Gossip algorithms: Tutorial & Recent advances

Part II: Recent problems, Optimization, and Learning

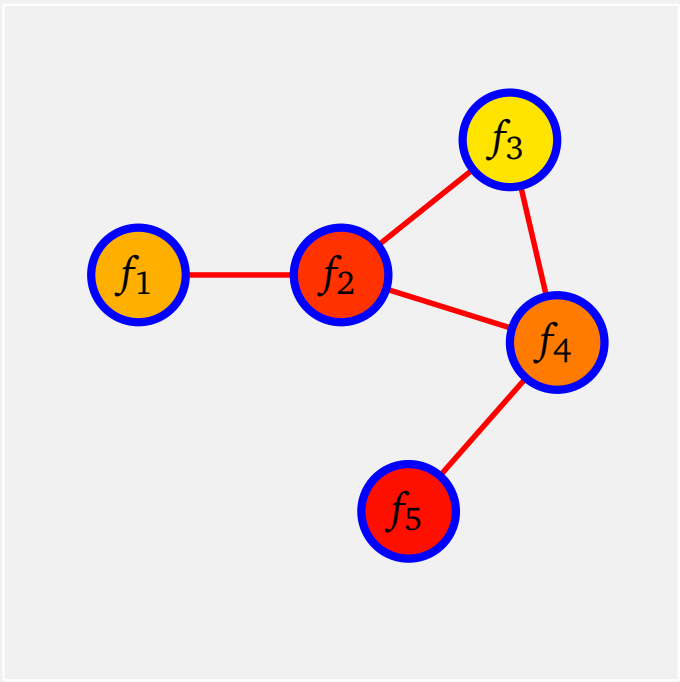
Franck Iutzeler LJK, Université Grenoble Alpes

Smile Paris – November 3, 2016

- Recent Advances and Problems in Gossiping
- Distributed Optimization and Learning **[FOCUS]**
  - Gradient algorithms and variations
  - Advanced algorithms

Problem:

$$\min_{x \in \mathbb{R}^d} \; f(x) \triangleq \sum_{i \in V} f_i(x)$$



- f_i is a **convex** function **local** to agent i
- f is **nowhere available**
- Agents should reach consensus over a minimizer $x^\star$ of f

- Recent Advances and Problems in Gossiping
- Distributed Gradient Algorithms
  - Synchronous Case
  - Asynchronous Case
  - Application Examples
- Advanced Algorithms
  - Synchronous Case
  - Asynchronous Case
  - Application Examples

- Recent Advances and Problems in Gossiping

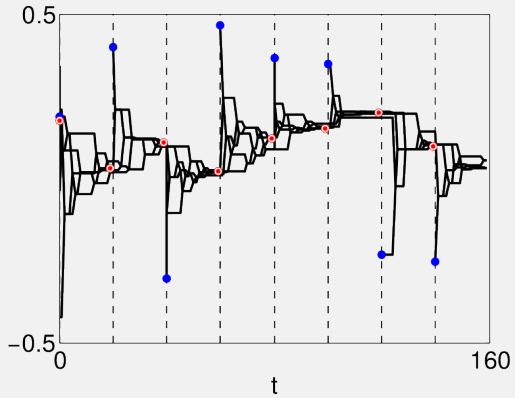
Distributed Gradient Algorithms

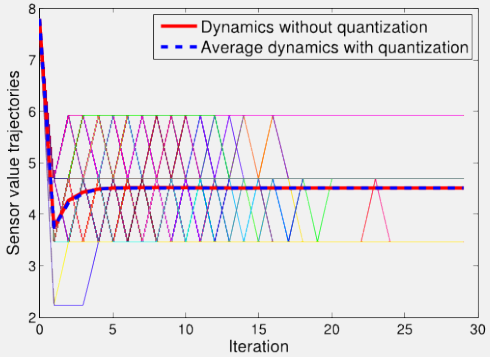
- Synchronous Case
- Asynchronous Case
- Application Examples

Advanced Algorithms

- Synchronous Case
- Asynchronous Case
- Application Examples

- **Open Systems**
  agents come and go; new paradigms to develop
  Hendrickx & Martin *Open Multi-Agent Systems: Gossiping with Deterministic Arrivals and Departures*, Allerton 2016.





- **Quantized Gossip**
  values or communications are quantized
  Frasca et al. *Average consensus by gossip algorithms with quantized communication*, CDC 2008
  Benezit et al. *Interval consensus: from quantized gossip to voting*, ICASSP 2009

- **Gossip with errors**
  error on communications, Byzantine attackers

- **Gossip as an intermediate**
  Szerenyi et al. *Gossip-based distributed stochastic bandit algorithms*, ICML 2013.

Recent Advances and Problems in Gossiping

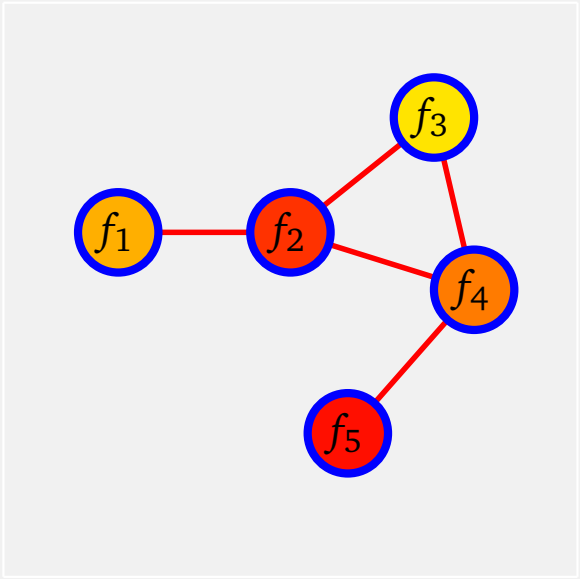
- **Distributed Gradient Algorithms**
  - Synchronous Case
  - Asynchronous Case
  - Application Examples

Advanced Algorithms

Synchronous Case
Asynchronous Case
Application Examples

Problem:

$$\min_{x\in\mathbb{R}^d} \; f(x) \triangleq \sum_i f_i(x)$$



- f_i is an L-**smooth convex** function **local** to agent i
- f is **nowhere available**
- Agents should reach consensus over a minimizer of f

Reformulation: Product-Space trick

- $F(x) = \sum_i f_i(x_i)$
- Dimension went from d to Nd
- $\nabla F(x) = [\nabla f_1(x_1), .., \nabla f_N(x_N)]$

Dimension went from d to Nd but $\nabla F(x) = [\nabla f_1(x_1), .., \nabla f_N(x_N)]$

Gradient Computation is parallel

$$\min \sum_i f_i(x) \Leftrightarrow \min F(x) \; + \; \text{consensus } x = (x, x, .., x)$$

- Average Consensus, thus Gossip, is needed to obtain the sought solution

Recent Advances and Problems in Gossiping

- **Distributed Gradient Algorithms**
  - Synchronous Case
    - Asynchronous Case
    - Application Examples

Advanced Algorithms

- Synchronous Case
- Asynchronous Case
- Application Examples

Natural Algorithm: $x^{k+1} = W(x^k - \gamma^k \nabla F(x^k))$

At each iteration k:

- Each agent i performs a *local* gradient descent

$$y_i^{k+1} = x_i^k - \gamma^k \nabla f_i(x_i^k)$$

- The agents perform an average gossip step

$$x^{k+1} = W y^{k+1}$$

Otherwise said: $x^k = (W)^k x^0 - \sum_{\ell=1}^{k-1} \gamma^\ell (W)^{k-\ell} \nabla F(x^\ell)$

- What can be say about the convergence?
  Fixed stepsize or decreasing one as in stochastic algorithms?

$$x^{k+1} = W(x^k - \gamma^k \nabla F(x^k)) = (W)^k x^0 - \sum_{\ell=1}^{k-1} \gamma^\ell (W)^{k-\ell} \nabla F(x^\ell)$$

Assumptions:

- W is doubly-stochastic and primitive
  Same for average gossip
- $\sum_\ell \gamma^\ell = +\infty\,; \quad \sum_\ell (\gamma^\ell)^2 < +\infty\,; \quad \gamma^{k+1}/\gamma^k \to 1$
  As in stochastic algorithm even if there is no noise

1. $\|(I-J)x^{k+1}\|_2 \leq \underbrace{(\sigma)^{k+1}\|x^0\|_2}_{\text{exponential decay of gossip}} + \gamma^k \underbrace{C \sum_{\ell=1}^{k} \frac{\gamma^\ell}{\gamma^k} (\sigma)^{k+1-\ell}}_{\text{finite}} = \mathcal{O}(\gamma^k)$

where $\sigma = \|(I-J)W\|_2 < 1$ is a bound on the gossiping rate.

2. $f(\bar{x}^{k+1}) \leq f(\bar{x}^k) - \gamma^k \|\nabla f(\bar{x}^k)\|_2^2 + C(\gamma^k)^2 \Rightarrow \sum_{\ell=1}^{\infty} \gamma^k \|\nabla f(\bar{x}^k)\|_2^2 < +\infty$
where $\bar{x}^k = \frac{1}{N} \sum_{i=}^{N} x_i^k$.

- (x^k) converges to a *consensus* over a minimizer of f

$$x^{k+1} = W(x^k - \gamma^k \nabla F(x^k)) = (W)^k x^0 - \sum_{\ell=1}^{k-1} \gamma^\ell (W)^{k-\ell} \nabla F(x^\ell)$$

Assumptions:

- W is doubly-stochastic and primitive
  Same for average gossip
- $\sum_\ell \gamma^\ell = +\infty\,; \quad \sum_\ell (\gamma^\ell)^2 < +\infty\,; \quad \gamma^{k+1}/\gamma^k \to 1$
  As in stochastic algorithm even if there is no noise

Remark:

- if $W = J$ i.e. full consensus at each iteration, a *constant stepsize* is possible!

Even though Gossip is $\mathcal{O}(\sigma^k)$ and gradient is $\mathcal{O}(1/k)$, parallel gradient and distributed gradient have very different behaviors.

Reference

Tsitsiklis, Bertsekas & Athans *Distributed Asynchronous Deterministic and Stochastic gradient optimization algorithm*, IEEE TAC, 1986

Recent Advances and Problems in Gossiping

- **Distributed Gradient Algorithms**
  - Synchronous Case
  - Asynchronous Case
  - Application Examples

Advanced Algorithms
- Synchronous Case
- Asynchronous Case
- Application Examples

Natural Algorithm: $x^{k+1} = W_{\xi^{k+1}}(x^k - \gamma^k \nabla F(x^k))$

At each iteration k:

- Each agent i performs a *local* gradient descent

$$y_i^{k+1} = x_i^k - \gamma^k \nabla f_i(x_i^k)$$

- The agents perform a random average gossip step

$$x^{k+1} = W_{\xi^{k+1}} y^{k+1}$$

Assumptions:

- (W_{ξ^k}) is an i.i.d. sequence of doubly-stochastic and $\mathbb{E}[W]$ is primitive
  Same for average gossip
- $\sum_\ell \gamma^\ell = +\infty$; $\sum_\ell (\gamma^\ell)^2 < +\infty$; $\gamma^{k+1}/\gamma^k \to 1$
  As in stochastic algorithm even if there is no noise

- (x^k) converges almost surely to a *consensus* over a minimizer of f

Reference

Nedic & Ozdaglar *Distributed Subgradient Methods for Multi Agent Optimization*, IEEE TAC, 2009

Distributed gradient algorithm $x^{k+1} = W_{\xi^{k+1}}(x^k - \gamma^k \nabla F(x^k))$ converges

- Hypotheses on mixing matrices = for average gossip
- Hypotheses on stepsizes = stochastic approximation

Extensions

- Subgradients; non-convex functions
- Stochastic gradient
- Column-Stochastic mixing matrices
  it is *as if* the average gossiping was lauched at each iteration with the current gradient

References

Bianchi & Jakubowicz *Convergence of a Multi Agent Projected Stochastic gradient for non convex optimization*, IEEE TAC, 2013

Bianchi, Fort & Hachem *Performance of a Distributed Stochastic Approximation algorithm*, IEEE TIT, 2013

Duchi, Agarwal & Wainwright *Dual Averaging for Distributed Optimization*, IEEE TAC, 2012

Recent Advances and Problems in Gossiping

- **Distributed Gradient Algorithms**
  - Synchronous Case
  - Asynchronous Case
  - Application Examples

Advanced Algorithms

- Synchronous Case
- Asynchronous Case
- Application Examples

- At each time k, each sensor i receive $h_i(k)$ and $y_i(k) = h_i(k)^{\mathrm{T}} w^\star + n_i(k)$
- The goal is to find $w^\star$ by minimizing

$$J(w) = \sum_{k=1}^{K} \sum_{i=1}^{N} \left(y_i(k) - h_i(k)^{\mathrm{T}} w \right)^2$$

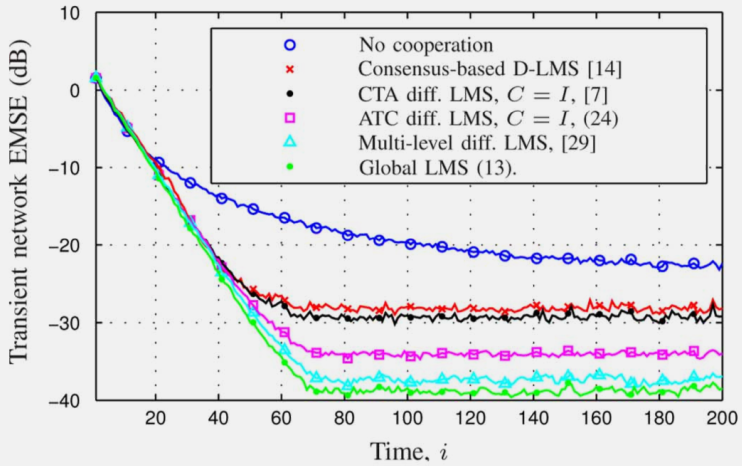


image: Cattivelli & Sayed

References: Diffusion LMS, Tracking for $w^\star(k)$

Schizas, Mateos, & Giannakis *Distributed LMS for Consensus-Based In-Network Adaptive Processing*, IEEE TSP, 2010.

Cattivelli & Sayed *Diffusion LMS Strategies for Distributed Estimation*, IEEE TSP, 2010.

Chen, Richard & Sayed *Diffusion LMS over Multitask Networks*, IEEE TSP, 2015

- The goal is to minimize $\sum_{i=1}^{N} f_i(x)$ over a closed set $\mathcal{X}$
- the f_i are convex but not necessarily smooth
  $\Rightarrow$ use of Nesterov dual averaging method.

$$z^{k+1} = Wz^k + g^k \quad \text{with } g^k = \left(g_i^k \in \partial f_i(x_i^k)\right)$$

$$\text{For each agent } i \quad x_i^{k+1} = \arg\min_{x \in \mathcal{X}} \left\{ \langle z_i^{k+1}; x \rangle + \frac{1}{\alpha^k} \psi(x) \right\}$$

Doubly stochastic W
a^k decreasing (typically $\propto 1/\sqrt{k}$)

$$f(\bar{x_i}^T) - f^\star \leq OPT + NET \leq \mathcal{O}\left(\frac{\log(N)}{\sqrt{T\lambda_{N-1}(L)}}\right)$$

References: Optimization, Learning

Nesterov *Primal-dual subgradient methods for convex problems*, Math. Prog., 2009.
Xiao *Dual averaging methods for regularized stochastic learning and online optimization*, JMLR, 2010.
Duchi, Agarwal, & Wainwright *Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling*, NIPS 2010 & IEEE TAC 2012.
Ma, Smith, Jaggi, Jordan, Richtarik, & Takac *Adding vs. Averaging in Distributed Primal-Dual Optimization*, ICML 2015.

