w not active.

   b The active primal variables z_{ei} and z_{ej} are updated as

$$z_{ei}^{k+1}, z_{ej}^{k+1} \in \operatorname*{argmin}_{z_{ei}+z_{ej}=0} \sum_{q=i,j} (p_{eq}^k)' z_{eq} + \frac{\beta}{2} \left|\left| D_{eq} x_q^{k+1} - z_{eq} \right|\right|^2.$$

   with $z_l^{k+1} = z_l^k$ for l not active.

   c The active dual variables p_{eq} for $q = i, j$ are updated as

$$p_{eq}^{k+1} = p_{eq}^k - \beta \left[D_q x_q^{k+1} - z_{eq}^{k+1} \right].$$

- Update in z is a quadratic programming with linear constraint: has closed form solution and can be easily computed in a distributed way.

Asynchronous ADMM Algorithm

A Initialization: choose some arbitrary x^0 in X, z^0 in Z and $p^0 = 0$.

B At time step k, an edge $e = (i, j)$ and its end points become active.

   a The active primal variables x_q for $q = i, j$ are updated as

$$x_q^{k+1} \in \operatorname*{argmin}_{x_q \in X_q} f_q(x_q) - \sum_{e \in E(q)} (p_{eq}^k)' D_{eq} x_q + \frac{\beta}{2} \sum_{e \in E(q)} \left|\left| D_{eq} x_q - z_{eq}^k \right|\right|^2.$$

   with $x_w^{k+1} = x_w^k$ for w not active.

   b The active primal variables z_{ei} and z_{ej} are updated as

$$z_{ei}^{k+1}, z_{ej}^{k+1} \in \operatorname*{argmin}_{z_{ei}+z_{ej}=0} \sum_{q=i,j} (p_{eq}^k)' z_{eq} + \frac{\beta}{2} \left|\left| D_{eq} x_q^{k+1} - z_{eq} \right|\right|^2.$$

   with $z_l^{k+1} = z_l^k$ for l not active.

   c The active dual variables p_{eq} for $q = i, j$ are updated as

$$p_{eq}^{k+1} = p_{eq}^k - \beta \left[D_q x_q^{k+1} - z_{eq}^{k+1} \right].$$

- Update in z is a quadratic programming with linear constraint: has closed form solution and can be easily computed in a distributed way.

Asynchronous ADMM Algorithm

A Initialization: choose some arbitrary x^0 in X, z^0 in Z and $p^0 = 0$.

B At time step k, an edge $e = (i, j)$ and its end points become active.

a For $q = i, j$, the active primal variable x_q is updated as

$$x_q^{k+1} \in \operatorname*{argmin}_{x_q \in X_q} f_q(x_q) - \sum_{e \in E(q)} (p_{eq}^k)' D_{eq} x_q + \frac{\beta}{2} \sum_{e \in E(q)} \left|\left| D_{eq} x_q - z_{eq}^k \right|\right|^2.$$

with $x_w^{k+1} = x_w^k$ for w not active.

b To compute z update,

$$v^{k+1} = \frac{1}{2}(-p_{ei}^k - p_{ej}^k) + \frac{\beta}{2}(D_{ei} x_i^{k+1} + D_{ej} x_j^{k+1}),$$

$$z_{eq}^{k+1} = \frac{1}{\beta}(-p_{eq}^k - v^{k+1}) + D_{eq} x_q^{k+1}.$$

c The active dual variables p_{eq} for $q = i, j$ are updated as

$$p_{eq}^{k+1} = -v^{k+1}.$$

Asynchronous ADMM Algorithm

A Initialization: choose some arbitrary x^0 in X, z^0 in Z and $p^0 = 0$.

B At time step k, an edge $e = (i,j)$ and its end points become active.

a For $q = i, j$, the active primal variable x_q is updated as

$$x_q^{k+1} \in \operatorname*{argmin}_{x_q \in X_q} f_q(x_q) - \sum_{e \in E(q)} (p_{eq}^k)' D_{eq} x_q + \frac{\beta}{2} \sum_{e \in E(q)} \left\| D_{eq} x_q - z_{eq}^k \right\|^2.$$

with $x_w^{k+1} = x_w^k$ for w not active.

