Distributed Alternating Direction Method of Multipliers for Multi-agent Optimization

Asu Ozdaglar

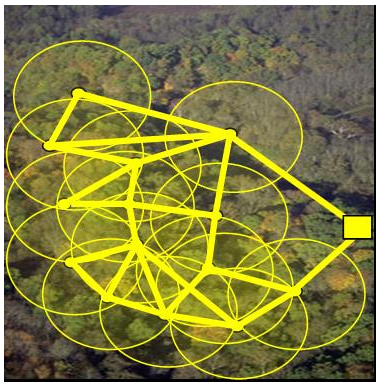
Laboratory for Information and Decision Systems
Operations Research Center
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology

Lund Workshop on Dynamics and Control in Networks
October, 2014

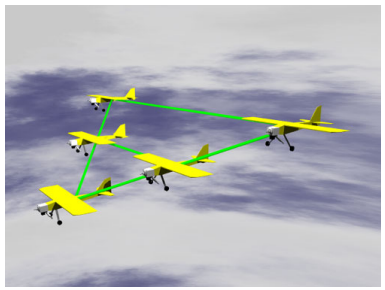
Motivation

- 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.

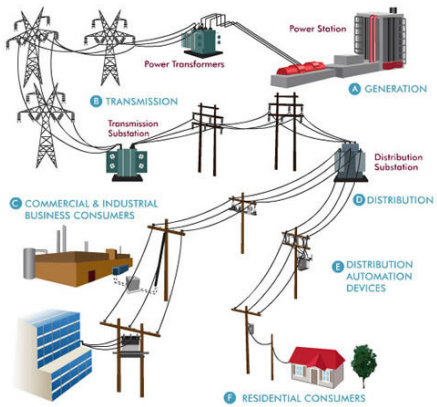
Routing and congestion control in wireline and wireless networks



Parameter estimation in sensor networks



Multi-agent cooperative control and coordination



Smart grid systems

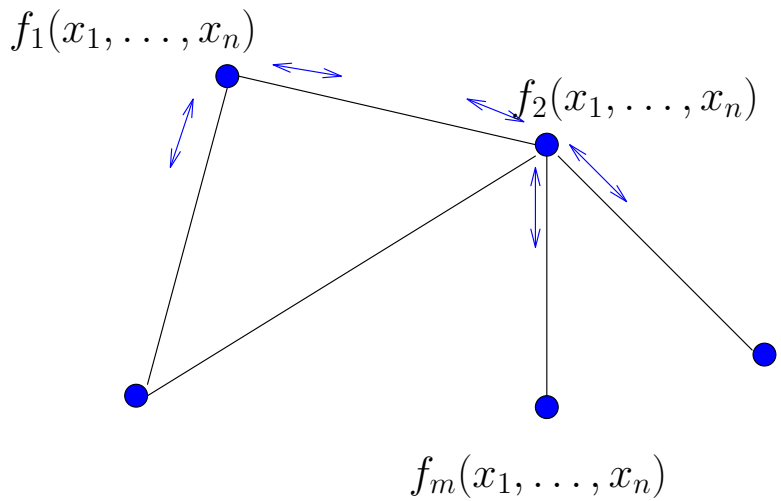
Distributed Multi-agent Optimization

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

- 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 only with neighbors.

Machine Learning Example

- 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].

Literature: Parallel and Distributed Optimization

- Lagrangian relaxation and dual optimization methods:
  - Dual gradient ascent, coordinate ascent methods.
- Parallel computation and optimization:
  - [Tsitsiklis, Bertsekas, Athans 86], [Bertsekas and Tsitsiklis 89].
- Consensus and cooperative control:
  - [Jadbabaie, Lin, Morse 03], [Olfati-Saber, Murray 04], [Boyd et al. 05], [Olshevsky, Tsitsiklis 07], [Fagnani, Zampieri 09].
- Multi-agent optimization
  - Distributed primal subgradient methods [Nedic, Ozdaglar 07].
  - Distributed dual averaging methods [Duchi, Agarwal Wainwright 12].
  - These methods converge at the rate $O(1/\sqrt{k})$, where k is the number of iterations.

Literature: Alternating Direction Method of Multipliers

- A large literature on alternating direction method of multipliers (ADMM) due to fast computational performance and distributed-memory, parallel implementations:
  - [Glowinski, Marrocco 75], [Fortin, Glowinski 83], [Gabay 83], [Eckstein, Bertsekas 92].
  - Recent tutorials: [Boyd et al. 10], [Eckstein 12].
  - Relation to proximal point algorithm: [Rockafellar 76, 76], [Luque 84].
  - Decentralized estimation and compressive sensing applications: [Schizas, Ribeiro, Giannakis 08], [Mota, Xavier, Aguiar, Puschel 11].

This Talk

- We present distributed ADMM-type algorithms for multi-agent optimization.
  - Distributed ADMM over undirected networks [Shtern, Wei, and Ozdaglar 14].
  - Distributed ADMM over directed networks [Makhdoumi and Ozdaglar 14].
- In both cases, we show that these algorithms converge at the faster rate $O(1/k)$.

Standard ADMM

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

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

- 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 some applications, these minimizations are easy (quadratic minimization, l_1 minimization, which arises in Huber fitting, basis pursuit, LASSO, total variation denoising).
- Best known convergence rate: $O(1/k)$ [He, Yuan 11]. [1]

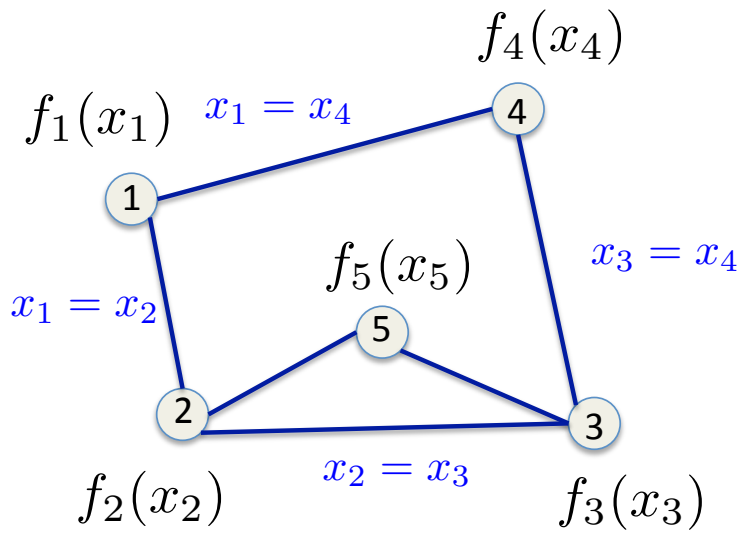
[1] Under stronger assumptions (strong convexity, Lipschitz gradient), ADMM converges linearly [Goldfarb et. al 10], [Deng, Yin 12], [Hong, Luo 12].