- The goal is to minimize $\sum_{i=1}^{N} f_i(x)$ over a closed set $\mathcal{X}$
- the f_i are convex but not necessarily smooth
  $\Rightarrow$ use of Nesterov dual averaging method.

For One-Way/Broadcast communications, there exists a Sum-Weight version!

$$
\begin{aligned}
s^{k+1} &= Ws^k + g^k \quad \text{with } g^k = \left(g_i^k \in \partial f_i(x_i^k)\right) \\
w^{k+1} &= Ww^k \\
\text{For each agent } i \quad x_i^{k+1} &= \arg\min_{x \in \mathcal{X}} \left\{ \left\langle \frac{s_i^{k+1}}{w_i^{k+1}}; x \right\rangle + \frac{1}{\alpha^k}\psi(x) \right\}
\end{aligned}
$$

Column stochastic W
a^k decreasing (typically $\propto 1/\sqrt{k}$) $\quad f(\bar{x_i}^T) - f^\star \leq OPT + NET \leq \mathcal{O}\left(\frac{\log(N)}{\sqrt{T\lambda_{N-1}(L)}}\right)$
Same rate but larger set of communications (thus L) possible

Reference:

Tsianos, Lawlor, & Rabbat *Push-sum distributed dual averaging for convex optimization*, CDC 2012.

Recent Advances and Problems in Gossiping

Distributed Gradient Algorithms

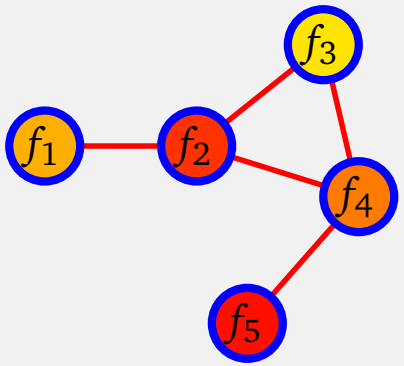
- Synchronous Case
- Asynchronous Case
- Application Examples

Advanced Algorithms

- Synchronous Case
- Asynchronous Case
- Application Examples

Problem:

$$\min_{x \in \mathbb{R}^d} \; f(x) \triangleq \sum_i f_i(x)$$



- f_i is a **convex** function **local** to agent i
- f is **nowhere available**
- Agents should reach consensus over a minimizer of f

Goal: Mitigate the Optimization + Consensus problem differently

- Overcome the decreasing stepsize drawback of Distributed Gradient
- Include general convex functions (non-smooth, composite)

A proper problem for distributed optimization

The original problem is not suited as it does take into account

- the fact that each agent only has access to its own cost function;
- the fact that they have to exchange to reach the wanted optimum.

$$\min_{x \in \mathbb{R}} \sum_{i \in V} f_i(x)$$

- Starting from the original problem

A proper problem for distributed optimization

The original problem is not suited as it does take into account

- the fact that each agent only has access to its own cost function;
- the fact that they have to exchange to reach the wanted optimum.

$$\min_{x \in \mathbb{R}} \sum_{i \in V} f_i(x)$$

$$\min_{x \in \mathbb{R}^{\mathbf{N}}} F(x) \triangleq \sum_{i \in V} f_i(x_{\mathbf{i}})$$

$$\text{subject to} \quad x_1 = x_2 = \ldots = x_N$$

- Starting from the original problem
- Adding the fact that the agents only know their own functions

A proper problem for distributed optimization

The original problem is not suited as it does take into account

- the fact that each agent only has access to its own cost function;
- the fact that they have to exchange to reach the wanted optimum.

$$\min_{x\in\mathbb{R}} \sum_{i\in V} f_i(x)$$

$$\min_{x\in\mathbb{R}^{\mathbf{N}}} F(x) \triangleq \sum_{i\in V} f_i(x_{\mathbf{i}})$$

subject to $x_1 = x_2 = \ldots = x_N$

$$\min_{x\in\mathbb{R}^N} F(x) + \iota_{\mathrm{Span}(\mathbf{1})}(x)$$

- Starting from the original problem
- Adding the fact that the agents only know their own functions
- We put the constraint into the function to minimize

with the *indicator function*

$$\iota_C(x) = \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{elsewhere} \end{cases}$$

A proper problem for distributed optimization

The original problem is not suited as it does take into account

- the fact that each agent only has access to its own cost function;
- the fact that they have to exchange to reach the wanted optimum.

$$\min_{x\in\mathbb{R}} \sum_{i\in V} f_i(x)$$

$$\min_{x\in\mathbb{R}^{\mathbf{N}}} F(x) \triangleq \sum_{i\in V} f_i(x_{\mathbf{i}})$$

subject to $x_1 = x_2 = \ldots = x_N$

$$\min_{x\in\mathbb{R}^N} F(x) + \iota_{\mathrm{Span}(\mathbf{1})}(x)$$

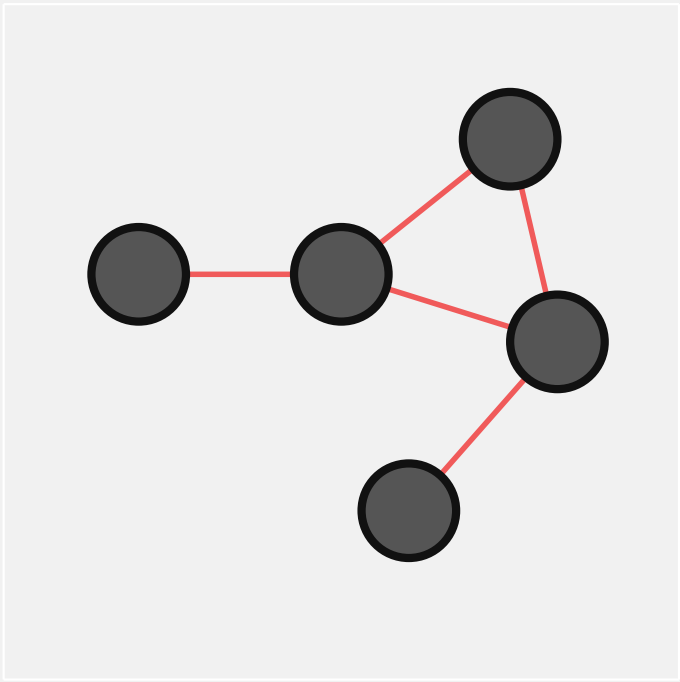
- Starting from the original problem
- Adding the fact that the agents only know their own functions
- We put the constraint into the function to minimize

with the *indicator function*

$$\iota_C(x) = \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{elsewhere} \end{cases}$$

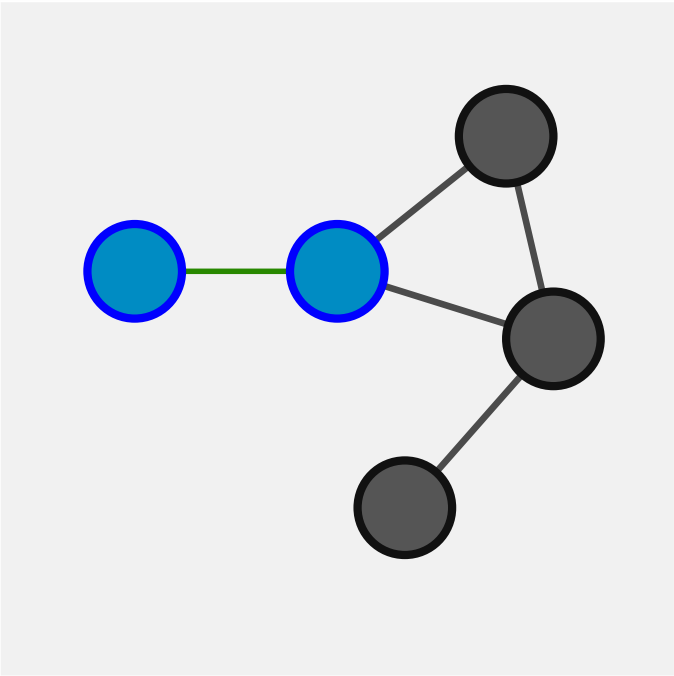
General idea: Ensure consensus over *L overlapping connected subsets*

e.g. over all edges. Subset = 2 connected nodes + connecting link



General idea: Ensure consensus over *L overlapping connected subsets*

e.g. over all edges. Subset = 2 connected nodes + connecting link

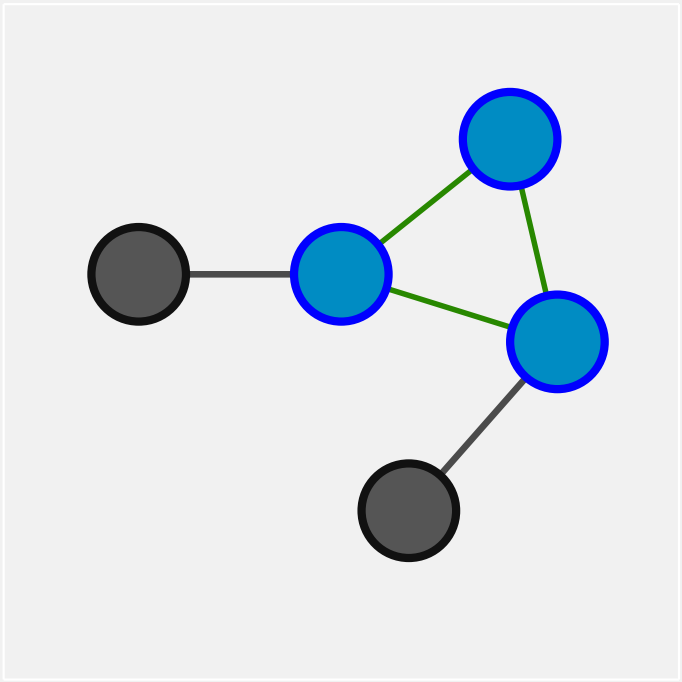


- $A_1 = \{1, 2\}$ $\quad M_1 x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$$\iota_{\mathrm{Span}(\mathbf{1})} \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right)$$

General idea: Ensure consensus over *L overlapping connected subsets*

e.g. over all edges. Subset = 2 connected nodes + connecting link

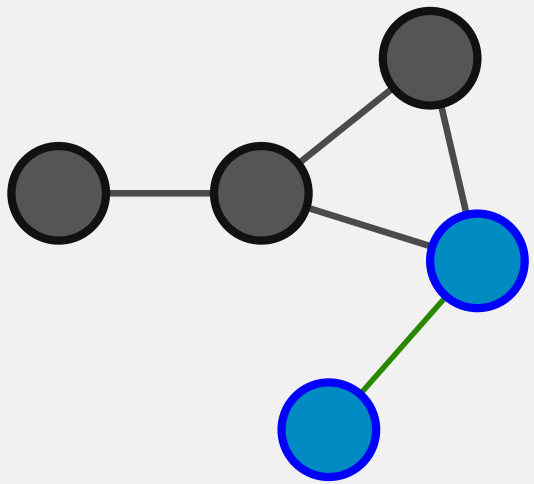


- $A_1 = \{1, 2\}$ $\quad M_1 x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $A_2 = \{2, 3, 4\}$ $\quad M_2 x = \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}$

$$\iota_{\mathrm{Span}(\mathbf{1})}\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) + \iota_{\mathrm{Span}(\mathbf{1})}\left(\begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}\right)$$

General idea: Ensure consensus over *L overlapping connected subsets*

e.g. over all edges. Subset = 2 connected nodes + connecting link

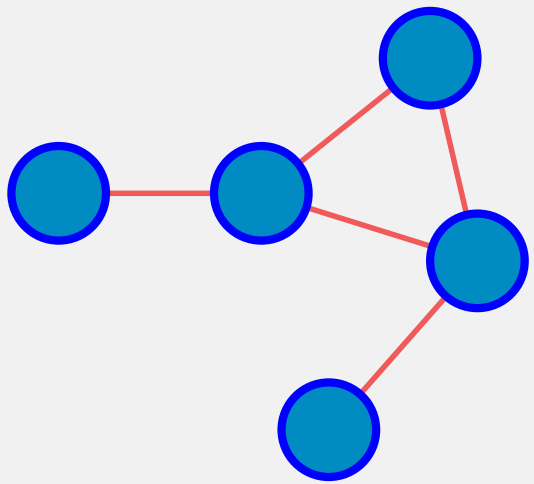


- $A_1 = \{1, 2\}$ $\quad M_1 x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $A_2 = \{2, 3, 4\}$ $\quad M_2 x = \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}$
- $A_3 = \{4, 5\}$ $\quad M_3 x = \begin{bmatrix} x_4 \\ x_5 \end{bmatrix}$

$$\iota_{\mathrm{Span}(\mathbf{1})}\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) + \iota_{\mathrm{Span}(\mathbf{1})}\left(\begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}\right) + \iota_{\mathrm{Span}(\mathbf{1})}\left(\begin{bmatrix} x_4 \\ x_5 \end{bmatrix}\right)$$

General idea: Ensure consensus over *L overlapping connected subsets*

e.g. over all edges. Subset = 2 connected nodes + connecting link

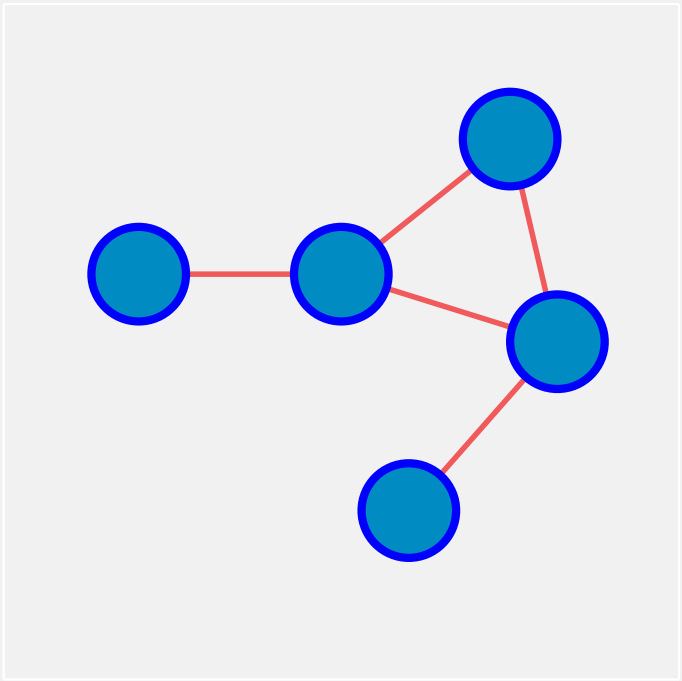


- $A_1 = \{1, 2\}$ $\quad M_1 x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $A_2 = \{2, 3, 4\}$ $\quad M_2 x = \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}$
- $A_3 = \{4, 5\}$ $\quad M_3 x = \begin{bmatrix} x_4 \\ x_5 \end{bmatrix}$

$$\iota_{\mathrm{Span}(\mathbf{1})}\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) + \iota_{\mathrm{Span}(\mathbf{1})}\left(\begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}\right) + \iota_{\mathrm{Span}(\mathbf{1})}\left(\begin{bmatrix} x_4 \\ x_5 \end{bmatrix}\right) = \iota_{\mathrm{Span}(\mathbf{1})}(x)$$

General idea: Ensure consensus over *L overlapping connected subsets*

e.g. over all edges. Subset = 2 connected nodes + connecting link



- $A_1 = \{1, 2\}$ $\quad M_1 x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $A_2 = \{2, 3, 4\}$ $\quad M_2 x = \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}$
- $A_3 = \{4, 5\}$ $\quad M_3 x = \begin{bmatrix} x_4 \\ x_5 \end{bmatrix}$
- $M \triangleq \begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix}$: size $\sum_{\ell=1}^{L} |A_\ell| \triangleq M \times N$

$$\iota_{\mathrm{Span}(\mathbf{1})}(M_1 x) + \iota_{\mathrm{Span}(\mathbf{1})}(M_2 x) + \iota_{\mathrm{Span}(\mathbf{1})}(M_3 x) \triangleq G(Mx)$$

Equivalent problem

$$\min_{x \in \mathbb{R}^N} F(x) + G(Mx)$$

Reformulated Problem

$$\min_{x\in\mathbb{R}^N} \underbrace{\sum_{\text{nodes } i=1}^{N} f_i(x_i)}_{F(x)} + \underbrace{\sum_{\text{areas } \ell=1}^{L} \iota_{\mathrm{Span}(\mathbf{1})}\left(M_\ell x\right)}_{G(Mx)}$$

- A separable networked separated problem...
  The functions f_i act on local variables
  The consensus is ensured locally on overlapping areas

Does this leads to a distributed algorithm?