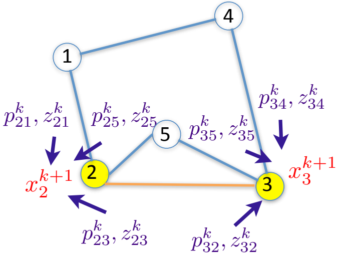
b To compute z update,

$$v^{k+1} = \frac{1}{2}(-p_{ei}^k - p_{ej}^k) + \frac{\beta}{2}(D_{ei} x_i^{k+1} + D_{ej} x_j^{k+1})$$

$$z_{eq}^{k+1} = \frac{1}{\beta}(-p_{eq}^k - v^{k+1}) + D_{eq} x_q^{k+1}.$$

c The active dual variables p_{eq} for $q = i, j$ are updated as

$$p_{eq}^{k+1} = -v^{k+1}.$$



Asynchronous ADMM Algorithm

A Initialization: choose some arbitrary x^0 in X, z^0 in Z and $p^0 = 0$.

B At time step k, an edge $e = (i, j)$ and its end points become active.

a For $q = i, j$, the active primal variable x_q is updated as

$$x_q^{k+1} \in \underset{x_q \in X_q}{\operatorname{argmin}} f_q(x_q) - \sum_{e \in E(q)} (p_{eq}^k)' D_{eq} x_q + \frac{\beta}{2} \sum_{e \in E(q)} \left|\left| D_{eq} x_q - z_{eq}^k \right|\right|^2.$$

with $x_w^{k+1} = x_w^k$ for w not active.

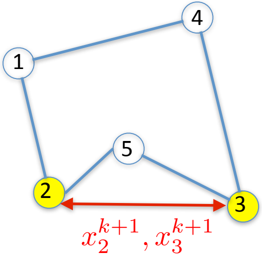
b To compute z update,

$$v^{k+1} = \frac{1}{2}(-p_{ei}^k - p_{ej}^k) + \frac{\beta}{2}(D_{ei} x_i^{k+1} + D_{ej} x_j^{k+1})$$

$$z_{eq}^{k+1} = \frac{1}{\beta}(-p_{eq}^k - v^{k+1}) + D_{eq} x_q^{k+1}.$$

c The active dual variables p_{eq} for $q = i, j$ are updated as

$$p_{eq}^{k+1} = -v^{k+1}.$$



Asynchronous ADMM Algorithm

A Initialization: choose some arbitrary x^0 in X, z^0 in Z and $p^0 = 0$.

B At time step k, an edge $e = (i, j)$ and its end points become active.

a For $q = i, j$, the active primal variable x_q is updated as

$$x_q^{k+1} \in \operatorname*{argmin}_{x_q \in X_q} f_q(x_q) - \sum_{e \in E(q)} (p_{eq}^k)' D_{eq} x_q + \frac{\beta}{2} \sum_{e \in E(q)} \left|\left| D_{eq} x_q - z_{eq}^k \right|\right|^2.$$

with $x_w^{k+1} = x_w^k$ for w not active.

b To compute z update,

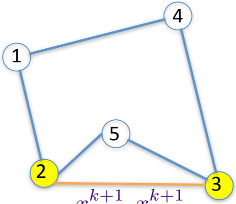
$$v^{k+1} = \frac{1}{2}(-p_{ei}^k - p_{ej}^k) + \frac{\beta}{2}(D_{ei} x_i^{k+1} + D_{ej} x_j^{k+1})$$

$$= -p_{ei}^k + \frac{\beta}{2}(D_{ei} x_i^{k+1} + D_{ej} x_j^{k+1}),$$

$$z_{eq}^{k+1} = \frac{1}{\beta}(-p_{eq}^k - v^{k+1}) + D_{eq} x_q^{k+1}.$$

c The active dual variables p_{eq} for $q = i, j$ are updated as

$$p_{eq}^{k+1} = -v^{k+1}.$$



x_2^{k+1}, x_3^{k+1}

$p_{23}^k = p_{32}^k$

$v^{k+1} =$

$- p_{23}^k + \frac{\beta}{2}(D_{23} x_2^{k+1} + D_{32} x_3^{k+1})$

Asynchronous ADMM Algorithm

A Initialization: choose some arbitrary x^0 in X, z^0 in Z and $p^0 = 0$.

