DISTRIBUTED ASYNCHRONOUS OPTIMIZATION WITH THE ALTERNATING DIRECTION METHOD OF MULTIPLIERS

Franck IUTZELER

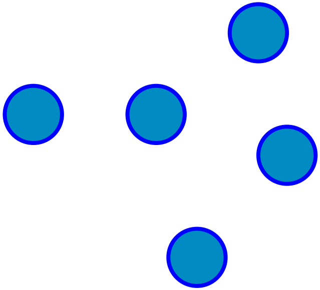
Alcatel-Lucent Chair on Flexible Radio – Supélec

Université Catholique de Louvain – April 28th, 2014

General Context: Distributed Optimization

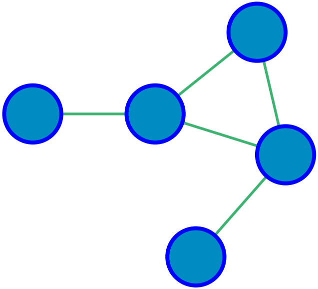
- OPTIMIZATION
  ressource allocation, learning

General Context: Distributed Optimization



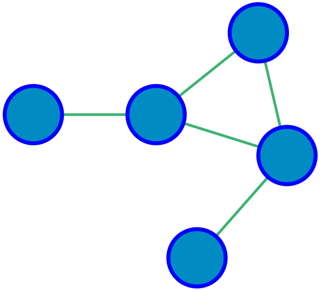
- OPTIMIZATION
  ressource allocation, learning
- AGENTS
  local data, computational power

General Context: Distributed Optimization



- OPTIMIZATION
  ressource allocation, learning
- AGENTS
  local data, computational power
- NETWORK
  communication between agents

General Context: Distributed Optimization



- OPTIMIZATION
  ressource allocation, learning
- AGENTS
  local data, computational power
- NETWORK
  communication between agents

Goal

The goal is to **distributively** reach a **consensus** over a solution of a **global optimization problem** using only **local computations and communications**.

Outline

Outline

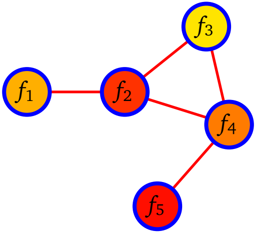
Introduction

Problem

Solving a distributed optimization problem based on the sum of the agents private functions.

Problem:

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



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

Reformulating our problem

A proper problem for distributed optimization

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

- the fact that each sensor only has access to its own cost function;
- the fact that the agents 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

Reformulating our problem

A proper problem for distributed optimization

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

- the fact that each sensor only has access to its own cost function;
- the fact that the agents 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

Reformulating our problem

A proper problem for distributed optimization

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

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

- Starting from the original problem

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

- Adding the fact that the agents only know their own functions

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

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

- We put the constraint into the function to minimize

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

with the *indicator function*

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

Reformulating our problem

A proper problem for distributed optimization

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

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

- Starting from the original problem

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

- Adding the fact that the agents only know their own functions

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

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

- We put the constraint into the function to minimize

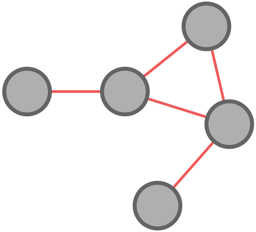
$$\min_{x \in \mathbb{R}^N} F(x) + \iota_{\mathrm{Span}(\mathbb{1})}(x)$$

with the *indicator function*

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

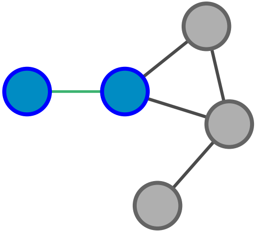
Reformulating our consensus constraint

- We ensure consensus on L *overlapping connected subsets* [Schizas2008]



Reformulating our consensus constraint

- We ensure consensus on L *overlapping connected subsets* [Schizas2008]