Not exactly: $G(M\cdot) \equiv \iota_{\text{whole consensus}}(\cdot)$ so any direct algorithm will look like

$$x_{k+1} = \begin{bmatrix} \vdots \\ \bar{x}^k \\ \vdots \end{bmatrix} - \gamma \begin{bmatrix} \vdots \\ \nabla f_i(\bar{x}^k) \\ \vdots \end{bmatrix} \quad \text{with } \bar{x}^k = \tfrac{1}{N}\textstyle\sum_{i=1}^{L} x_i^k$$

This is a simple proximal gradient with F as the smooth function and G as the proximable one.

Reformulated Problem

$$\min_{x\in\mathbb{R}^N} \underbrace{\sum_{\text{nodes } i=1}^{N} f_i(x_i)}_{F(x)} + \underbrace{\sum_{\text{areas } \ell=1}^{L} \iota_{\mathrm{Span}(\mathbf{1})}\left(M_\ell x\right)}_{G(Mx)}$$

- A separable networked separated problem...
  The functions f_i act on local variables
  The consensus is ensured locally on overlapping areas

Does this leads to a distributed algorithm?

Not exactly: $G(M\cdot) \equiv \iota_{\text{whole consensus}}(\cdot)$ so any direct algorithm will look like

otherwise said $x^{k+1} = J(x^k - \gamma\nabla F(x^k))$; as already seen before.

This is a simple proximal gradient with F as the smooth function and G as the proximable one.

Reformulated Problem

$$\min_{x\in\mathbb{R}^N, y\in\mathbb{R}^M} \underbrace{\sum_{\text{nodes } i=1}^{N} f_i(x_i)}_{F(x)} + \underbrace{\sum_{\text{areas } \ell=1}^{L} \iota_{\text{Span}(\mathbf{1})}\left(y_{|\ell}\right)}_{G(y)}$$

$$\text{s.t. } Mx = y$$

- A separable networked separated problem...
  The functions f_i act on local variables
  The consensus is ensured locally on overlapping areas
- ... that we have to split.
  Splitting algorithms are popular in Optimization (proximal gradient, ADMM, primal-dual)
  We divide variable y into L adapted blocks $(y_{|\ell})$

- Let us investigate the resulting algorithms

Recent Advances and Problems in Gossiping

Distributed Gradient Algorithms

- Synchronous Case
- Asynchronous Case
- Application Examples

Advanced Algorithms

- Synchronous Case
- Asynchronous Case
- Application Examples

Distributed Optimization algorithm: Reformulated Problem + Splitting method

$$\min_{x\in\mathbb{R}^N, y\in\mathbb{R}^M} \underbrace{\sum_{\text{nodes } i=1}^{N} f_i(x_i)}_{F(x)} + \underbrace{\sum_{\text{areas } \ell=1}^{L} \iota_{\text{Span}(\mathbf{1})}\left(y_{|\ell}\right)}_{G(y)}$$

$$\text{s.t. } Mx = y$$

Splitting method what to treat with a *prox* v.s. a *gradient*

- *G has* to be treated with a *prox*
  prox on indicator = projection
- *F it depends!*

gradient: f smooth
$x^{k+1} = x^k - \gamma \nabla f(x^k)$

prox: f convex
$x^{k+1} = \arg\min_w \{f(w) + \frac{1}{2}\|w - x^k\|^2\} = x^k - \gamma \partial f(x^{k+1})$

$$\min_{x\in\mathbb{R}^N, y\in\mathbb{R}^M} F(x) + G(y)$$
$$\text{s.t. } Mx = y$$

ADMM

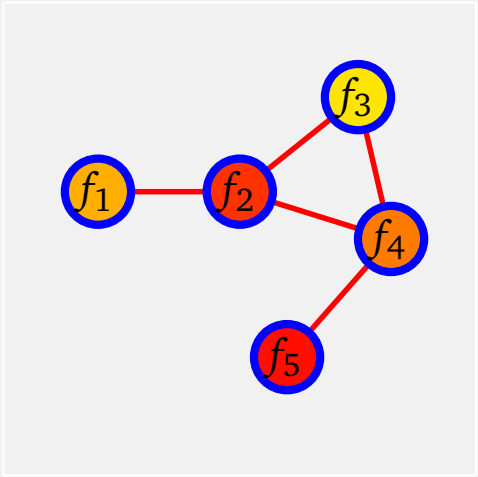
$$x^{k+1} = \arg\min_w \{F(w) + \frac{\rho}{2}\|Mw - z^k + \frac{\lambda^k}{\rho}\|^2\}$$
$$z^{k+1} = \arg\min_u \{G(u) + \frac{\rho}{2}\|Mx^{k+1} - u + \frac{\lambda^k}{\rho}\|^2\}$$
$$\lambda^{k+1} = \lambda^k + \rho(Mx^{k+1} - z^{k+1})$$

- any F convex; $\rho > 0$ free parameter
- most popular and studied

Primal-Dual

$$x^{k+1} = \arg\min_w \{F^c(w) + \langle \nabla F^s(x^k); w\rangle + \frac{1}{2\tau}\|Mw - v^k + \tau\lambda^k\|^2\}$$
$$z^{k+1} = \arg\min_u \{G(u) + \frac{\rho}{2}\|Mx^{k+1} - u + \frac{\lambda^k}{\rho}\|^2\}$$
$$\lambda^{k+1} = \lambda^k + \rho(Mx^{k+1} - z^{k+1})$$
$$u^{k+1} = (1 - \tau\rho)Mx^{k+1} + \tau\rho z^{k+1}$$

- $F = F^s + F^c$ with F^s smooth, both convex; $\rho, \tau > 0$ parameter (bounded choice)
- Includes the ADMM as a special case



Distributed ADMM

At each clock tick k:

- Every sensor i performs a minimization:

$$x_i^{k+1} = \arg\min_x \left\{ f_i(x) + \frac{\rho}{2} \sum_{\ell \in \sigma_i} \left(x_i - \bar{z}_{|\ell}^k + \frac{\lambda_{i,|\ell}^k}{\rho} \right)^2 \right\}$$

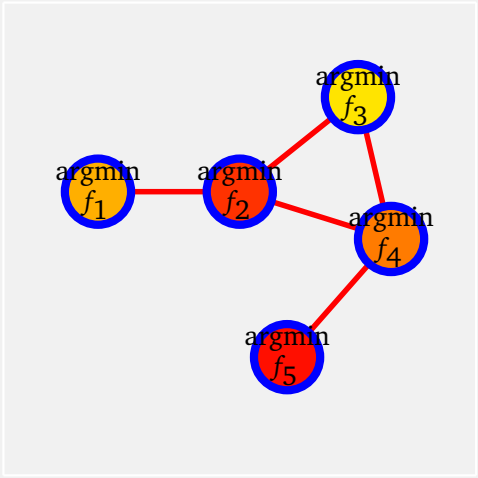
- Every subset A_ℓ computes its average:

$$\bar{z}_{|\ell}^{k+1} = \frac{1}{|A_\ell|} \sum_{i \in A_\ell} x_i^{k+1}$$

- Every sensor i updates:

$$\forall \ell \in \sigma_i, \quad \lambda_{i,|\ell}^{k+1} = \lambda_{i,|\ell}^k + \rho(x_i^{k+1} - \bar{z}_\ell^{k+1})$$

- Each step can be split by agent/block
  - $z_{|\ell}$: $|A_\ell|$-sized block, corresponds to subset ℓ
  - $\lambda_{i,|\ell}$: scalar, corresponds to agent i's entry in subset $\ell \in \sigma_i \triangleq \{l : i \in A_l\}$
- Convergence is immediate from Optimization theory
- Similar algorithm from Primal-Dual when F has a smooth part



Distributed ADMM

At each clock tick k:

- Every sensor i performs a minimization:

$$x_i^{k+1} = \arg\min_x \left\{ f_i(x) + \frac{\rho}{2} \sum_{\ell \in \sigma_i} \left(x_i - \bar{z}_{|\ell}^k + \frac{\lambda_{i,|\ell}^k}{\rho} \right)^2 \right\}$$

- Every subset A_ℓ computes its average:

$$\bar{z}_{|\ell}^{k+1} = \frac{1}{|A_\ell|} \sum_{i \in A_\ell} x_i^{k+1}$$

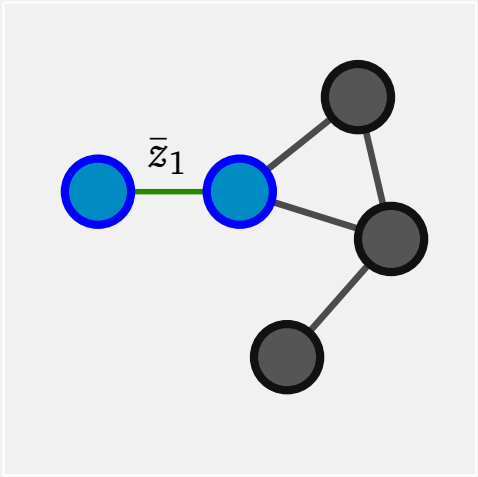
- Every sensor i updates:

$$\forall \ell \in \sigma_i, \quad \lambda_{i,|\ell}^{k+1} = \lambda_{i,|\ell}^k + \rho(x_i^{k+1} - \bar{z}_\ell^{k+1})$$

- Each step can be split by agent/block
  - $z_{|\ell}$: $|A_\ell|$-sized block, corresponds to subset ℓ
  - $\lambda_{i,|\ell}$: scalar, corresponds to agent i's entry in subset $\ell \in \sigma_i \triangleq \{l : i \in A_l\}$
- Convergence is immediate from Optimization theory
- Similar algorithm from Primal-Dual when F has a smooth part

Distributed ADMM

At each clock tick k:



- Every sensor i performs a minimization:

$$x_i^{k+1} = \arg\min_x \left\{ f_i(x) + \frac{\rho}{2} \sum_{\ell \in \sigma_i} \left(x_i - \bar{z}_{|\ell}^k + \frac{\lambda_{i,|\ell}^k}{\rho} \right)^2 \right\}$$

- Every subset A_ℓ computes its average:

$$\bar{z}_{|\ell}^{k+1} = \frac{1}{|A_\ell|} \sum_{i \in A_\ell} x_i^{k+1}$$

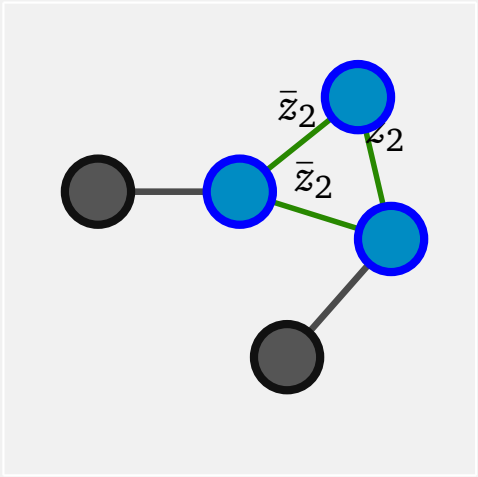
- Every sensor i updates:

$$\forall \ell \in \sigma_i, \quad \lambda_{i,|\ell}^{k+1} = \lambda_{i,|\ell}^k + \rho(x_i^{k+1} - \bar{z}_\ell^{k+1})$$

- Each step can be split by agent/block
  - $z_{|\ell}$: $|A_\ell|$-sized block, corresponds to subset ℓ
  - $\lambda_{i,|\ell}$: scalar, corresponds to agent i's entry in subset $\ell \in \sigma_i \triangleq \{l : i \in A_l\}$
- Convergence is immediate from Optimization theory
- Similar algorithm from Primal-Dual when F has a smooth part

Distributed ADMM

At each clock tick k:



- Every sensor i performs a minimization:

$$x_i^{k+1} = \arg\min_x \left\{ f_i(x) + \frac{\rho}{2} \sum_{\ell \in \sigma_i} \left(x_i - \bar{z}_{|\ell}^k + \frac{\lambda_{i,|\ell}^k}{\rho} \right)^2 \right\}$$

- Every subset A_ℓ computes its average:

$$\bar{z}_{|\ell}^{k+1} = \frac{1}{|A_\ell|} \sum_{i \in A_\ell} x_i^{k+1}$$

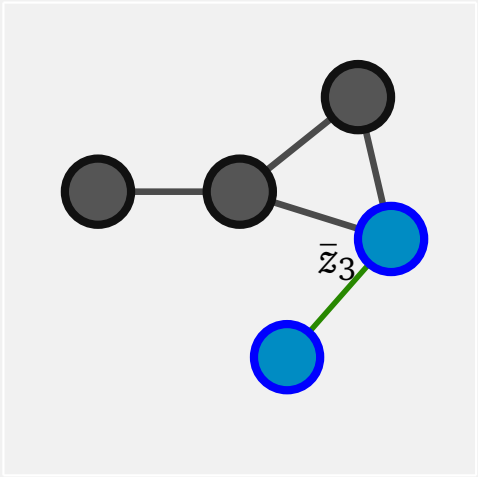
- Every sensor i updates:

$$\forall \ell \in \sigma_i, \quad \lambda_{i,|\ell}^{k+1} = \lambda_{i,|\ell}^k + \rho(x_i^{k+1} - \bar{z}_\ell^{k+1})$$

- Each step can be split by agent/block
  - $z_{|\ell}$: $|A_\ell|$-sized block, corresponds to subset ℓ
  - $\lambda_{i,|\ell}$: scalar, corresponds to agent i's entry in subset $\ell \in \sigma_i \triangleq \{l : i \in A_l\}$
- Convergence is immediate from Optimization theory
- Similar algorithm from Primal-Dual when F has a smooth part

Distributed ADMM

At each clock tick k:



- Every sensor i performs a minimization:

$$x_i^{k+1} = \arg\min_x \left\{ f_i(x) + \frac{\rho}{2} \sum_{\ell \in \sigma_i} \left(x_i - \bar{z}_{|\ell}^k + \frac{\lambda_{i,|\ell}^k}{\rho} \right)^2 \right\}$$

- Every subset A_ℓ computes its average:

$$\bar{z}_{|\ell}^{k+1} = \frac{1}{|A_\ell|} \sum_{i \in A_\ell} x_i^{k+1}$$

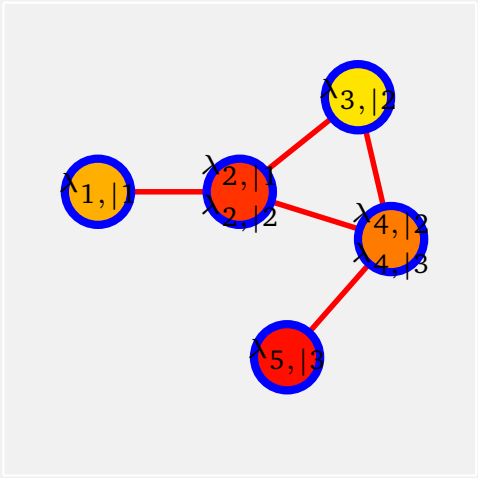
- Every sensor i updates:

$$\forall \ell \in \sigma_i, \quad \lambda_{i,|\ell}^{k+1} = \lambda_{i,|\ell}^k + \rho(x_i^{k+1} - \bar{z}_\ell^{k+1})$$

- Each step can be split by agent/block
  - $z_{|\ell}$: $|A_\ell|$-sized block, corresponds to subset ℓ
  - $\lambda_{i,|\ell}$: scalar, corresponds to agent i's entry in subset $\ell \in \sigma_i \triangleq \{l : i \in A_l\}$
- Convergence is immediate from Optimization theory
- Similar algorithm from Primal-Dual when F has a smooth part

Distributed ADMM

At each clock tick k:



- Every sensor i performs a minimization:

$$x_i^{k+1} = \arg\min_x \left\{ f_i(x) + \frac{\rho}{2} \sum_{\ell \in \sigma_i} \left(x_i - \bar{z}_{|\ell}^k + \frac{\lambda_{i,|\ell}^k}{\rho} \right)^2 \right\}$$

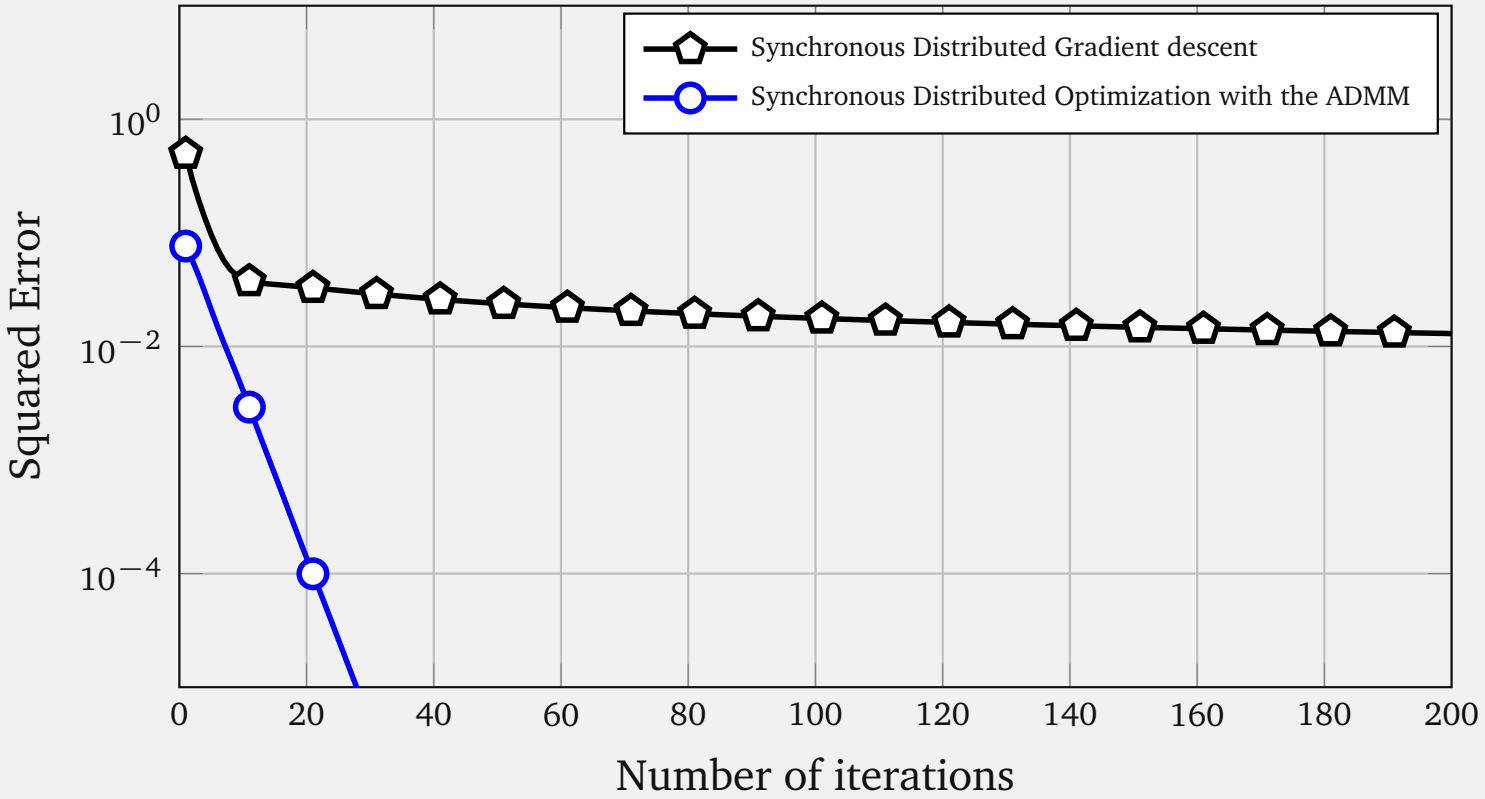
- Every subset A_ℓ computes its average:

$$\bar{z}_{|\ell}^{k+1} = \frac{1}{|A_\ell|} \sum_{i \in A_\ell} x_i^{k+1}$$

- Every sensor i updates:

$$\forall \ell \in \sigma_i, \quad \lambda_{i,|\ell}^{k+1} = \lambda_{i,|\ell}^k + \rho(x_i^{k+1} - \bar{z}_\ell^{k+1})$$

- Each step can be split by agent/block
  - $z_{|\ell}$: $|A_\ell|$-sized block, corresponds to subset ℓ
  - $\lambda_{i,|\ell}$: scalar, corresponds to agent i's entry in subset $\ell \in \sigma_i \triangleq \{l : i \in A_l\}$
- Convergence is immediate from Optimization theory
- Similar algorithm from Primal-Dual when F has a smooth part

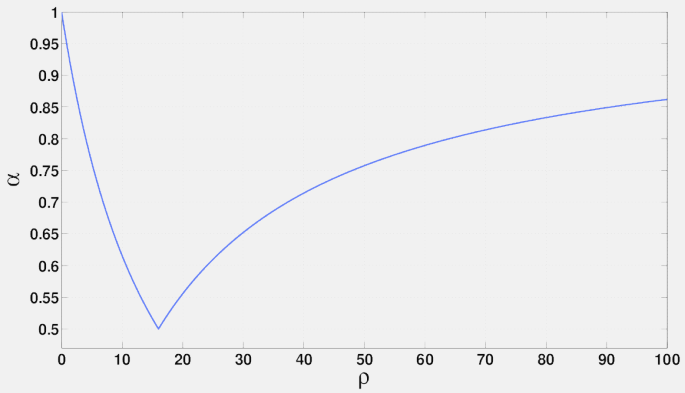


- A drawback of distributed gradient was the decreasing stepsize and slow convergence

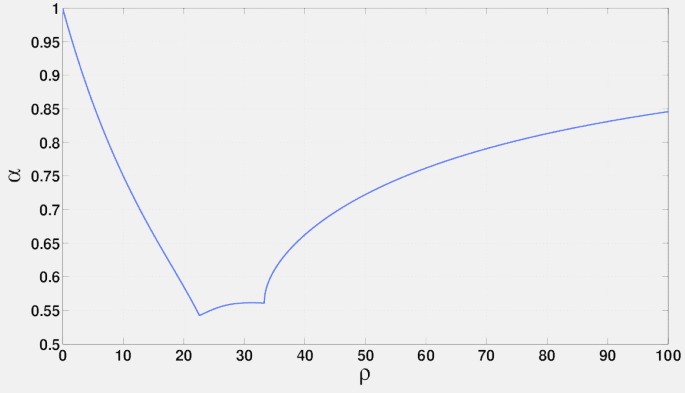
Changes compared with Distributed Gradient:

- Fixed, free parameter $\rho > 0$
- Convergence in $\mathcal{O}(1/k)$ in the general case
- If $\sum_i \nabla^2 f_i(x^\star) > 0$ locally strongly convex at optimum
  Exponential rate with exact rate as the spectral radius of some matrix

Centralized
(1 block)

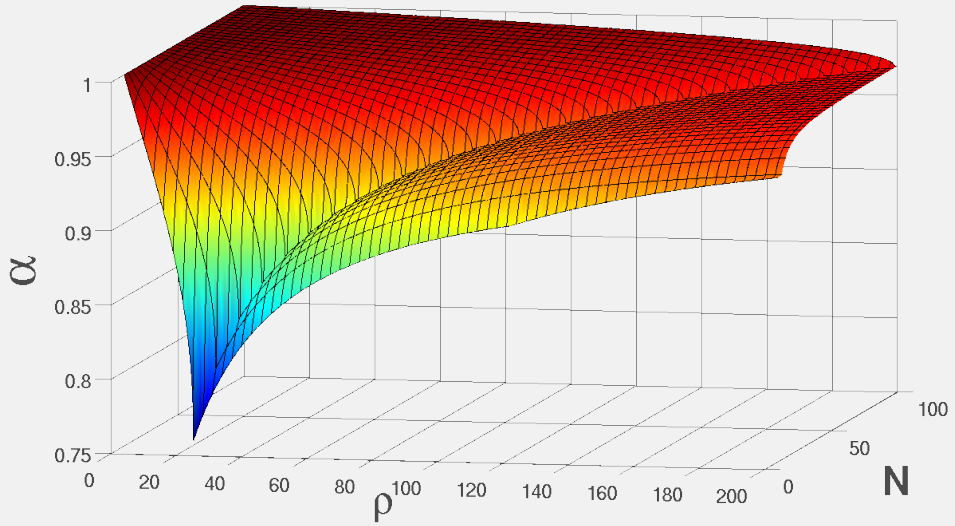


Same $\sigma^2 = 16$



Different $\sigma^2 = 4, 9, 16, 25, 39$

Ring
(N blocks)



Same $\sigma^2 = 16$

Vertical Asymptote!
optimal $\rho = \frac{\sigma^2}{2\sin(2\pi/N)}$

- Any reasonable ρ gives a reasonable speed (~ 1)
- Optimal one is tricky!

- Problem Formulation + Adapted Splitting algorithm = Distributed (Synchronous) Algorithm
- Very intensively investigated
  non-convex case, errors in prox, etc.
- Basis for the Asynchronous case!

References: Optimization & Fixed point theory

Lions & Mercier *Splitting algorithms for the sum of two nonlinear operators*, SIAM NUMA, 1979.
Eckstein & Bertsekas *On the Douglas—Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Prog., 1992.
Condat *A primal–dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms*, JOTA, 2013.

Problem Formulation

Schizas, Ribeiro & Giannakis *Consensus in ad hoc WSNs with noisy links*, IEEE TSP, 2008.

Distributed ADMM Analysis

I. et al. *Explicit convergence rate of a distributed ADMM*, IEEE TAC, 2015.

Recent Advances and Problems in Gossiping

Distributed Gradient Algorithms

- Synchronous Case
- Asynchronous Case
- Application Examples

Advanced Algorithms

- Synchronous Case
- Asynchronous Case
- Application Examples

Distributed Optimization algorithm: Reformulated Problem + Splitting method

$$\min_{x\in\mathbb{R}^N, y\in\mathbb{R}^M} \underbrace{\sum_{\text{nodes } i=1}^{N} f_i(x_i)}_{F(x)} + \underbrace{\sum_{\text{areas } \ell=1}^{L} \iota_{\text{Span}(\mathbf{1})}\left(y_{|\ell}\right)}_{G(y)}$$

$$\text{s.t. } Mx = y$$

- This formulation + a splitting method allowed to produce a distributed algorithm
  information exchanges were only local, supported by the communication graph
- Link between i) entries of the variables and ii) communications areas

Distributed ADMM

At each clock tick k:

Every sensor i performs a minimization: $x_i^{k+1} = \arg\min_x \left\{ f_i(x) + \frac{\rho}{2} \sum_{\ell\in\sigma_i} \left(x_i - \bar{z}_{|\ell}^k + \frac{\lambda_{i,|\ell}^k}{\rho} \right)^2 \right\}$

Every subset A_ℓ computes its average: $\bar{z}_{|\ell}^{k+1} = \frac{1}{|A_\ell|} \sum_{i\in A_\ell} x_i^{k+1} \quad (z_{|\ell}^{k+1} = (\bar{z}_{|\ell}^{k+1}, \bar{z}_{|\ell}^{k+1}, .., \bar{z}_{|\ell}^{k+1}))$

Every sensor i updates: $\forall \ell \in \sigma_i, \quad \lambda_{i,|\ell}^{k+1} = \lambda_{i,|\ell}^k + \rho(x_i^{k+1} - \bar{z}_\ell^{k+1})$

Idea to produce randomized gossip algorithm

- Update only some entries corresponding to one/some areas

- a large variety of optimization algorithm can be written as fixed points operations of an *averaged* operator: contraction property coming from monotone operator theory

$$\zeta^{k+1} = \mathsf{T}(\zeta^k)$$

For instance, it is the case for ADMM with $\zeta^k = \rho z^k + \lambda^k$

Splitting: x is split in L blocks along the areas defined previously with the indicator

$$x^{k+1} = \begin{bmatrix} x_{|1}^{k+1} \\ \vdots \\ x_{|\ell}^{k+1} \\ \vdots \\ x_{|L}^{k+1} \end{bmatrix} = \begin{bmatrix} \mathsf{T}_{(1)}(x^k) \\ \vdots \\ \mathsf{T}_{(\ell)}(x^k) \\ \vdots \\ \mathsf{T}_{(L)}(x^k) \end{bmatrix} = \mathsf{T}(x^k)$$

- a large variety of optimization algorithm can be written as fixed points operations of an *averaged* operator: contraction property coming from monotone operator theory

$$\zeta^{k+1} = \mathsf{T}(\zeta^k)$$

For instance, it is the case for ADMM with $\zeta^k = \rho z^k + \lambda^k$

Splitting: x is split in L blocks along the areas defined previously with the indicator

Randomized Descent: update of one block ℓ chosen at random
the update rely on the whole x_k
(contrary to whole descent on one coordinate $x_k^{(i)}$ [Fercoq,Richtarik'15])
the other blocks keep their entries fixed

$$x^{k+1} = \begin{bmatrix} x_{|1}^{k+1} \\ \vdots \\ x_{|\ell}^{k+1} \\ \vdots \\ x_{|L}^{k+1} \end{bmatrix} = \begin{bmatrix} x_{|1}^{k} \\ \vdots \\ \mathsf{T}_{(\ell)}(x^k) \\ \vdots \\ x_{|L}^{k} \end{bmatrix} \triangleq \hat{\mathsf{T}}_{(\ell)}(x^k)$$

Problem: T is *averaged* but $\hat{\mathsf{T}}_{(\ell)}$ is not in general...
Updating only a subset of entries made us lose the contraction property and thus convergence

- Almost sure convergence can be retrieved if the blocs are chosen in an i.i.d. manner

Theorem
Let T be an *averaged* operator.
Let (ξ^k) be an *i.i.d.* process with values in $\{1, ..., L\}$ such that $\mathbb{P}[\xi^1 = \ell] = p_\ell > 0$ for all $\ell = 1, ..., L$. The iterates sequence (x_k) generated by

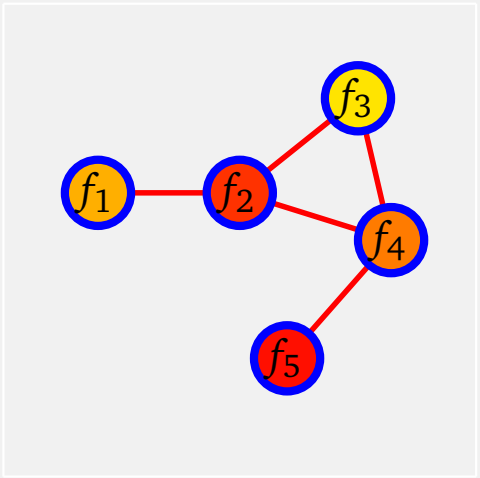
$$x_{k+1} = \hat{\mathsf{T}}_{(\xi^{k+1})}(x_k)$$

converges almost-surely to a solution of our problem.