ADMM for Multi-agent Optimization Problem

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

$$\min_{x} \quad \sum_{i=1}^{N} f_i(x_i)$$

$$s.t. \quad x_i = x_j, \quad \text{for} \quad (i, j) \in E,$$

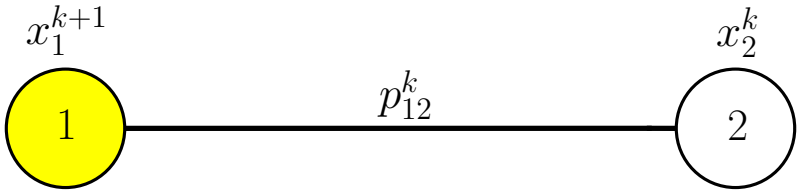


Special Case Study: 2-agent Optimization Problem

- Multi-agent optimization problem with two agents:

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

- ADMM applied to this problem yields:



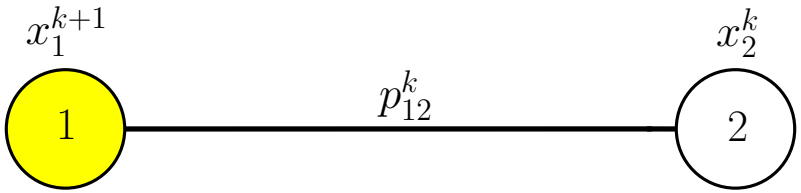
- $x_1^{k+1} = \text{argmin}_{x_1} \quad f_1(x_1) + f_2(x_2^k) - (p_{12}^k)'(x_1 - x_2^k) + \frac{\beta}{2}\left|\left|x_1 - x_2^k\right|\right|_2^2$

Special Case Study: 2-agent Optimization Problem

- Multi-agent optimization problem with two agents:

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

- ADMM applied to this problem yields:



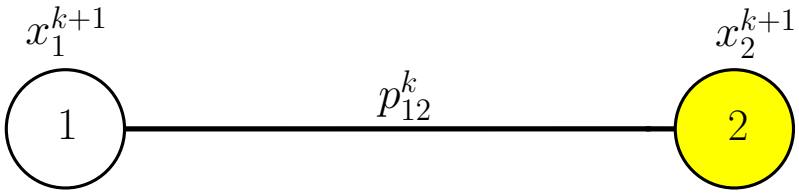
- $x_1^{k+1} = \text{argmin}_{x_1} \quad f_1(x_1) - (p_{12}^k)' x_1 + \frac{\beta}{2} \left|\left| x_1 - x_2^k \right|\right|_2^2$

Special Case Study: 2-agent Optimization Problem

- Multi-agent optimization problem with two agents:

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

- ADMM applied to this problem yields:



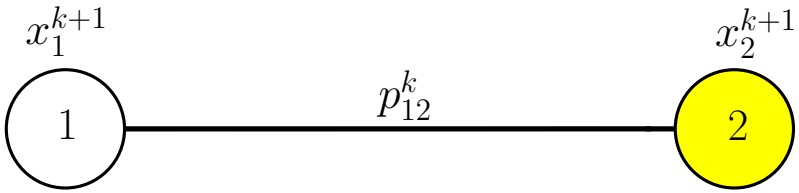
- $x_2^{k+1} = \text{argmin}_{x_2} \quad f_1(x_1^{k+1}) + f_2(x_2) - (p_{12}^k)'(x_1^{k+1} - x_2) + \frac{\beta}{2} \left|\left| x_1^{k+1} - x_2 \right|\right|_2^2$

Special Case Study: 2-agent Optimization Problem

- Multi-agent optimization problem with two agents:

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

- ADMM applied to this problem yields:



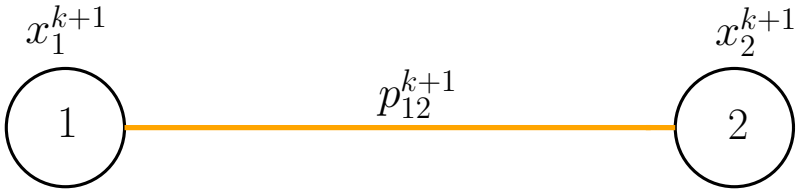
- $x_2^{k+1} = \text{argmin}_{x_2} \quad f_2(x_2) + (p_{12}^k)' x_2 + \frac{\beta}{2} \left|\left| x_1^{k+1} - x_2 \right|\right|_2^2$

Special Case Study: 2-agent Optimization Problem

- Multi-agent optimization problem with two agents:

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

- ADMM applied to this problem yields:



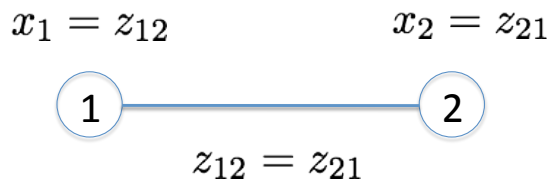
- $p_{12}^{k+1} = p_{12}^{k} - \beta(x_1^{k+1} - x_2^{k+1})$.

Multi-agent Optimization Problem: Reformulation

- Requires a globally known order on the agents [Wei, Ozdaglar 12].
- Reformulate to remove ordering: technique from [Bertsekas, Tsitsiklis 89].
- Rewrite each constraint $x_i - x_j = 0$ for edge $e = (i, j)$ as

$$x_i = z_{ij}, \quad x_j = z_{ji},$$

$$z_{ij} = z_{ji}.$$



- The reformulated problem can be written compactly as

$$\min_{x \in \mathbb{R}^{Nn}, z \in Z} \quad F(x) = \sum_{i=1}^{N} f_i(x_i) \qquad (1)$$

$$s.t. \quad Dx + z = 0.$$

where $Z = \{z \in \mathbb{R}^{2Mn} \mid z_{ij} = z_{ji}, \text{ for } (i, j) \text{ in } E\}$.