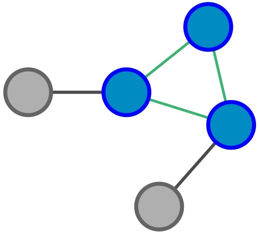


- $A_1 = \{1, 2\}$ $\qquad \mathbf{M}_1 x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

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

Reformulating our consensus constraint

- We ensure consensus on L *overlapping connected subsets* [Schizas2008]

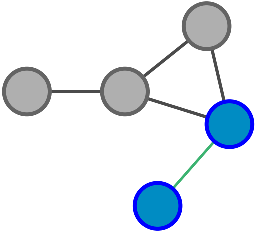


- $A_1 = \{1, 2\}$ $\quad \mathbf{M}_1 x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $A_2 = \{2, 3, 4\}$ $\quad \mathbf{M}_2 x = \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}$

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

Reformulating our consensus constraint

- We ensure consensus on L *overlapping connected subsets* [Schizas2008]

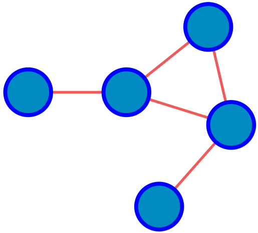


- $A_1 = \{1, 2\}$ $\quad \mathbf{M}_1 x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $A_2 = \{2, 3, 4\}$ $\quad \mathbf{M}_2 x = \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}$
- $A_3 = \{4, 5\}$ $\quad \mathbf{M}_3 x = \begin{bmatrix} x_4 \\ x_5 \end{bmatrix}$

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

Reformulating our consensus constraint

- We ensure consensus on L *overlapping connected subsets* [Schizas2008]

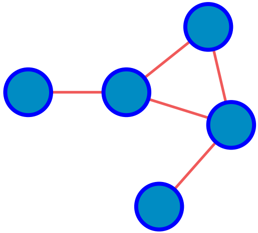


- $A_1 = \{1,2\}$ $\quad \mathbf{M}_1 x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $A_2 = \{2,3,4\}$ $\quad \mathbf{M}_2 x = \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}$
- $A_3 = \{4,5\}$ $\quad \mathbf{M}_3 x = \begin{bmatrix} x_4 \\ x_5 \end{bmatrix}$

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

Reformulating our consensus constraint

- We ensure consensus on L *overlapping connected subsets* [Schizas2008]



- $A_1 = \{1,2\}$ $\quad \mathbf{M}_1 x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $A_2 = \{2,3,4\}$ $\quad \mathbf{M}_2 x = \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}$
- $A_3 = \{4,5\}$ $\quad \mathbf{M}_3 x = \begin{bmatrix} x_4 \\ x_5 \end{bmatrix}$
- $\mathbf{M} \triangleq \begin{bmatrix} \mathbf{M}_1 \\ \mathbf{M}_2 \\ \mathbf{M}_3 \end{bmatrix}$: size $\sum_{\ell=1}^{L} |A_\ell| \triangleq M \times N$

$$\iota_{\mathrm{Span}(\mathbb{1})}\left(\mathbf{M}_1 x\right) + \iota_{\mathrm{Span}(\mathbb{1})}\left(\mathbf{M}_2 x\right) + \iota_{\mathrm{Span}(\mathbb{1})}\left(\mathbf{M}_3 x\right) \triangleq G\left(\mathbf{M} x\right)$$

Equivalent problem

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

Outline

Introducing the Alternating Direction Method of Multipliers

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

- A separable networked separated problem...

Introducing the Alternating Direction Method of Multipliers

$$\begin{aligned}\min_{x\in\mathbb{R}^N, z\in\mathbb{R}^M} \quad & F(x) + G(z) \\ \text{subject to} \quad & \mathbf{M}x = z\end{aligned}$$

- A separable networked separated problem...
- ... that we can split by adding a constraint.