- Gives algorithms where only some agents communicate per iteration
- The derivation has to be careful: updated entries have to be *as if* the whole operation happens

References: Monotone operators theory + Probabilities

Bianchi, Hachem, & I. *A Coordinate Descent Primal-Dual Algorithm and Application to Distributed Asynchronous Optimization*, CDC 2013 and IEEE TAC, 2015.
Combettes & Pesquet *Stochastic quasi-Fejér block-coordinate fixed point iterations with random sweeping*, SIOPT, 2015.



Asynchronous Optimization w/ ADMM

At each clock tick k, let ξ^{k+1} be the index of the active block:

- Every sensor $i \in A_{\xi^{k+1}}$ of the block computes:

$$x_i^{k+1} = \underset{x}{\text{argmin}} \left\{ f_i(x) + \frac{\rho}{2} \sum_{\ell \in \sigma_i} \left(x_i - \bar{z}_{|\ell}^k + \frac{\lambda_{i,|\ell}^k}{\rho} \right)^2 \right\}$$

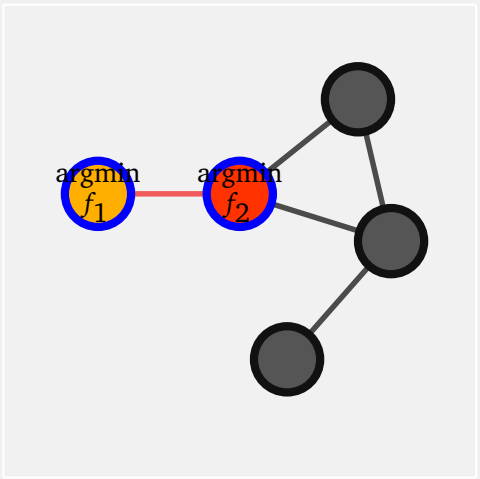
- The block computes its average:

$$\bar{z}_{|\xi^{k+1}}^{k+1} = \frac{1}{|A_{\xi^{k+1}}|} \sum_{i \in A_{\xi^{k+1}}} x_i^{k+1}$$

- Every sensor $i \in A_{\xi^{k+1}}$ of the block updates:

$$\lambda_{i,|\xi^{k+1}}^{k+1} = \lambda_{i,|\xi^{k+1}}^{k} + \rho(x_i^{k+1} - \bar{z}_{\xi^{k+1}}^{k+1})$$

- Exchange similar to Random Gossip for average if a link is selected



Asynchronous Optimization w/ ADMM

At each clock tick k, let ξ^{k+1} be the index of the active block:

- Every sensor $i \in A_{\xi^{k+1}}$ of the block computes:

$$x_i^{k+1} = \underset{x}{\text{argmin}} \left\{ f_i(x) + \frac{\rho}{2} \sum_{\ell \in \sigma_i} \left(x_i - \bar{z}_{|\ell}^k + \frac{\lambda_{i,|\ell}^k}{\rho} \right)^2 \right\}$$

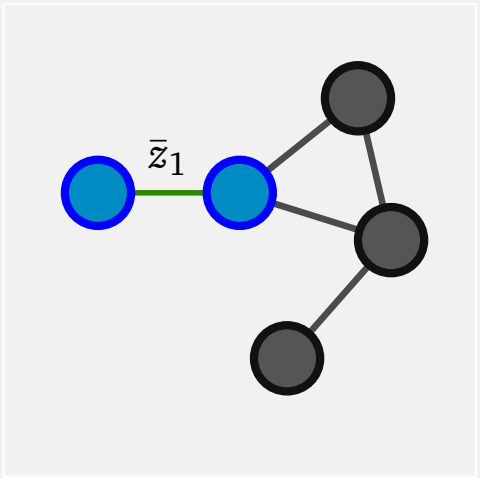
- The block computes its average:

$$\bar{z}_{|\xi^{k+1}}^{k+1} = \frac{1}{|A_{\xi^{k+1}}|} \sum_{i \in A_{\xi^{k+1}}} x_i^{k+1}$$

- Every sensor $i \in A_{\xi^{k+1}}$ of the block updates:

$$\lambda_{i,|\xi^{k+1}}^{k+1} = \lambda_{i,|\xi^{k+1}}^{k} + \rho(x_i^{k+1} - \bar{z}_{\xi^{k+1}}^{k+1})$$

- Exchange similar to Random Gossip for average if a link is selected



Asynchronous Optimization w/ ADMM

At each clock tick k, let ξ^{k+1} be the index of the active block:

- Every sensor $i \in A_{\xi^{k+1}}$ of the block computes:

$$x_i^{k+1} = \underset{x}{\operatorname{argmin}} \left\{ f_i(x) + \frac{\rho}{2} \sum_{\ell \in \sigma_i} \left(x_i - \bar{z}_{|\ell}^k + \frac{\lambda_{i,|\ell}^k}{\rho} \right)^2 \right\}$$

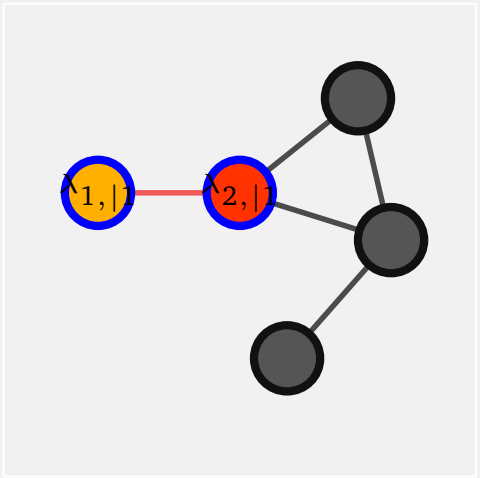
- The block computes its average:

$$\bar{z}_{|\xi^{k+1}}^{k+1} = \frac{1}{|A_{\xi^{k+1}}|} \sum_{i \in A_{\xi^{k+1}}} x_i^{k+1}$$

- Every sensor $i \in A_{\xi^{k+1}}$ of the block updates:

$$\lambda_{i,|\xi^{k+1}}^{k+1} = \lambda_{i,|\xi^{k+1}}^{k} + \rho(x_i^{k+1} - \bar{z}_{\xi^{k+1}}^{k+1})$$

- Exchange similar to Random Gossip for average if a link is selected



Asynchronous Optimization w/ ADMM

At each clock tick k, let ξ^{k+1} be the index of the active block:

- Every sensor $i \in A_{\xi^{k+1}}$ of the block computes:

$$x_i^{k+1} = \underset{x}{\text{argmin}} \left\{ f_i(x) + \frac{\rho}{2} \sum_{\ell \in \sigma_i} \left(x_i - \bar{z}_{|\ell}^k + \frac{\lambda_{i,|\ell}^k}{\rho} \right)^2 \right\}$$

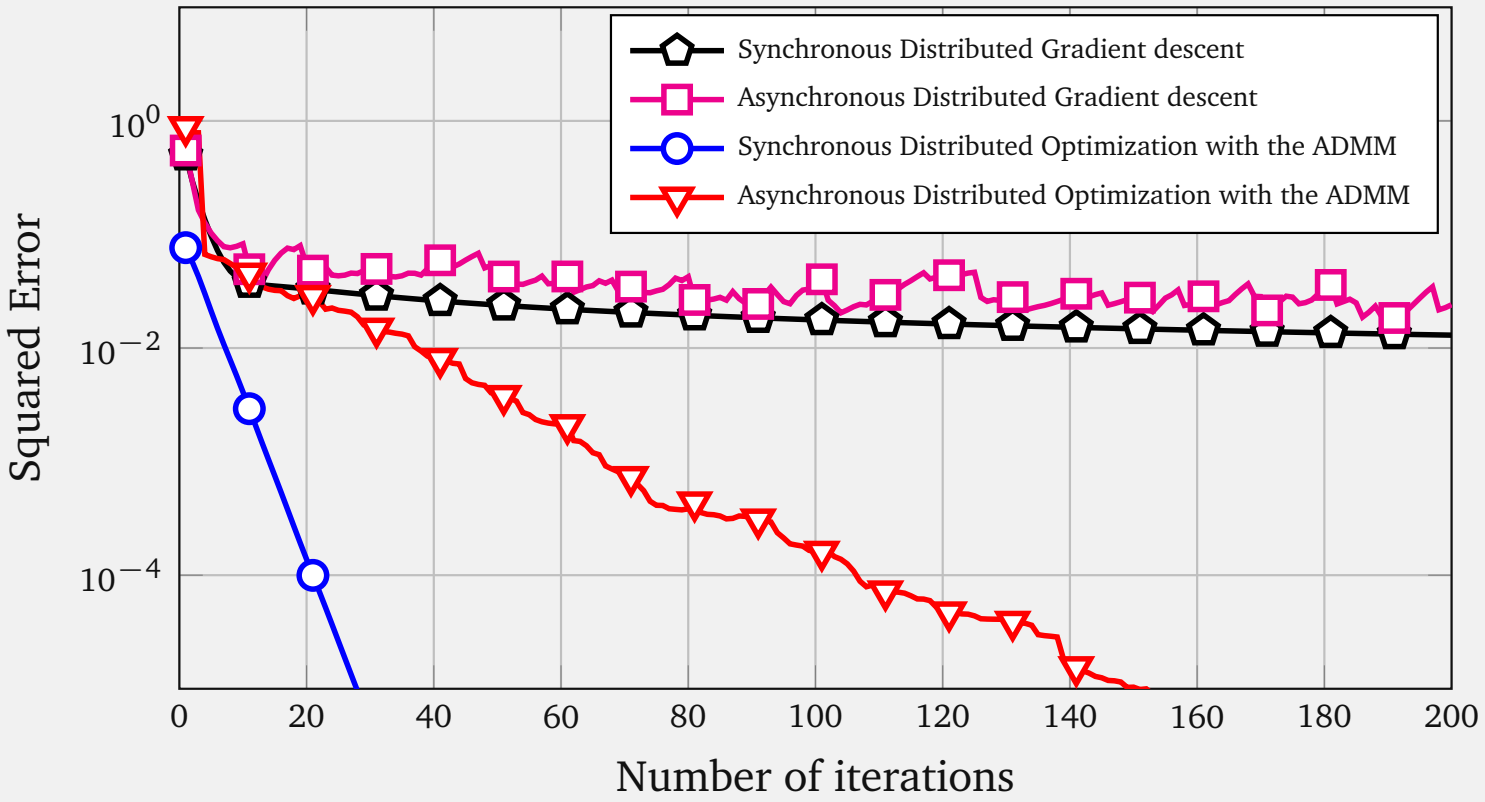
- The block computes its average:

$$\bar{z}_{|\xi^{k+1}}^{k+1} = \frac{1}{|A_{\xi^{k+1}}|} \sum_{i \in A_{\xi^{k+1}}} x_i^{k+1}$$

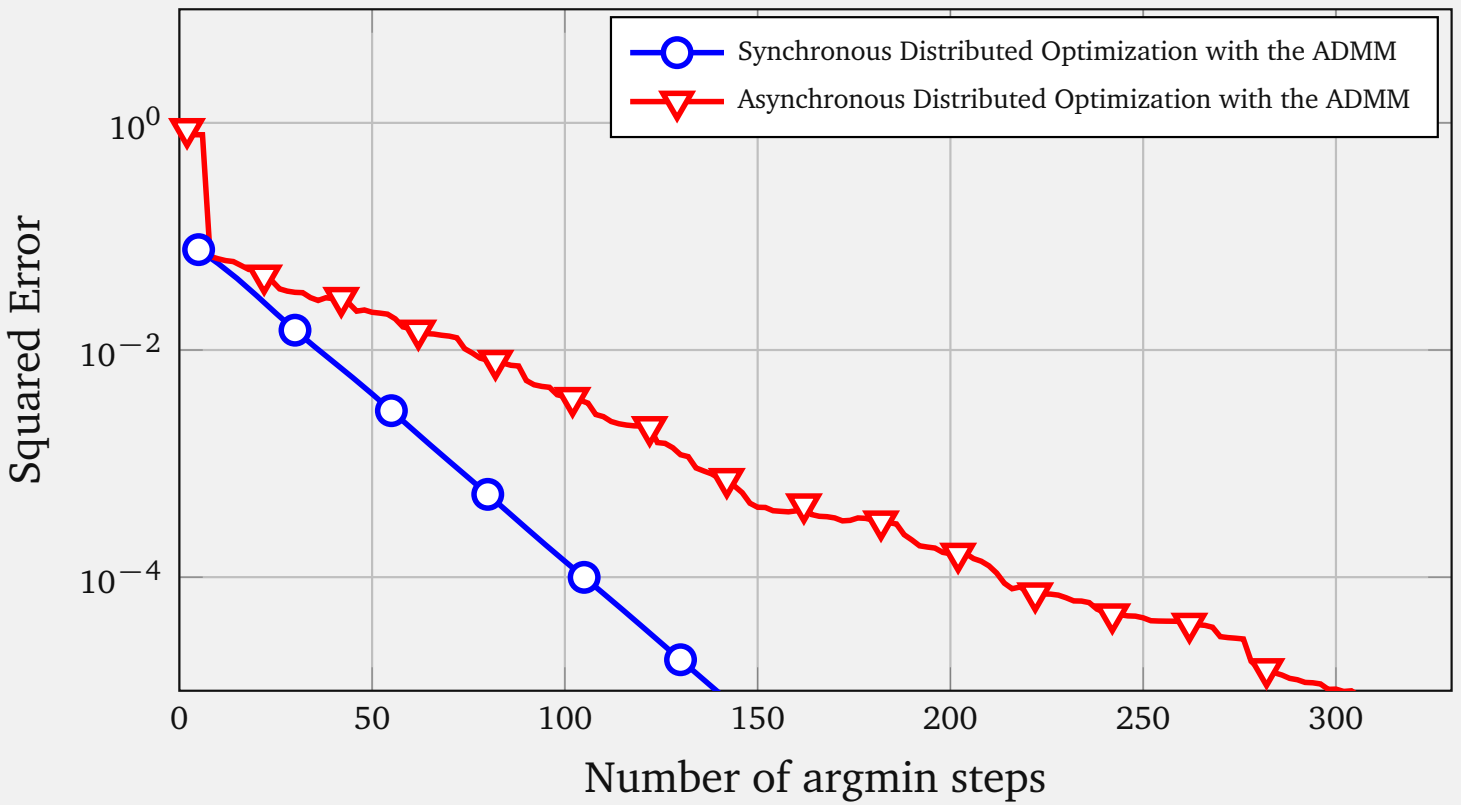
- Every sensor $i \in A_{\xi^{k+1}}$ of the block updates:

$$\lambda_{i,|\xi^{k+1}}^{k+1} = \lambda_{i,|\xi^{k+1}}^k + \rho(x_i^{k+1} - \bar{z}_{\xi^{k+1}}^{k+1})$$