B At time step k, an edge $e = (i, j)$ and its end points become active.

a For $q = i, j$, the active primal variable x_q is updated as

$$x_q^{k+1} \in \operatorname*{argmin}_{x_q \in X_q} f_q(x_q) - \sum_{e \in E(q)} (p_{eq}^k)' D_{eq} x_q + \frac{\beta}{2} \sum_{e \in E(q)} \left|\left| D_{eq} x_q - z_{eq}^k \right|\right|^2.$$

with $x_w^{k+1} = x_w^k$ for w not active.

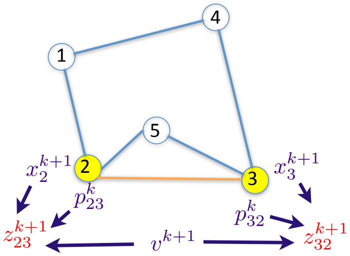
b To compute z update,

$$v^{k+1} = -p_{ei}^k + \frac{\beta}{2}(D_{ei} x_i^{k+1} + D_{ej} x_j^{k+1}),$$

$$z_{eq}^{k+1} = \frac{1}{\beta}(-p_{eq}^k - v^{k+1}) + D_{eq} x_q^{k+1}.$$

c The active dual variables p_{eq} for $q = i, j$ are updated as

$$p_{eq}^{k+1} = -v^{k+1}.$$



Asynchronous ADMM Algorithm

A Initialization: choose some arbitrary x^0 in X, z^0 in Z and $p^0 = 0$.

B At time step k, an edge $e = (i, j)$ and its end points become active.

a For $q = i, j$, the active primal variable x_q is updated as

$$x_q^{k+1} \in \operatorname*{argmin}_{x_q \in X_q} f_q(x_q) - \sum_{e \in E(q)} (p_{eq}^k)' D_{eq} x_q + \frac{\beta}{2} \sum_{e \in E(q)} \left|\left| D_{eq} x_q - z_{eq}^k \right|\right|^2.$$

with $x_w^{k+1} = x_w^k$ for w not active.

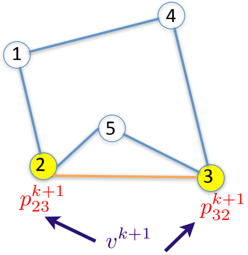
b To compute z update,

$$v^{k+1} = -p_{ei}^k + \frac{\beta}{2}(D_{ei} x_i^{k+1} + D_{ej} x_j^{k+1}),$$

$$z_{eq}^{k+1} = \frac{1}{\beta}(-p_{eq}^k - v^{k+1}) + D_{eq} x_q^{k+1}.$$

c The active dual variables p_{eq} for $q = i, j$ are updated as

$$p_{eq}^{k+1} = -v^{k+1}.$$



- Generalizes to any linear constraint $Dx + Hz = 0$.

Convergence

Assumption

(a) *(Infinitely often updates): For all k and all l in the set of linear constraints,* $\mathbb{P}(l \text{ is active at time } k) > 0$.

Theorem

Let $\{x^k, z^k, p^k\}$ be the iterates generated by the general asynchronous ADMM algorithm. The sequence $\{x^k, z^k, p^k\}$ converges to a saddle point (x^*, z^*, p^*) of the Lagrangian, i.e., (x^k, z^k) converges to a primal optimal solution (x^*, z^*) almost surely.

Proof Sketch

- Define auxiliary full information iterates y^k, v^k and μ^k.