- Assumption: The optimal solution set of this problem is nonempty.

ADMM for Multi-agent Optimization

a The primal variable x is updated as

$$x^{k+1} \in \underset{x}{\operatorname{argmin}} F(x) - (p^k)' Dx + \frac{\beta}{2} \left|\left| Dx + z^k \right|\right|^2.$$

b The primal variable z is updated as

$$z^{k+1} \in \underset{z \in Z}{\operatorname{argmin}} -(p^k)' Hz + \frac{\beta}{2} \left|\left| Dx^{k+1} + z \right|\right|^2.$$

c The dual variable p is updated as

$$p^{k+1} = p^k - \beta(Dx^{k+1} + z^{k+1}).$$

- The minimizers in the update equations exist (since matrix $D'D$ has full rank).
- Update in z is a quadratic program with linear constraint: has a closed form solution and can be computed using communication between neighboring nodes.

Distributed Implementation

- Each agent i maintains x_i^k and p_{ij}^k, z_{ij}^k, for $j \in \mathcal{N}(i)$.
- At iteration k,
  - a Agent i updates the primal variable x_i^k as

$$x_i^{k+1} \in \operatorname*{argmin}_{x_i} f_i(x_i) - \sum_{j\in\mathcal{N}(i)} (p_{ij}^k)' x_i + \frac{\beta}{2} \sum_{j\in\mathcal{N}(i)} \left|\left| x_i - z_{ij}^k \right|\right|^2.$$

    The value x_i^{k+1} is sent to all neighbors.

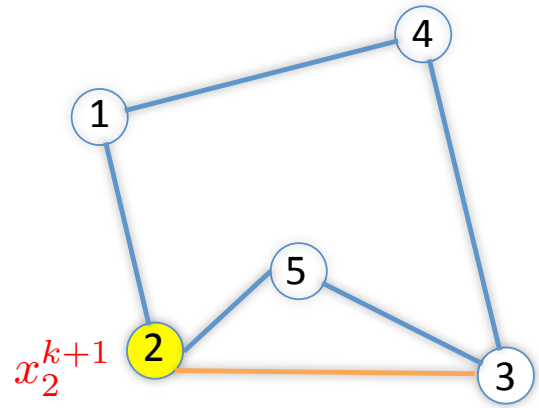
  - b Agent i computes

$$z_{ij}^{k+1} = z_{ji}^{k+1} = \frac{1}{2}(x_i^{k+1} + x_j^{k+1}).$$

  - c Agent i computes

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

    for all neighbors in j in $\mathcal{N}(i)$.



$p_{2j}^k, z_{2j}^k, j = \{1, 3, 5\}$

Distributed Implementation

- Each agent i maintains x_i^k and p_{ij}^k, z_{ij}^k, for $j \in \mathcal{N}(i)$.
- At iteration k,
  - a Agent i updates the primal variable x_i^k as

$$x_i^{k+1} \in \operatorname*{argmin}_{x_i} f_i(x_i) - \sum_{j\in\mathcal{N}(i)} (p_{ij}^k)' x_i + \frac{\beta}{2} \sum_{j\in\mathcal{N}(i)} \left|\left| x_i - z_{ij}^k \right|\right|^2$$

    The value x_i^{k+1} is sent to all neighbors.

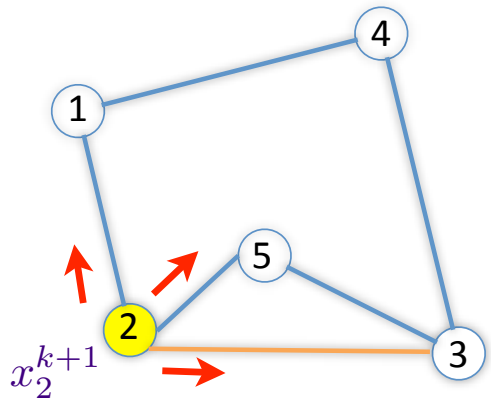
  - b Agent i computes

$$z_{ij}^{k+1} = z_{ji}^{k+1} = \frac{1}{2}(x_i^{k+1} + x_j^{k+1}).$$

  - c Agent i computes

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

    for all neighbors in j in $\mathcal{N}(i)$.



Distributed Implementation

- Each agent i maintains x_i^k and p_{ij}^k, z_{ij}^k, for $j \in \mathcal{N}(i)$.
- At iteration k,
  - a Agent i updates the primal variable x_i^k as

$$x_i^{k+1} \in \operatorname*{argmin}_{x_i} f_i(x_i) - \sum_{j\in\mathcal{N}(i)} (p_{ij}^k)' x_i + \frac{\beta}{2} \sum_{j\in\mathcal{N}(i)} \left|\left| x_i - z_{ij}^k \right|\right|^2.$$

    The value x_i^{k+1} is sent to all neighbors.

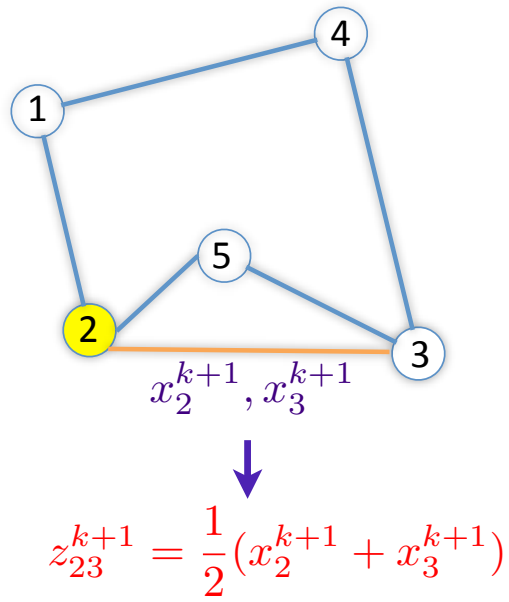
  - b Agent i computes

$$z_{ij}^{k+1} = z_{ji}^{k+1} = \frac{1}{2}(x_i^{k+1} + x_j^{k+1}).$$

  - c Agent i computes

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

    for all neighbors in j in $\mathcal{N}(i)$.