- Exchange similar to Random Gossip for average if a link is selected



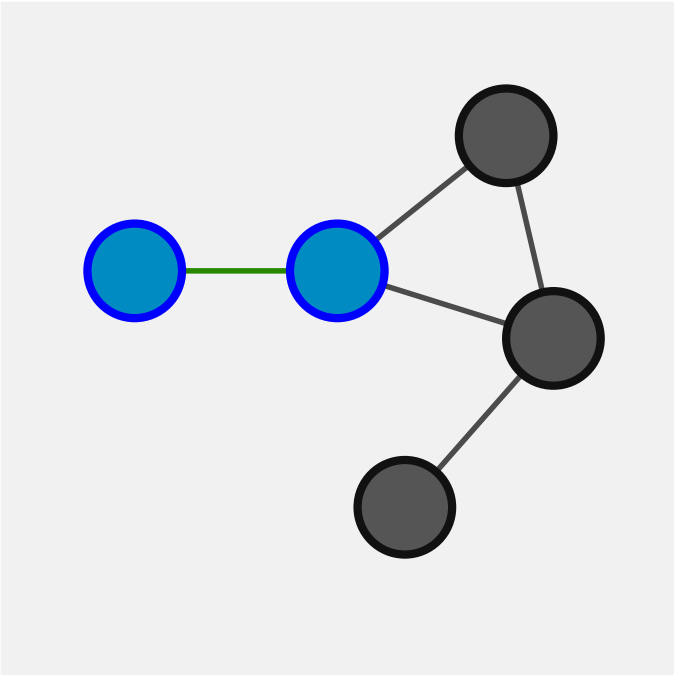
- Synchronous ADMM: 1 iteration = N argmin + L block-averaging
- Asynchronous ADMM: 1 iteration = $|A_{\xi^k}|$ argmin + 1 block-averaging



- Synchronous ADMM: 1 iteration = N argmin + L block-averaging
- Asynchronous ADMM: 1 iteration = $|A_{\xi^k}|$ argmin + 1 block-averaging

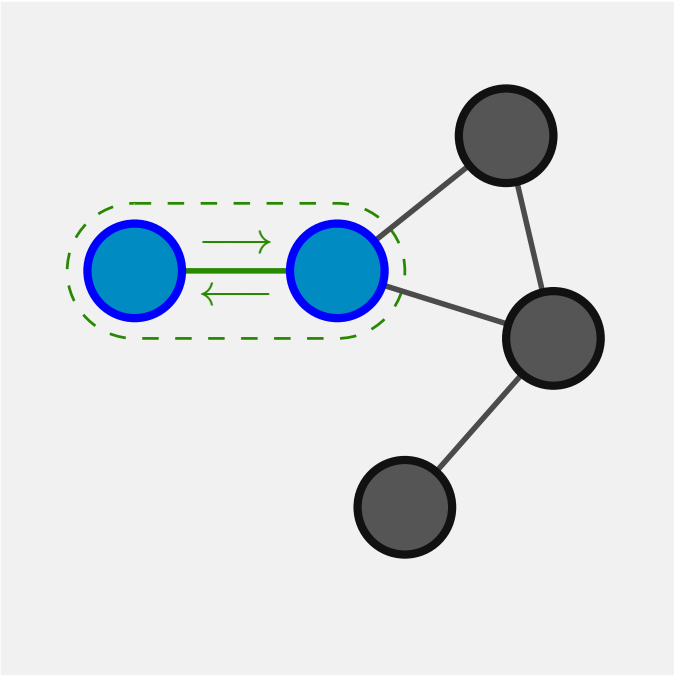
- This setup enables to deal with a large variety of situations

- This setup enables to deal with a large variety of situations



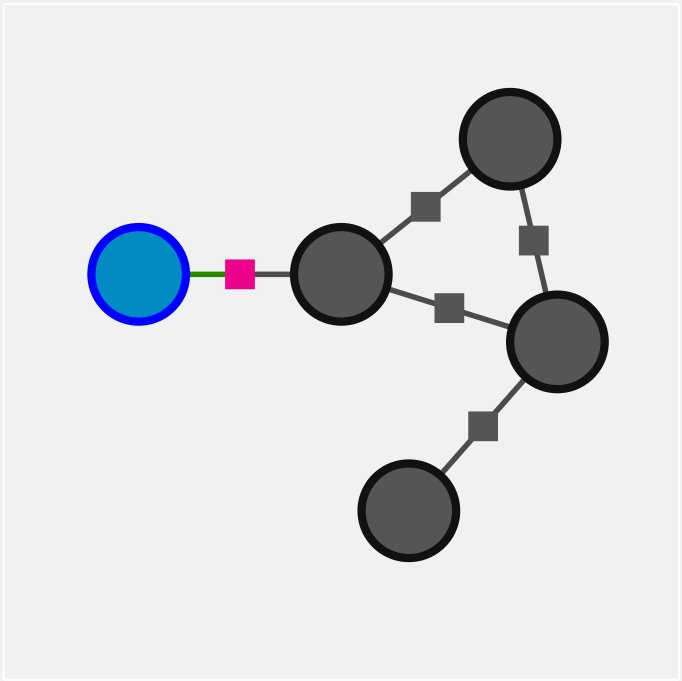
- Distributed Optimization using local coordinators

- This setup enables to deal with a large variety of situations



- Distributed Optimization using local coordinators

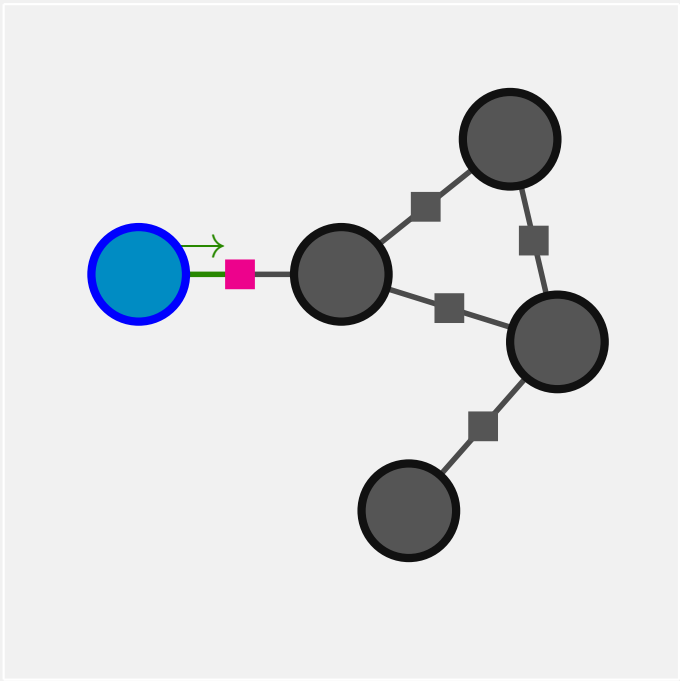
- This setup enables to deal with a large variety of situations



- Distributed Optimization using local coordinators
- Distributed Optimization with *One-Way* communications

By adding dummy nodes
with constant functions
also enables network failures

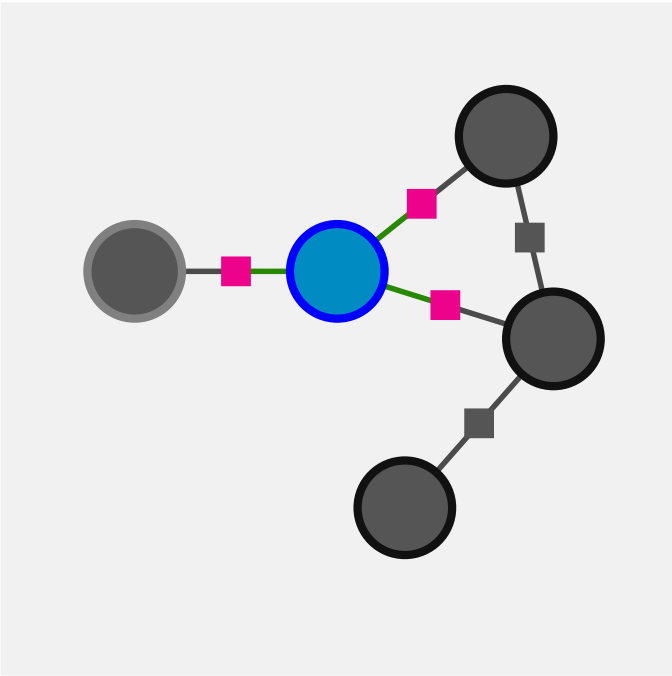
- This setup enables to deal with a large variety of situations



- Distributed Optimization using local coordinators
- Distributed Optimization with *One-Way* communications

By adding dummy nodes
with constant functions
also enables network failures

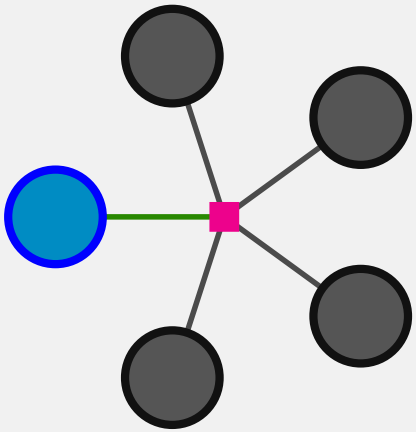
- This setup enables to deal with a large variety of situations



- Distributed Optimization using local coordinators
- Distributed Optimization with *One-Way* communications

By adding dummy nodes
with constant functions
also enables network failures

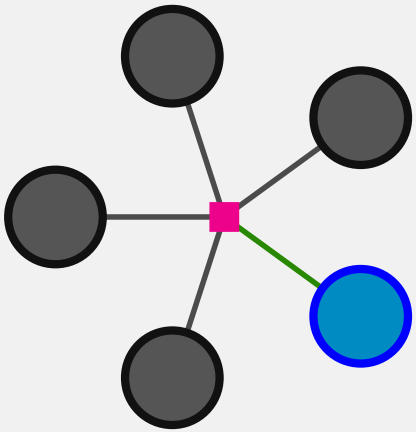
- This setup enables to deal with a large variety of situations



- Distributed Optimization using local coordinators
- Distributed Optimization with *One-Way* communications
- Mini-batch optimization/learning

The network is then just an artifact

- This setup enables to deal with a large variety of situations



- Distributed Optimization using local coordinators
- Distributed Optimization with *One-Way* communications
- Mini-batch optimization/learning

The network is then just an artifact

- Problem Formulation + Adapted Splitting algorithm
  + Randomized Coordinate Descent = Distributed Asynchronous Algorithm
- Almost sure convergence; MSE in $\mathcal{O}(1/k)$
- Highly flexible formulation
  Broadcast communications, change of Metric = change of averaging coefficients
  Sometimes at the expense of performance compared to dedicated algorithms

References: Monotone operators theory + Probabilities

Bianchi, Hachem, & I. *A Coordinate Descent Primal-Dual Algorithm and Application to Distributed Asynchronous Optimization*, CDC 2013 and IEEE TAC, 2015.
Wei & Ozdaglar *On the $O(1/k)$ convergence of asynchronous distributed ADMM*, Arxiv, 2014.
Chouzenoux, Pesquet, & Repetti *A block coordinate variable metric forward–backward algorithm*, J. Glob. Optim., 2013.

Recent Advances and Problems in Gossiping

Distributed Gradient Algorithms

- Synchronous Case
- Asynchronous Case
- Application Examples

Advanced Algorithms

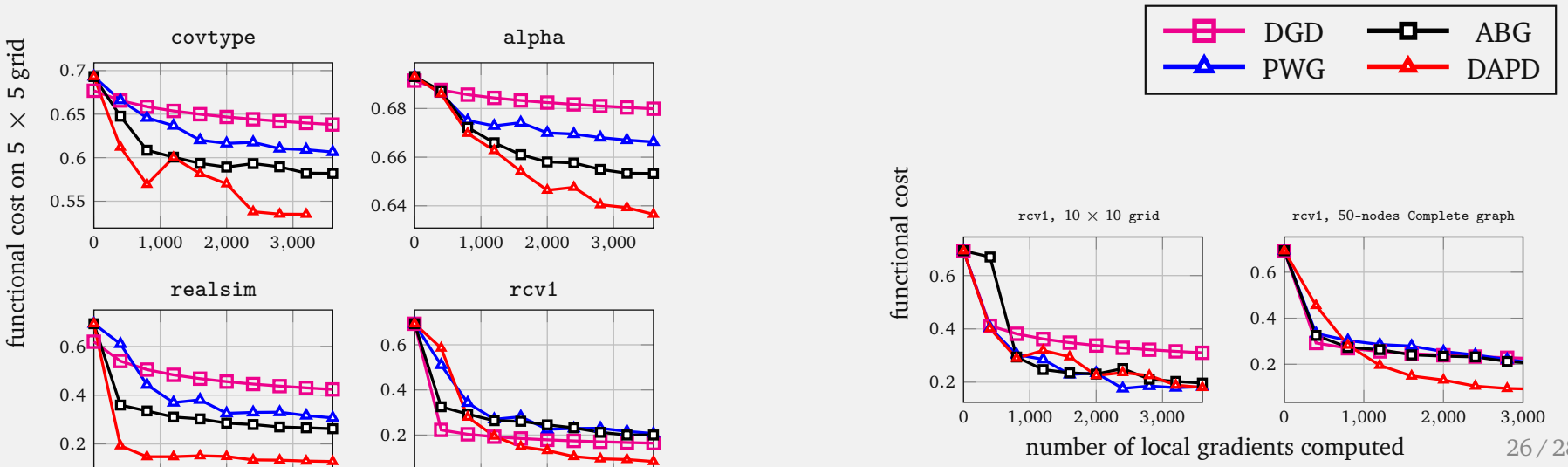
- Synchronous Case
- Asynchronous Case
- Application Examples

Problem: $\min_{\mathbf{x}\in\mathbb{R}^p} \frac{1}{m}\sum_{t=1}^{m} \log\left(1+\mathrm{e}^{-y_t \mathbf{a}_t^{\mathrm{T}}\mathbf{x}}\right) + \mu\|\mathbf{x}\|^2$

Reformulation: $\min_{\mathbf{x}\in\mathbb{R}^{Np}} \sum_{n=1}^{N} \Big(\underbrace{\sum_{t\in\mathcal{B}_n} \frac{1}{m} \log\left(1+\mathrm{e}^{-y_t \mathbf{a}_t^{\mathrm{T}}\mathbf{x}_n}\right)}_{f_n(x_n)} + \underbrace{\frac{\mu}{2N}\|\mathbf{x}_n\|_2^2}_{g_n(x_n)} \Big) + \sum_{\epsilon\in E} \iota_{\mathcal{C}_2}(y_\epsilon)$

DAPD Formulation + L. Condat's Primal-Dual algorithm + Randomization

- Select one (or more) agent n:
  - For all $m \sim n$, do $\lambda_{\{n,m\}}^{k+1}(n) = \frac{\lambda_{\{n,m\}}^{k}(n) - \lambda_{\{n,m\}}^{k}(m)}{2} + \frac{x_n^k - x_m^k}{2\rho}$
  - $x_n^{k+1} = \mathbf{prox}_{\tau g_n/d_n}\left[(1-\tau\rho^{-1})x_n^k - \frac{\tau}{d_n}\nabla f_n(x_n^k) + \frac{\tau}{d_n}\sum_{m\sim n}(\rho^{-1}x_m^k + \lambda_{\{n,m\}}^k(m))\right]$
  - For all $m \sim n$, send $\{x_n^{k+1}, \lambda_{\{n,m\}}^{k+1}(n)\}$ to Neighbor m.
- Other agents stay put.