$$y^{k+1} \in \operatorname*{argmin}_{y \in X} \sum_{i=1}^{N} f_i(y_i) - (p^k - \beta H z^k)' D_i y + \frac{\beta}{2} \left|\left| D_i y \right|\right|^2,$$

$$v^{k+1} \in \operatorname*{argmin}_{v \in Z} \sum_{l=1}^{W} -(p^k - \beta D y^{k+1})' H_l v + \frac{\beta}{2} \left|\left| H_l v \right|\right|^2,$$

$$\mu^{k+1} = p^k - \beta(D y^{k+1} + H v^{k+1}).$$

Convergence Analysis – Idea

- Active components of asynchronous iterates take the same value as full information iterates, inactive components remain at their previous value.
- Using the Lyapunov function $\frac{1}{2\beta}\left|\left|p^{k+1}-p^*\right|\right|^2+\frac{\beta}{2}\left|\left|H(z^{k+1}-z^*)\right|\right|^2$, we can show full information iterates converge to an optimal solution.
- To develop a Lyapunov function for the asynchronous iterates, define probabilities

$$\lambda_l = \mathbb{P}(l \text{ is active at time } k)$$

  and weighted norm induced by matrix $\bar{\Lambda}$ where $\bar{\Lambda}_{ll} = 1/\lambda_l$.
- Using supermartingale arguments, we show that the probability adjusted norm,

$$\frac{1}{2\beta}\left|\left|p^{k+1}-p^*\right|\right|_{\bar{\Lambda}}^2+\frac{\beta}{2}\left|\left|H(z^{k+1}-z^*)\right|\right|_{\bar{\Lambda}}^2$$

  serves as a Lyapunov function for the asynchronous iterates.

Rate of Convergence

Assumption

(a) *(Compact constraint set): Sets X and Z are compact.*

- Ergodic average: $\bar{x}_i(k) = \frac{\sum_{t=1}^{k} x_i^t}{k}$, for all i, $\bar{z}_l(T) = \frac{\sum_{t=1}^{k} z_l^t}{k}$, for all l.

Theorem

For $F(x) = \sum_{i=1}^{N} f_i(x_i)$, the iterates generated by the asynchronous ADMM algorithm satisfies

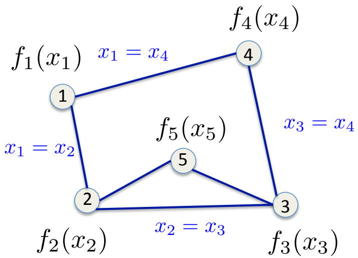
$$||\mathbb{E}(F(\bar{x}(k))) - F(x^*)|| \leq \frac{\alpha}{k},$$

where $\alpha = ||p^*||_\infty \left[\bar{Q} + \tilde{L}^0 + \frac{1}{2\beta}\left|\left|p^0 - p^*\right|\right|_{\bar{\Lambda}}^2 + \frac{\beta}{2}\left|\left|H(z^0 - z^*)\right|\right|_{\bar{\Lambda}}^2\right] +$ $\left[Q(p^*) + \tilde{L}(x^0, z^0, p^*) + \frac{1}{2\beta}\left|\left|p^0 - \bar{\theta}\right|\right|_{\bar{\Lambda}}^2 + \frac{\beta}{2}\left|\left|H(z^0 - z^*)\right|\right|_{\bar{\Lambda}}^2\right]$*, for some scalar* Q, $\bar{Q}$, $\bar{\Lambda}$, $\bar{\theta}$*, related to* p^* *and size of set* X *and* Z*.*

- A similar rate result holds for the constraint violation $||\mathbb{E}(D\bar{x}(k) + H\bar{z}(k))||$.

Simulations

- Sample Network:



- Asynchronous ADMM algorithm is compared against a gradient based asynchronous gossip algorithm [Ram, Nedic, Veeravalli 09]
- Tested in three 5−node graphs: sample network, line graph and complete graph.

Sample network

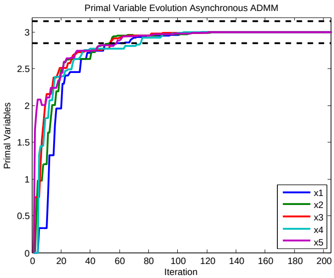


Figure: ADMM for the sample network.

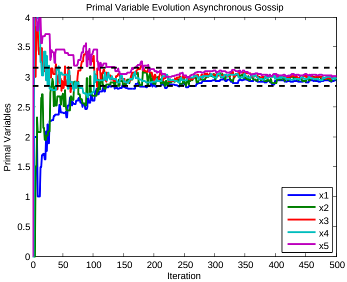


Figure: Asynchronous gossip for the sample network.