Distributed Implementation

- Each agent i maintains x_i^k and p_{ij}^k, z_{ij}^k, for $j \in \mathcal{N}(i)$.
- At iteration k,
  - a Agent i udpates the primal variable x_i^k as

$$x_i^{k+1} \in \operatorname*{argmin}_{x_i} f_i(x_i) - \sum_{j\in\mathcal{N}(i)} (p_{ij}^k)' x_i + \frac{\beta}{2} \sum_{j\in\mathcal{N}(i)} \left|\left| x_i - z_{ij}^k \right|\right|^2.$$

    The value x_i^{k+1} is sent to all neighbors.

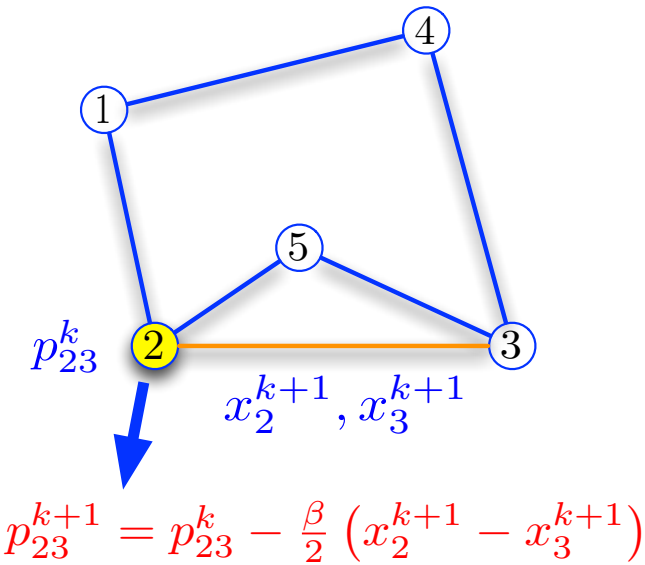
  - b Agent i computes

$$z_{ij}^{k+1} = z_{ji}^{k+1} = \frac{1}{2}(x_i^{k+1} + x_j^{k+1}).$$

  - c Agent i computes

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

    for all neighbors in j in $\mathcal{N}(i)$.



Rate of Convergence

- Let $\{x^k, z^k, p^k\}$ be the sequence generated by the synchronous ADMM algorithm. We define the ergodic averages:

$$\bar{x}^k = \frac{1}{k}\sum_{l=0}^{k-1} x^k, \qquad \bar{z}^k = \frac{1}{k}\sum_{l=0}^{k-1} z^k.$$

We also define the ergodic time average of the residual: $\bar{r}^k = D\bar{x}^k + \bar{z}^k$.

Theorem

Let (x^*, z^*, p^*) be a primal-dual optimal solution for problem (1). The following hold at each iteration k:

$$|F(\bar{x}^k) - F(x^*)| \leq \frac{1}{2\beta k}\left(\beta^2 \left|\left|z^0 - z^*\right|\right|^2 + \max\left\{\left|\left|p^0\right|\right|^2, \left|\left|p^0 - 2p^*\right|\right|^2\right\}\right),$$

$$||\bar{r}(k)|| \leq \frac{1}{2\beta k}\left(\beta^2 \left|\left|z^0 - z^*\right|\right|^2 + \left(\left|\left|p^0 - p^*\right|\right| + 1\right)^2\right).$$

Proof Idea

- Using optimality conditions and dual update, we obtain for all $p \in \mathbb{R}^{2Mn}$,

$$F(x^{k+1}) - F(x^*) - p' r^{k+1} \leq \frac{1}{2\beta}\left(\left|\left|p^k - p\right|\right|^2 - \left|\left|p^{k+1} - p\right|\right|^2\right) + \frac{\beta}{2}\left(\left|\left|z^k - z^*\right|\right|^2 - \left|\left|z^{k+1} - z^*\right|\right|^2\right).$$

- First bound follows from adding this over a window and using convexity to generate function values at ergodic averages.
- Second bound follows from picking $p = p^* - \frac{\bar{r}^k}{||\bar{r}^k||}$.
- One can use $\left|\left|p^k - p^*\right|\right|^2 + \beta^2 \left|\left|z^k - z^*\right|\right|^2$ as a Lyapunov function to show that the sequence $\{x^k, z^k, p^k\}$ converges to a primal-dual optimal solution of problem (1).

Network Effect

- The performance depends on the network topology through $\|p^*\|$.

Theorem

Let (x^*, z^*, p^*) be an optimal primal-dual solution for problem (1). Then,

$$||p^*||^2 \leq \frac{2C^2}{\rho_2(L(G))},$$

where C is a bound on the norms of all vectors in $\partial F(x^*)$ and $\rho_2(L(G))$ is the second smallest positive eigenvalue of the Laplacian matrix $L(G)$ of the underlying graph.[2]

- The performance depends on the algebraic connectivity of the graph: the more connected it is, the larger $\rho_2(L(G))$ is, the better the performance.

[2] $L(G)$ is a matrix with elements $[L(G)]_{ij} = \begin{cases} \text{degree}(i) & i = j, \\ -1 & (i,j) \in E. \end{cases}$

Dependence of Convergence Rate on Network

Theorem

Let (x^*, z^*) be a primal optimal solution for problem (1). Then with initialization $x^0 = 0$, $z^0 = 0$ and $p^0 = 0$, we have

$$|F(\bar{x}^k) - F(x^*)| \leq \frac{M\beta \left|\left|x^*\right|\right|^2}{kN} + \frac{4C^2}{k\beta\rho_2(L(G))},$$

$$\left|\left|\bar{r}^k\right|\right| \leq \frac{1}{2k\beta}\left(\frac{C\sqrt{2}}{\sqrt{\rho_2(L(G))}} + 1\right)^2 + \frac{M\beta \left|\left|x^*\right|\right|^2}{kN},$$