Give n agents a real number $(a_i)_{i=1,..,N}$.

What does the following algorithm do?

$$\text{all agents do:} \quad x_i^{k+1} = \begin{cases} \bar{x}^k - y_i^k + \beta & \text{if } \bar{x}^k - y_i^k + \beta < a_i \\ \bar{x}^k - y_i^k - 1 & \text{if } \bar{x}^k - y_i^k - 1 > a_i \\ a_i & \text{elsewhere} \end{cases} .$$

$$y^{k+1} = y^k + x^{k+1} - \bar{x}^{k+1} \qquad \bar{x}^k = 1/N \sum_{i=1}^{N} x_i^k$$

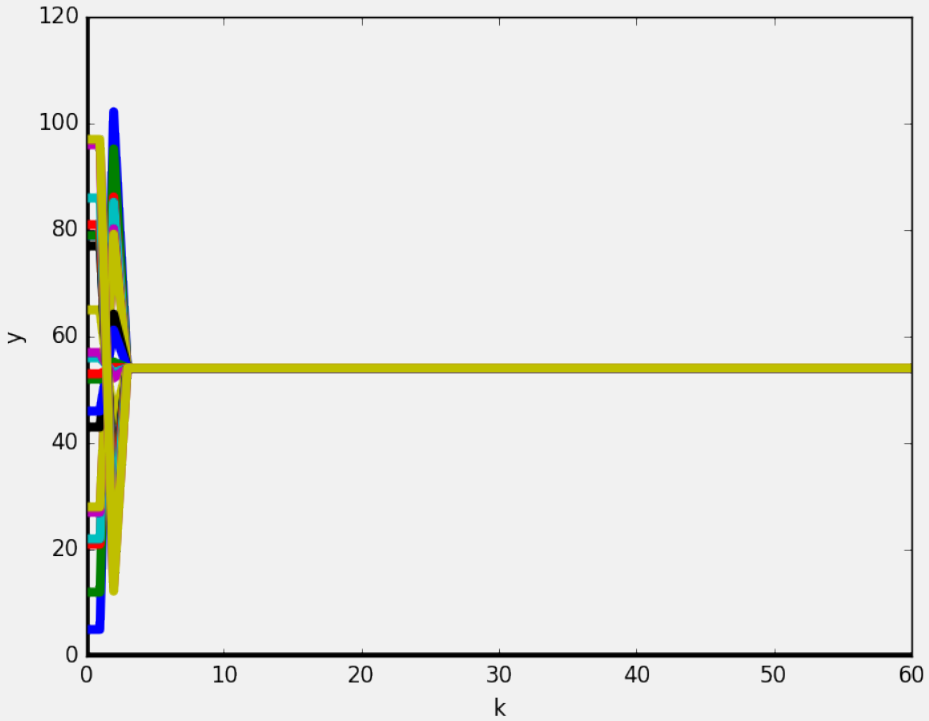
Give n agents a real number $(a_i)_{i=1,..,N}$.

What does the following algorithm do?

$$\text{all agents do: } x_i^{k+1} = \begin{cases} \bar{x}^k - y_i^k + \beta & \text{if } \bar{x}^k - y_i^k + \beta < a_i \\ \bar{x}^k - y_i^k - 1 & \text{if } \bar{x}^k - y_i^k - 1 > a_i \\ a_i & \text{elsewhere} \end{cases} .$$

$$y^{k+1} = y^k + x^{k+1} - \bar{x}^{k+1} \qquad \bar{x}^k = 1/N \sum_{i=1}^{N} x_i^k$$

Hint 1: 20 agents with values in $[0, 100]$



$\beta = 1$

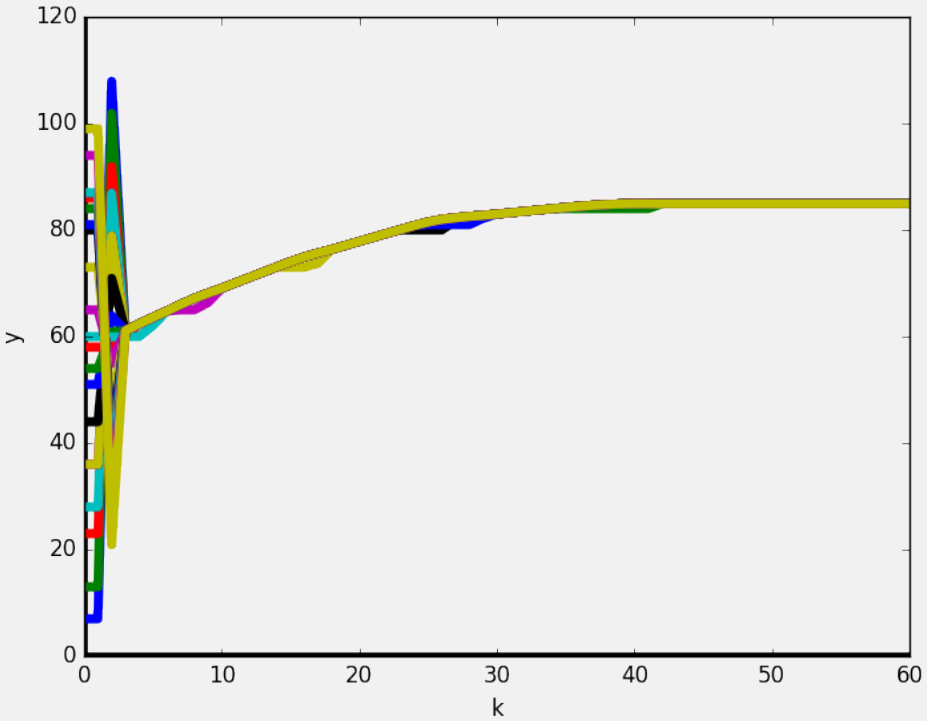
Give n agents a real number $(a_i)_{i=1,..,N}$.

What does the following algorithm do?

$$\text{all agents do: } x_i^{k+1} = \begin{cases} \bar{x}^k - y_i^k + \beta & \text{if } \bar{x}^k - y_i^k + \beta < a_i \\ \bar{x}^k - y_i^k - 1 & \text{if } \bar{x}^k - y_i^k - 1 > a_i \\ a_i & \text{elsewhere} \end{cases} .$$

$$y^{k+1} = y^k + x^{k+1} - \bar{x}^{k+1} \qquad \bar{x}^k = 1/N \sum_{i=1}^{N} x_i^k$$

Hint 1: 20 agents with values in $[0, 100]$



$\beta = 4$

Give n agents a real number $(a_i)_{i=1,..,N}$.

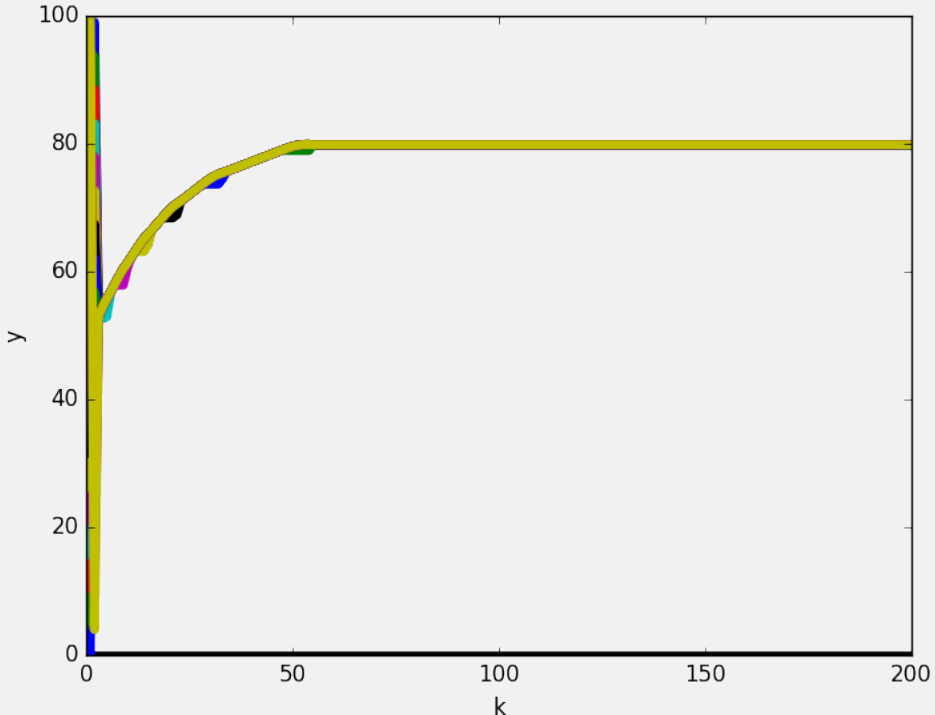
What does the following algorithm do?

$$\text{all agents do: } x_i^{k+1} = \begin{cases} \bar{x}^k - \lambda_i^k/\rho + \beta/\rho & \text{if } \bar{x}^k - \lambda_i^k/\rho + \beta/\rho < a_i \\ \bar{x}^k - \lambda_i^k/\rho - 1/\rho & \text{if } \bar{x}^k - \lambda_i^k/\rho - 1/\rho > a_i \\ a_i & \text{elsewhere} \end{cases}.$$

$$\lambda^{k+1} = \lambda^k + \rho(x^{k+1} - \bar{x}^{k+1}) \qquad \bar{x}^k = 1/N \sum_{i=1}^{N} x_i^k$$

Hint 2: It is ADMM on $\sum_i f_i(x_i) + \iota_{\mathcal{C}}(x)$

General remark: ρ controls the tradeoff between consensus and local minimization.



$\beta = 4, \rho = 1$
Sought value:
between 79 and 84.

Give n agents a real number $(a_i)_{i=1,..,N}$.

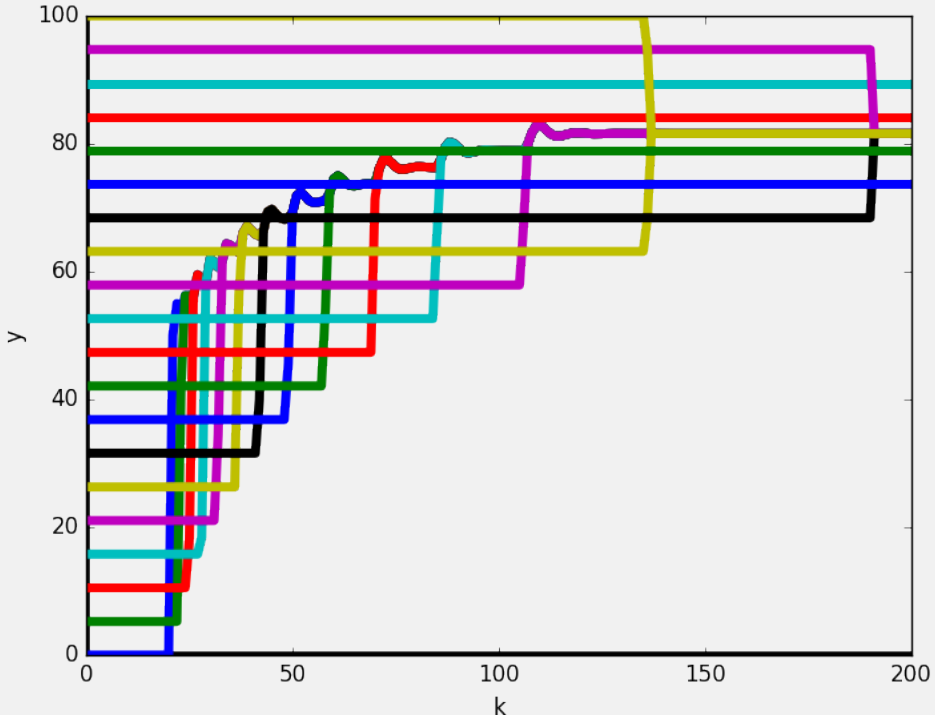
What does the following algorithm do?

$$\text{all agents do:} \quad x_i^{k+1} = \begin{cases} \bar{x}^k - \lambda_i^k/\rho + \beta/\rho & \text{if } \bar{x}^k - \lambda_i^k/\rho + \beta/\rho < a_i \\ \bar{x}^k - \lambda_i^k/\rho - 1/\rho & \text{if } \bar{x}^k - \lambda_i^k/\rho - 1/\rho > a_i \\ a_i & \text{elsewhere} \end{cases} .$$

$$\lambda^{k+1} = \lambda^k + \rho(x^{k+1} - \bar{x}^{k+1}) \qquad \bar{x}^k = 1/N \sum_{i=1}^{N} x_i^k$$

Hint 2: It is ADMM on $\sum_i f_i(x_i) + \iota_{\mathcal{C}}(x)$

General remark: ρ controls the tradeoff between consensus and local minimization.



$\beta = 4, \rho = 0.001$
Sought value:
between 79 and 84.

Give n agents a real number $(a_i)_{i=1,..,N}$.