To reach 5% neighborhood of the optimal solution: asynchronous ADMM takes 80 iterations, asynchronous gossip takes 250 iterations.

Line Graph

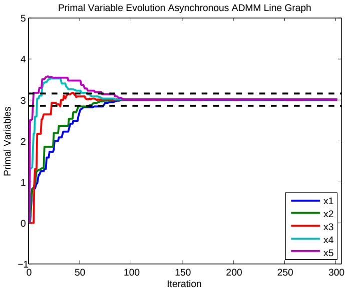


Figure: ADMM for the line graph.

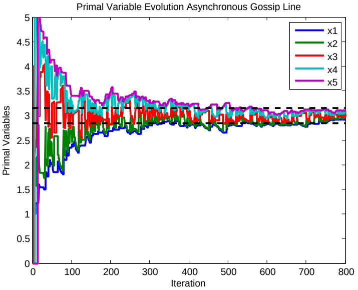
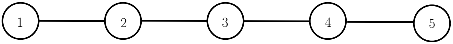


Figure: Asynchronous gossip for the line graph.



To reach 5% neighborhood of the optimal solution: asynchronous ADMM takes 70 iterations, asynchronous gossip takes 700 iterations.

Complete Graph

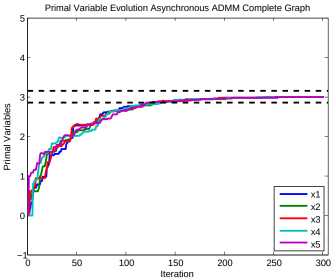


Figure: ADMM for the complete graph.

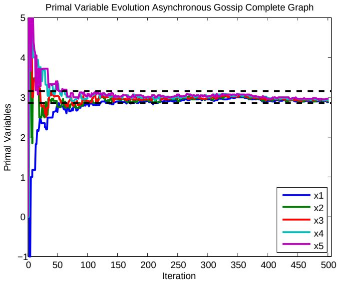


Figure: Asynchronous gossip for the complete network.

To reach 5% neighborhood of the optimal solution: asynchronous ADMM takes 140 iterations, asynchronous gossip takes 380 iterations.

Image denoising

Given a noisy image measure b, recover the original image by solving the following problem:

$$\min_x \frac{1}{2} ||x - b||_2^2 + \lambda ||x||_{TV},$$

where $||x||_{TV} = \sum_{i \sim j} |x_i - x_j|$.

Figure: Original cameraman figure.

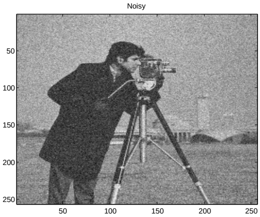


Figure: Added white noise with standard deviation 25.

Image denoising

Image data b_i is available at two different sensors.

Figure: Original cameraman figure.

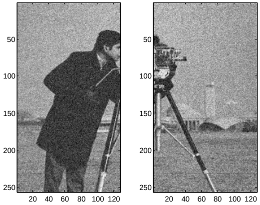


Figure: Noisy image data in 2 parts.

Image denoising

Recover the original image by solving the following problem:

$$\min_{x} \frac{1}{2} ||x - b_1||_2^2 + \frac{1}{2} ||x - b_2||_2^2 + \lambda ||x||_{TV},$$

with asynchronous ADMM algorithm with 3 agents. Algorithm converged after 87 iterations, 35 seconds on laptop.

Figure: Original cameraman figure.

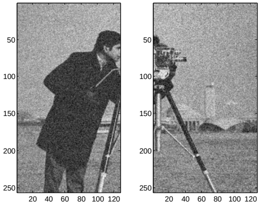


Figure: Noisy image data in 2 parts.

Figure: Recovered using total variation denoising formula with $\lambda = 20$.

Conclusions and Future Work

- For general convex problems, we developed an asynchronous distributed ADMM algorithm, which converges at the best known rate $O(1/k)$.
- Simulation results illustrate the superior performance of ADMM (even for network topologies with slow mixing).
- Ongoing and Future Work:
    - Online and dynamic distributed optimization problems.
    - ADMM type algorithm for time-varying graph topology.
    - Analyze network effects on ADMM algorithm.