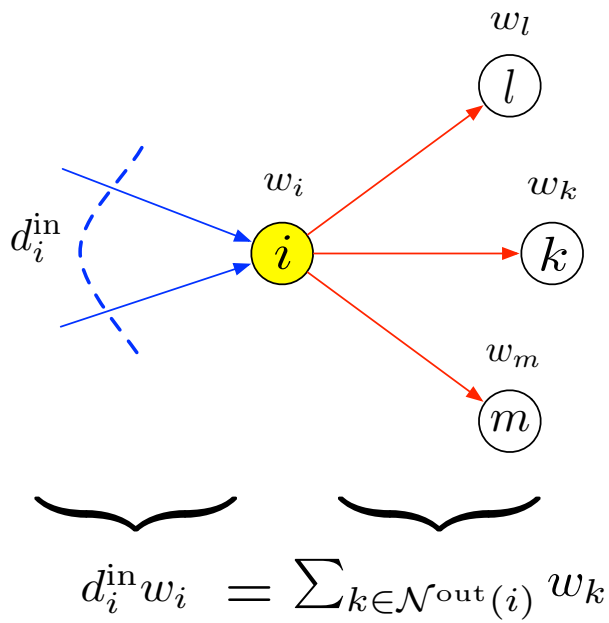
where M is the number of edges.

Distributed ADMM over Directed Networks

- So far, we have assumed communication among nodes bidirectional.
- This does not hold when nodes have different local interference patterns and different power transmission levels.
- Question: Can we design a distributed ADMM-like algorithm over directed networks with fast convergence? [Makhdoumi, Ozdaglar 14]
  - Every agent updates primal and dual variables based on information received from incoming links.
- Key idea: Normalize the information entering a node with the *right weight* so that in the limit the network becomes balanced (i.e., amount of information entering and leaving a node is equalized).
- Two approaches:
  - Push-sum [Kempe, Dobra, Gehrke 03], [Nedich, Olshevsky 13,14].
    - Leads to nonlinear updates and a non convergent algorithm for ADMM.
  - Graph balancing: [Gharesifard, Cortes 09], [Priolo et al. 13], [Rikos et al. 14].

Weight Balanced Graph

- Let $G = (V, E)$ be a strongly connected directed network.
- $\mathcal{N}^{\text{in}}(i)$: in-neighbors of node i; $\mathcal{N}^{\text{out}}(i)$: out-neighbors of node i.
- Each node has a weight w_i representing how much he scales his incoming information.
- The node weights $(w_1, \ldots, w_N)$ make the network balanced if for each node i, total weight of i's incoming information is equal to the total weight of its outgoing information.



Distributed Balancing

- Laplacian of the directed graph L: $L_{ii} = d_i^{\text{in}}$ and $L_{ij} = -1$, $j \in \mathcal{N}^{\text{in}}(i)$.
- The balancing weight vector $\mathbf{w} = (w_1, \ldots, w_N)'$ is the left eigenvector of L corresponding to eigenvalue 0, i.e., $\mathbf{w}'L = 0$; or for $i = 1, \ldots, N$,

$$(w_1, \ldots, w_i, \ldots, w_N) \begin{bmatrix} & -1 & \\ & \vdots & \\ \cdots & d_i^{\text{in}} & \cdots \\ & \vdots & \\ & \underbrace{\quad\quad} & \\ & i\text{th column} & \end{bmatrix} \Rightarrow d_i^{\text{in}} w_i = \sum_{k \in \mathcal{N}^{\text{out}}(i)} w_k.$$

- We will use a slight variation of [Priolo et al. 13] to design a distributed balancing algorithm.
- For each node i, let $0 < \delta_i < \frac{1}{d_i^{\text{in}}}$.
- Let $C = I - \Delta L$, where $\Delta = \text{diag}(\delta_1, \ldots, \delta_N)$.

Distributed Balancing Algorithm

Each agent i maintains weight vector $\tilde{\mathbf{w}}_i^k \in \mathbb{R}^N$. At iteration k:

- The weight matrix $\tilde{W}^k = [\tilde{\mathbf{w}'}_1^k; \ldots; \tilde{\mathbf{w}'}_N^k]$ is updated as

$$\tilde{W}^{k+1} = C\tilde{W}^k.$$

- This corresponds to each agent i updating his weight vector $\tilde{\mathbf{w}}_i^k$ using local information from his in-neighbors:

$$\tilde{\mathbf{w}'}_i^{k+1} = (1 - \delta_i d_i^{in})\tilde{\mathbf{w}'}_i^k + \sum_{j \in N^{\text{in}}(i)} \delta_i \tilde{\mathbf{w}'}_j^k.$$

- Agent i sends $\tilde{\mathbf{w}}_i^{k+1}$ to all his out-neighbors.

Convergence of Balancing Algorithm

Theorem

Starting from $\tilde{W}^0 = I$, *we have*

$$\lim_{k\to\infty} \tilde{W}^k = \lim_{k\to\infty} C^k = \mathbf{1}\mathbf{v}' = \begin{bmatrix} v_1 & v_2 & \cdots & v_N \\ v_1 & v_2 & \cdots & v_N \\ \vdots & \vdots & \ddots & \vdots \\ v_1 & v_2 & \cdots & v_N \end{bmatrix},$$

where $\mathbf{v}'$ *is the left eigenvector of* C *corresponding to eigenvalue 1. Convergence is exponentially fast with exponent given by the second largest eigenvalue of* L.

Proof: Follows from Perron-Frobenius Theorem (C is a primitive matrix since G is strongly connected).

- Hence the weight vector of agent i converges to $\tilde{\mathbf{w}}_i^\infty = [v_1, \ldots, v_N]'$.
- The balancing weight for agent i is $w_i^\infty = \delta_i \tilde{\mathbf{w}}_i^\infty(i) = \delta_i v_i$.