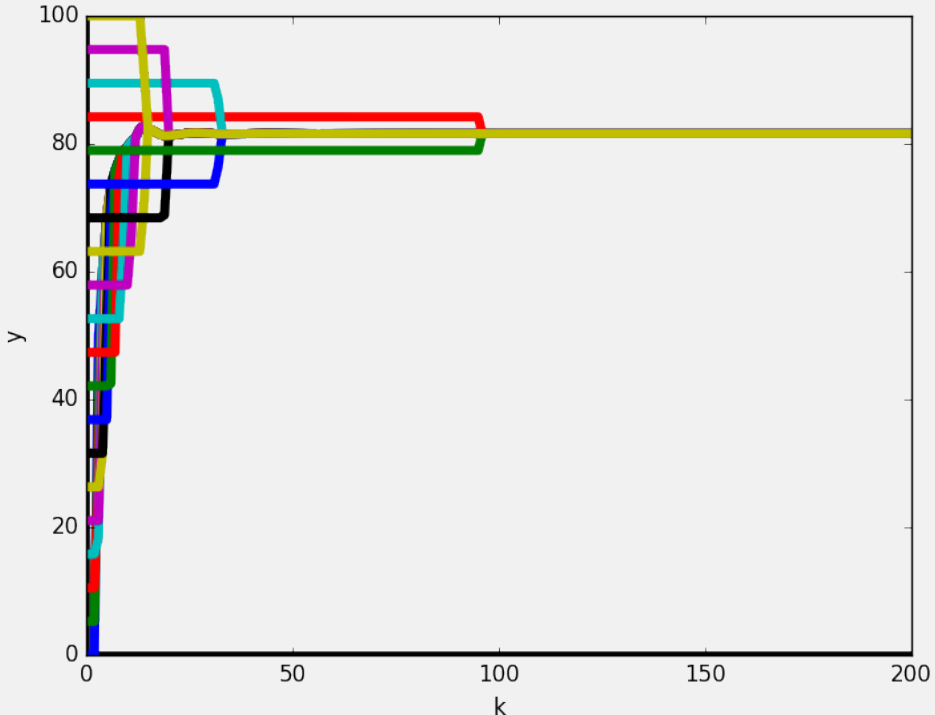
What does the following algorithm do?

$$\text{all agents do:} \quad x_i^{k+1} = \begin{cases} \bar{x}^k - \lambda_i^k/\rho + \beta/\rho & \text{if } \bar{x}^k - \lambda_i^k/\rho + \beta/\rho < a_i \\ \bar{x}^k - \lambda_i^k/\rho - 1/\rho & \text{if } \bar{x}^k - \lambda_i^k/\rho - 1/\rho > a_i \\ a_i & \text{elsewhere} \end{cases} .$$

$$\lambda^{k+1} = \lambda^k + \rho(x^{k+1} - \bar{x}^{k+1}) \qquad \bar{x}^k = 1/N \sum_{i=1}^{N} x_i^k$$

Hint 2: It is ADMM on $\sum_i f_i(x_i) + \iota_{\mathcal{C}}(x)$

General remark: ρ controls the tradeoff between consensus and local minimization.



$\beta = 4, \rho = 0.01$
Sought value:
between 79 and 84.

Give n agents a real number $(a_i)_{i=1,..,N}$.

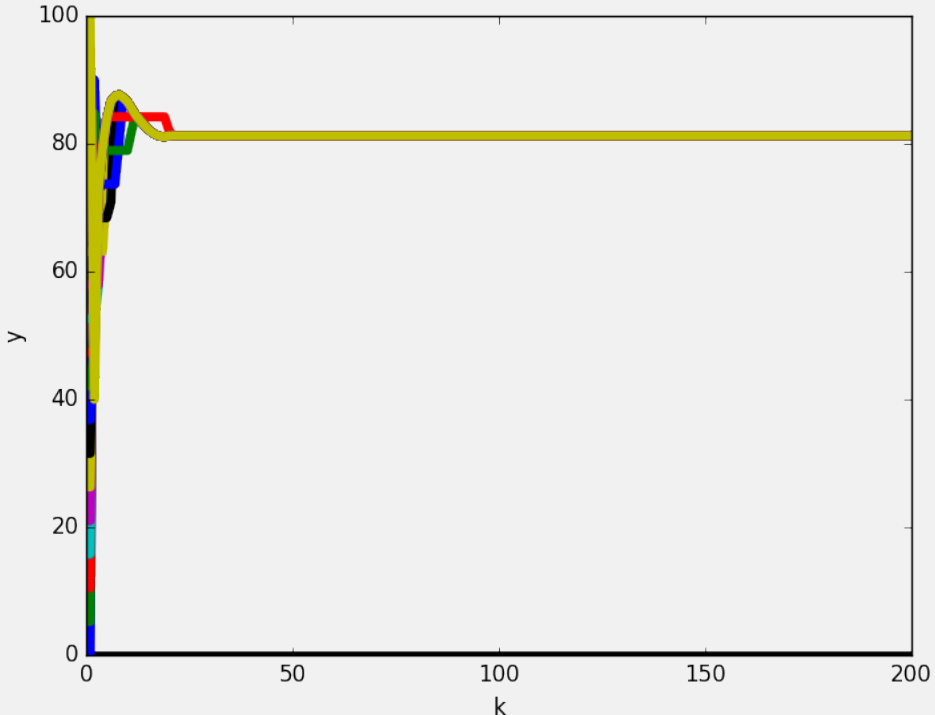
What does the following algorithm do?

$$\text{all agents do:} \quad x_i^{k+1} = \begin{cases} \bar{x}^k - \lambda_i^k/\rho + \beta/\rho & \text{if } \bar{x}^k - \lambda_i^k/\rho + \beta/\rho < a_i \\ \bar{x}^k - \lambda_i^k/\rho - 1/\rho & \text{if } \bar{x}^k - \lambda_i^k/\rho - 1/\rho > a_i \\ a_i & \text{elsewhere} \end{cases} .$$

$$\lambda^{k+1} = \lambda^k + \rho(x^{k+1} - \bar{x}^{k+1}) \qquad \bar{x}^k = 1/N \sum_{i=1}^{N} x_i^k$$

Hint 2: It is ADMM on $\sum_i f_i(x_i) + \iota_{\mathcal{C}}(x)$

General remark: ρ controls the tradeoff between consensus and local minimization.



$\beta = 4, \rho = 0.1$
Sought value:
between 79 and 84.

Give n agents a real number $(a_i)_{i=1,..,N}$.

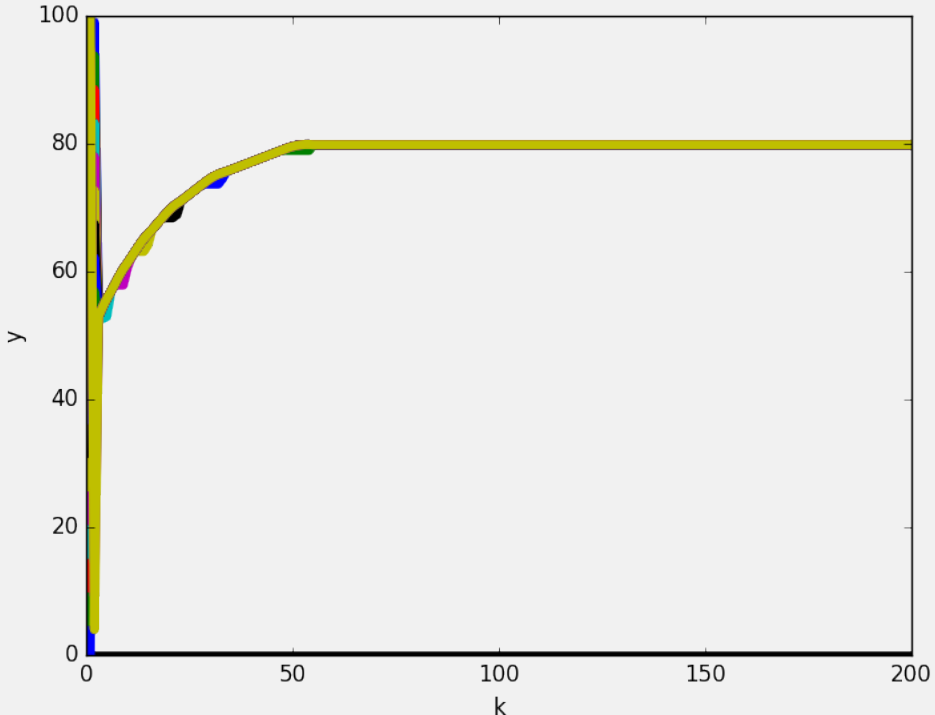
What does the following algorithm do?

$$\text{all agents do: } x_i^{k+1} = \begin{cases} \bar{x}^k - \lambda_i^k/\rho + \beta/\rho & \text{if } \bar{x}^k - \lambda_i^k/\rho + \beta/\rho < a_i \\ \bar{x}^k - \lambda_i^k/\rho - 1/\rho & \text{if } \bar{x}^k - \lambda_i^k/\rho - 1/\rho > a_i \\ a_i & \text{elsewhere} \end{cases}.$$

$$\lambda^{k+1} = \lambda^k + \rho(x^{k+1} - \bar{x}^{k+1}) \qquad \bar{x}^k = 1/N \sum_{i=1}^{N} x_i^k$$

Hint 2: It is ADMM on $\sum_i f_i(x_i) + \iota_{\mathcal{C}}(x)$

General remark: ρ controls the tradeoff between consensus and local minimization.



$\beta = 4, \rho = 1$
Sought value:
between 79 and 84.

Give n agents a real number $(a_i)_{i=1,..,N}$.

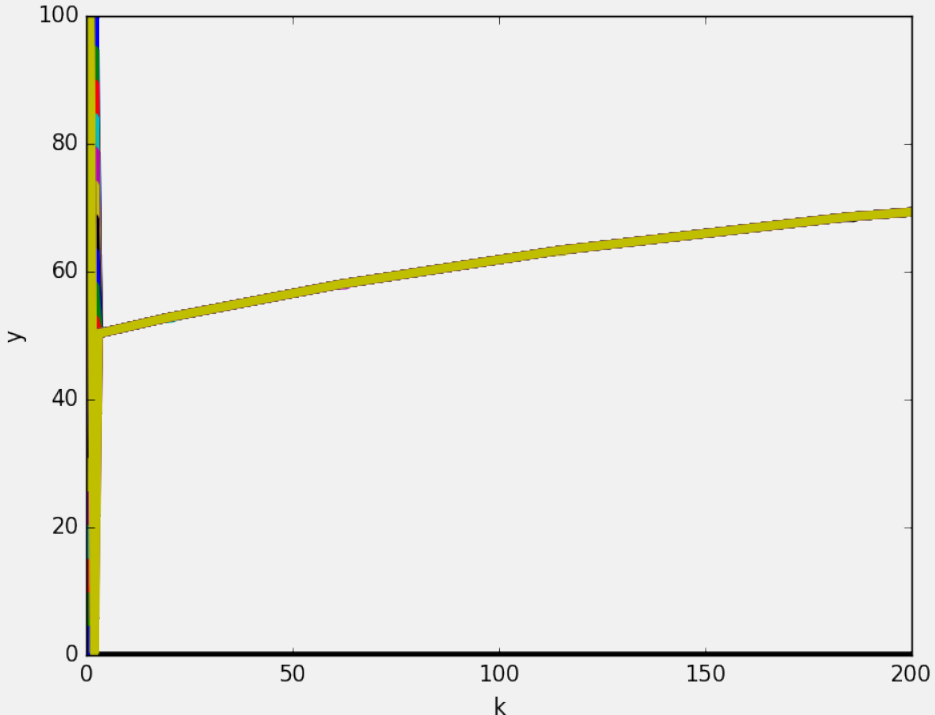
What does the following algorithm do?

$$\text{all agents do:} \quad x_i^{k+1} = \begin{cases} \bar{x}^k - \lambda_i^k/\rho + \beta/\rho & \text{if } \bar{x}^k - \lambda_i^k/\rho + \beta/\rho < a_i \\ \bar{x}^k - \lambda_i^k/\rho - 1/\rho & \text{if } \bar{x}^k - \lambda_i^k/\rho - 1/\rho > a_i \\ a_i & \text{elsewhere} \end{cases} .$$

$$\lambda^{k+1} = \lambda^k + \rho(x^{k+1} - \bar{x}^{k+1}) \qquad \bar{x}^k = 1/N \sum_{i=1}^{N} x_i^k$$

Hint 2: It is ADMM on $\sum_i f_i(x_i) + \iota_{\mathcal{C}}(x)$

General remark: ρ controls the tradeoff between consensus and local minimization.



$\beta = 4, \rho = 10$
Sought value:
between 79 and 84.

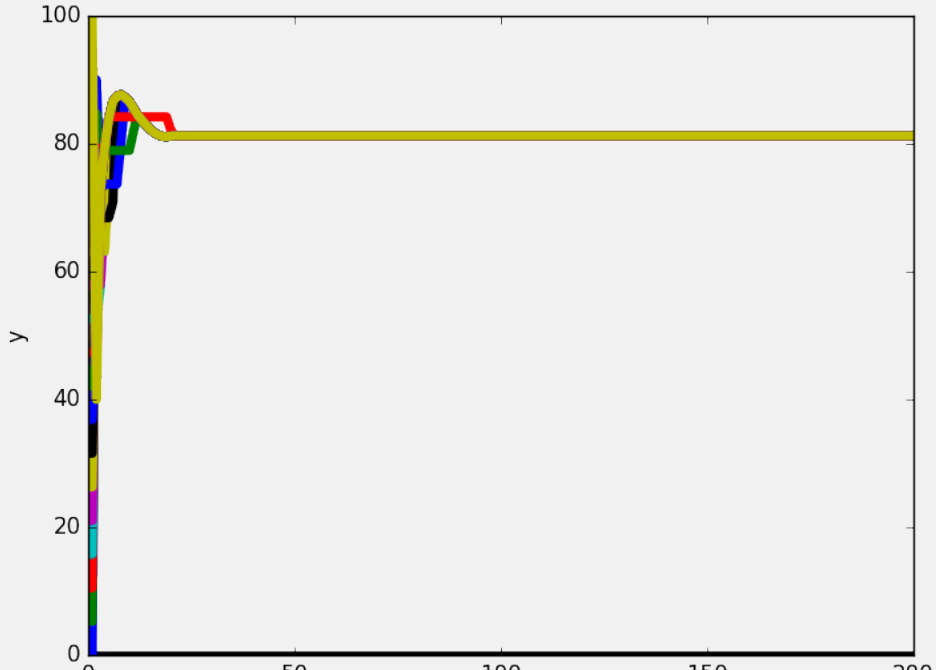
Give n agents a real number $(a_i)_{i=1,..,N}$.

What does the following algorithm do?

$$\text{all agents do:} \quad x_i^{k+1} = \begin{cases} \bar{x}^k - \lambda_i^k/\rho + \beta/\rho & \text{if } \bar{x}^k - \lambda_i^k/\rho + \beta/\rho < a_i \\ \bar{x}^k - \lambda_i^k/\rho - 1/\rho & \text{if } \bar{x}^k - \lambda_i^k/\rho - 1/\rho > a_i \\ a_i & \text{elsewhere} \end{cases} .$$

$$\lambda^{k+1} = \lambda^k + \rho(x^{k+1} - \bar{x}^{k+1}) \qquad \bar{x}^k = 1/N \sum_{i=1}^{N} x_i^k$$

Solution: It is a quantile at 100 $\beta/(1+\beta)$ % !
We saw: how to make it distributed, randomized, and rules for choosing ρ
$\rightarrow$ Complete gossip algorithm for median/quantile estimation!



$\beta = 4, \rho = 0.1$
Sought value:
between 79 and 84.

- Main mathematical tools: Matrix analysis, Optimization
- Guidelines: Gossiping is exponentially fast but disagreement can hurt joint task (e.g. gradient)
- Performance evaluation is an issue
  How to measure properly communication vs. computation time
- `MPI` $\rightarrow$ `Spark` implementation
- a Community of *Optimization and Control* for now...
  arXiv:OC, IEEE TAC and TSP, Automatica, SICON
  Tsitsiklis/Bertsekas ; Nedic/Ozdaglar ; Boyd