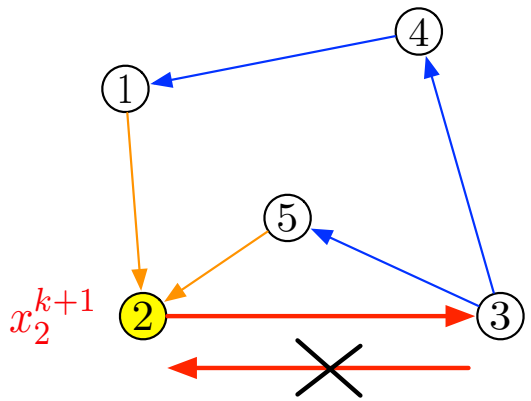
$$v_i = (1 - \delta_i d_i^{\text{in}}) v_i + \sum_{j \in N^{\text{out}}(i)} \delta_j v_j \Rightarrow d_i^{\text{in}} \underbrace{\delta_i v_i}_{w_i^\infty} = \sum_{j \in N^{\text{out}}(i)} \underbrace{\delta_j v_j}_{w_j^\infty}$$

Implementation of ADMM over Directed Graphs

- Consider the distributed ADMM iteration over undirected networks.
- At iteration k, agent i updates the primal variable x_i^k as

$$x_i^{k+1} \in \operatorname*{argmin}_{x_i} f_i(x_i) - \sum_{j\in\mathcal{N}^{\text{in}}(i)} (p_{ij}^k)' x_i + \frac{\beta}{2} \sum_{j\in\mathcal{N}^{\text{in}}(i)} \left|\left| x_i - z_{ij}^k \right|\right|^2 - \sum_{j\in\mathcal{N}^{\text{out}}(i)} (p_{ij}^k)' x_i + \frac{\beta}{2} \sum_{j\in\mathcal{N}^{\text{out}}(i)} \left|\left| x_i - z_{ij}^k \right|\right|^2.$$

- This requires node i to receive information from $j \in \mathcal{N}^{\text{out}}(i)$.

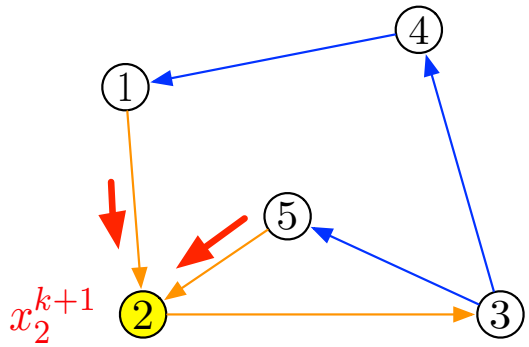


Implementation of ADMM over Directed Graphs

- Consider the distributed ADMM iteration over undirected networks.
- At iteration k, agent i updates the primal variable x_i^k as

$$x_i^{k+1} \in \operatorname*{argmin}_{x_i} f_i(x_i) - \sum_{j \in \mathcal{N}^{\text{in}}(i)} (p_{ij}^k)' x_i + \frac{\beta}{2} \sum_{j \in \mathcal{N}^{\text{in}}(i)} \left\| x_i - z_{ij}^k \right\|^2 .$$

- Fix:
  - Use only incoming information in the update.

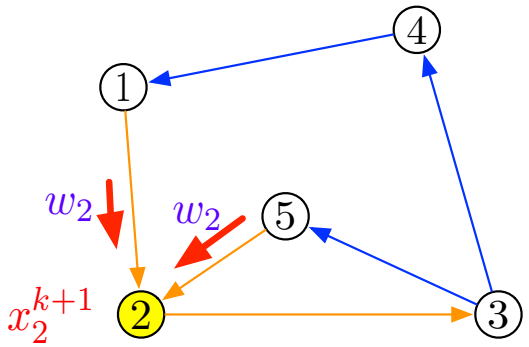


Implementation of ADMM over Directed Graphs

- Consider the distributed ADMM iteration over undirected networks.
- At iteration k, agent i updates the primal variable x_i^k as

$$x_i^{k+1} \in \operatorname*{argmin}_{x_i} f_i(x_i) - w_i \sum_{j \in \mathcal{N}^{\text{in}}(i)} (p_{ij}^k)' x_i + w_i \frac{\beta}{2} \sum_{j \in \mathcal{N}^{\text{in}}(i)} \left\| x_i - z_{ij}^k \right\|^2 .$$

- Fix:
  - Use only incoming information in the update.
  - Scale incoming information by balancing weight w_i.
  - Update w_i^k using distributed balancing in the same time scale.



Distributed ADMM over Directed Networks

- Each agent i maintains x_i^k, $\tilde{\mathbf{w}}_i^k$ (w_i^k) and p_{ij}^k, z_{ij}^k, for $j \in \mathcal{N}^{\text{in}}(i)$.
- At iteration k,
  - a Agent i updates the primal variable x_i^k as

$$x_i^{k+1} \in \operatorname*{argmin}_{x_i} f_i(x_i) - w_i^k \sum_{j\in\mathcal{N}^{\text{in}}(i)} (p_{ij}^k)' x_i + w_i^k \frac{\beta}{2} \sum_{j\in\mathcal{N}^{\text{in}}(i)} \left|\left| x_i - z_{ij}^k \right|\right|^2,$$

    The values x_i^{k+1} and $\tilde{\mathbf{w}}_i^k$ are sent to all out-neighbors.

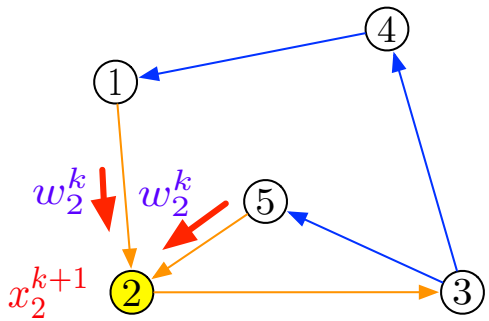
  - b For all $j \in \mathcal{N}^{\text{in}}(i)$, agent i computes

$$z_{ij}^{k+1} = \frac{1}{2}(x_i^{k+1} + x_j^{k+1}),$$

$$p_{ij}^{k+1} = p_{ij}^k - \frac{\beta}{2}(x_i^{k+1} - x_j^{k+1}).$$

  - d Agent i lets $w_i^{k+1} = \delta_i \tilde{\mathbf{w}}_i^{k+1}(i)$, where

$$\tilde{\mathbf{w}}'^{k+1}_i = (1 - d_i^{\text{in}}\delta_i)\tilde{\mathbf{w}}'^{k}_i + \sum_{j\in N^{\text{in}}(i)} \delta_i \tilde{\mathbf{w}}'^{k}_j$$



Distributed ADMM over Directed Networks

- Each agent i maintains x_i^k, $\tilde{\mathbf{w}}_i^k$ (w_i^k) and p_{ij}^k, z_{ij}^k, for $j \in \mathcal{N}^{\text{in}}(i)$.
- At iteration k,
  - a Agent i updates the primal variable x_i^k as

$$x_i^{k+1} \in \operatorname*{argmin}_{x_i} f_i(x_i) - w_i^k \sum_{j\in\mathcal{N}^{\text{in}}(i)} (p_{ij}^k)' x_i + w_i^k \frac{\beta}{2} \sum_{j\in\mathcal{N}^{\text{in}}(i)} \left|\left| x_i - z_{ij}^k \right|\right|^2,$$

  The values x_i^{k+1} and $\tilde{\mathbf{w}}_i^k$ are sent to all out-neighbors.

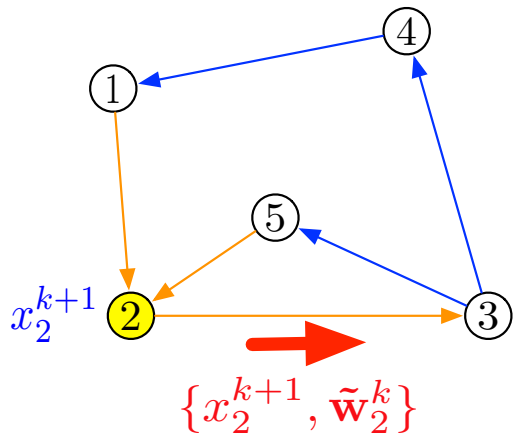
  - b For all $j \in \mathcal{N}^{\text{in}}(i)$, agent i computes

$$z_{ij}^{k+1} = \frac{1}{2}(x_i^{k+1} + x_j^{k+1}),$$

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

  - d Agent i lets $w_i^{k+1} = \delta_i \tilde{\mathbf{w}}_i^{k+1}(i)$, where

$$\tilde{\mathbf{w}}'^{k+1}_i = (1 - d_i^{\text{in}}\delta_i)\tilde{\mathbf{w}}'^{k}_i + \sum_{j\in N^{\text{in}}(i)} \delta_i \tilde{\mathbf{w}}'^{k}_j$$



Distributed ADMM over Directed Networks

- Each agent i maintains x_i^k, $\tilde{\mathbf{w}}_i^k$ (w_i^k) and p_{ij}^k, z_{ij}^k, for $j \in \mathcal{N}^{\text{in}}(i)$.
- At iteration k,
  - a Agent i updates the primal variable x_i^k as

$$x_i^{k+1} \in \operatorname*{argmin}_{x_i} f_i(x_i) - w_i^k \sum_{j\in\mathcal{N}^{\text{in}}(i)} (p_{ij}^k)' x_i + w_i^k \frac{\beta}{2} \sum_{j\in\mathcal{N}^{\text{in}}(i)} \left|\left| x_i - z_{ij}^k \right|\right|^2,$$

  The values x_i^{k+1} and $\tilde{\mathbf{w}}_i^k$ are sent to all out-neighbors.

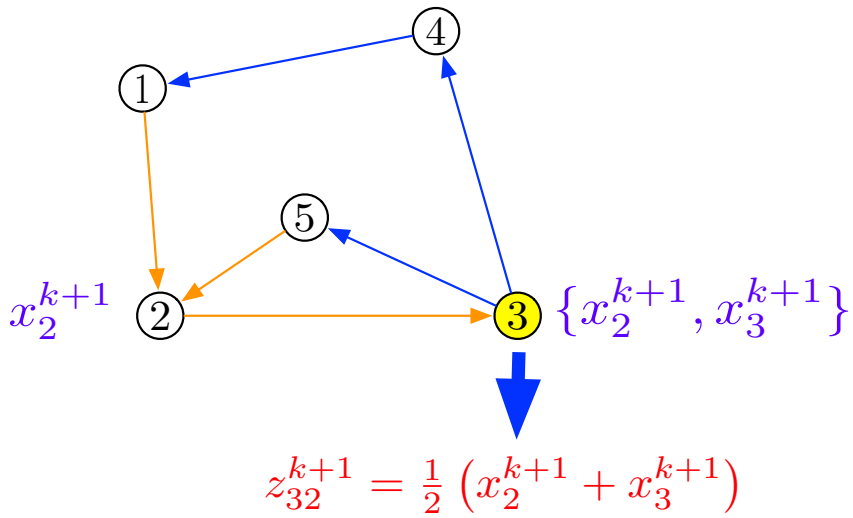
  - b For all $j \in \mathcal{N}^{\text{in}}(i)$, agent i computes

$$z_{ij}^{k+1} = \frac{1}{2}(x_i^{k+1} + x_j^{k+1}),$$

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

  - d Agent i lets $w_i^{k+1} = \delta_i \tilde{\mathbf{w}}_i^{k+1}(i)$, where

$$\tilde{\mathbf{w}}'^{k+1}_i = (1 - d_i^{\text{in}}\delta_i)\tilde{\mathbf{w}}'^{k}_i + \sum_{j\in N^{\text{in}}(i)} \delta_i \tilde{\mathbf{w}}'^{k}_j$$



Distributed ADMM over Directed Networks

- Each agent i maintains x_i^k, $\tilde{\mathbf{w}}_i^k$ (w_i^k) and p_{ij}^k, z_{ij}^k, for $j \in \mathcal{N}^{\text{in}}(i)$.
- At iteration k,
  - a Agent i updates the primal variable x_i^k as

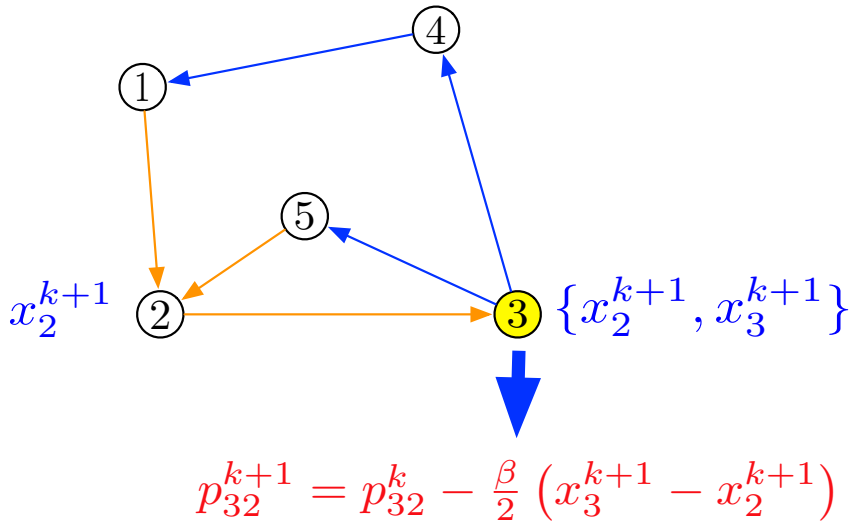
$$x_i^{k+1} \in \operatorname*{argmin}_{x_i} f_i(x_i) - w_i^k \sum_{j\in\mathcal{N}^{\text{in}}(i)} (p_{ij}^k)' x_i + w_i^k \frac{\beta}{2} \sum_{j\in\mathcal{N}^{\text{in}}(i)} \left|\left| x_i - z_{ij}^k \right|\right|^2,$$

  The values x_i^{k+1} and $\tilde{\mathbf{w}}_i^k$ are sent to all out-neighbors.

  - b For all $j \in \mathcal{N}^{\text{in}}(i)$, agent i computes

$$z_{ij}^{k+1} = \frac{1}{2}(x_i^{k+1} + x_j^{k+1}),$$

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



  - d Agent i lets $w_i^{k+1} = \delta_i \tilde{\mathbf{w}}_i^{k+1}(i)$, where

$$\tilde{\mathbf{w}}'^{k+1}_i = (1 - d_i^{\text{in}}\delta_i)\tilde{\mathbf{w}}'^{k}_i + \sum_{j\in N^{\text{in}}(i)} \delta_i \tilde{\mathbf{w}}'^{k}_j$$

Distributed ADMM over Directed Networks

- Each agent i maintains x_i^k, $\tilde{\mathbf{w}}_i^k$ (w_i^k) and p_{ij}^k, z_{ij}^k, for $j \in \mathcal{N}^{\text{in}}(i)$.
- At iteration k,
  - a Agent i updates the primal variable x_i^k as

$$x_i^{k+1} \in \operatorname*{argmin}_{x_i} f_i(x_i) - w_i^k \sum_{j\in\mathcal{N}^{\text{in}}(i)} (p_{ij}^k)' x_i + w_i^k \frac{\beta}{2} \sum_{j\in\mathcal{N}^{\text{in}}(i)} \left|\left| x_i - z_{ij}^k \right|\right|^2,$$

  The values x_i^{k+1} and $\tilde{\mathbf{w}}_i^k$ are sent to all out-neighbors.

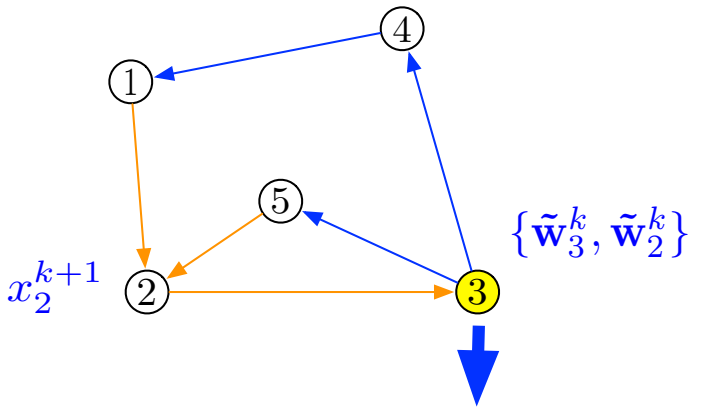
  - b For all $j \in \mathcal{N}^{\text{in}}(i)$, agent i computes

$$z_{ij}^{k+1} = \frac{1}{2}(x_i^{k+1} + x_j^{k+1}),$$

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

  - d Agent i let $w_i^{k+1} = \delta_i \tilde{\mathbf{w}}_i^{k+1}(i)$, where

$$\tilde{\mathbf{w}'}_i^{k+1} = (1 - d_i^{\text{in}}\delta_i)\tilde{\mathbf{w}'}_i^k + \sum_{j\in N^{\text{in}}(i)} \delta_i \tilde{\mathbf{w}'}_j^k$$



$$\tilde{\mathbf{w}'}_3^{k+1} = (1 - \delta_3 d_3^{\text{in}})\tilde{\mathbf{w}'}_3^k + \delta_3 \tilde{\mathbf{w}'}_2^k$$

$$w_3^{k+1} = \delta_3 \tilde{\mathbf{w}}_3^{k+1}(3)$$

Rate of Convergence

- Let $\{x^k, z^k, p^k, w^k\}$ be the sequence generated by the Directed ADMM algorithm. We define the ergodic average:

$$\bar{x}^k = \frac{1}{k}\sum_{l=0}^{k-1} x^k.$$

- Assume the sequence $\{x^k\}$ is bounded.

Theorem

Let (x^*, z^*, p^*) be a primal-dual optimal solution for problem (1). The following hold at each iteration k:

$$|F(\bar{x}^k) - F(x^*)| \leq \frac{1}{k}D, \quad \sum_{(i,j)\in E} |\bar{x}_i^k - \bar{x}_j^k| \leq \frac{1}{k}C,$$

where the constants C and D depends on number of nodes N, number of edges M, $||p^*||$, and the second largest eigenvalue of Laplacian.

Conclusions and Future Work

- We presented distributed ADMM-based algorithms for solving multi-agent optimization problems over undirected and directed networks.
- For general convex cost functions, we showed that these methods converge at the faster rate $O(1/k)$ and provided rate estimates that highlighted dependence on network structure.
- Simulation results illustrate the superior performance of ADMM (even for network topologies with slow mixing).
  - This comes at the expense of a more complex local optimization step.
- Extensions and Future Work:
  - Asynchronous distributed ADMM algorithms [Wei, Ozdaglar 13].
  - Broadcast-based ADMM algorithms [Makhdoumi, Ozdaglar 14].
  - Graph balancing for distributed optimization over directed networks.
  - ADMM-type algorithms for time-varying networks.
  - Distributed and incremental second order methods [Gurbuzbalaban, Ozdaglar, Parrilo 14].