Introducing the Alternating Direction Method of Multipliers

$$\min_{x\in\mathbb{R}^N, z\in\mathbb{R}^M} \quad F(x)+G(z)$$
$$\text{subject to} \quad \mathbf{M}x = z$$

- A separable networked separated problem...
- ... that we can split by adding a constraint.
- As it is constrained, we consider its *Lagrangian*:

$$\mathcal{L}\ (x,z;\lambda) = F(x) + G(z) + \langle \mathbf{M}x - z; \lambda \rangle$$

Introducing the Alternating Direction Method of Multipliers

$$\min_{x\in\mathbb{R}^N, z\in\mathbb{R}^M} \quad F(x) + G(z)$$
$$\text{subject to} \quad \mathbf{M}x = z$$

- A separable networked separated problem...
- ... that we can split by adding a constraint.
- As it is constrained, we consider its **augmented** *Lagrangian*:

$$\mathcal{L}_{\rho}(x, z; \lambda) = F(x) + G(z) + \langle \mathbf{M}x - z; \lambda \rangle + \frac{\rho}{2} \|\mathbf{M}x - z\|_2^2$$

The Alternating Direction Method of Multipliers

The ALTERNATING DIRECTION METHOD OF MULTIPLIERS consists in three steps:

- $x^{k+1} = \underset{x}{\operatorname{argmin}}\, \mathcal{L}_\rho(x, z^k; \lambda^k)$ – *primal* optimum computation w.r.t F
- $z^{k+1} = \underset{z}{\operatorname{argmin}}\, \mathcal{L}_\rho(x^{k+1}, z; \lambda^k)$ – *primal* optimum computation w.r.t. G
- $\lambda^{k+1} = \lambda^k + \rho\left(\mathbf{M}x^{k+1} - z^{k+1}\right)$ – *dual* update

The Alternating Direction Method of Multipliers

The ALTERNATING DIRECTION METHOD OF MULTIPLIERS consists in three steps:

- $x^{k+1} = \underset{x}{\operatorname{argmin}}\, \mathcal{L}_{\rho}(x, z^k; \lambda^k)$

- $z^{k+1} = \underset{z}{\operatorname{argmin}}\, \mathcal{L}_{\rho}(x^{k+1}, z; \lambda^k)$

- $\lambda^{k+1} = \lambda^k + \rho \left(\mathbf{M}x^{k+1} - z^{k+1} \right)$

- The hyper-parameter ρ is **common** to the 3 steps

The Alternating Direction Method of Multipliers

The ALTERNATING DIRECTION METHOD OF MULTIPLIERS consists in three steps:

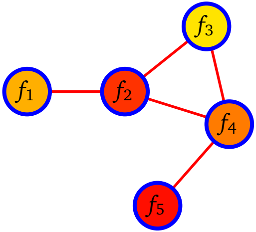
- $$x^{k+1} = \operatorname*{argmin}_x \left\{ F(x) + G(z^k) + \langle \mathbf{M}x - z^k; \lambda^k \rangle + \frac{\rho}{2} \left\| \mathbf{M}x - z^k \right\|_2^2 \right\}$$

- $$z^{k+1} = \operatorname*{argmin}_z \left\{ F(x^{k+1}) + G(z) + \langle \mathbf{M}x^{k+1} - z; \lambda^k \rangle + \frac{\rho}{2} \left\| \mathbf{M}x^{k+1} - z \right\|_2^2 \right\}$$

- $$\lambda^{k+1} = \lambda^k + \rho \left(\mathbf{M}x^{k+1} - z^{k+1} \right)$$

- The hyper-parameter ρ is **common** to the 3 steps

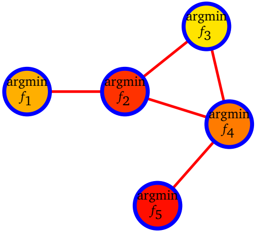
Distributed optimization with the ADMM [Schizas2008]



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 $\ell \in \sigma_i \triangleq \{l : i \in A_l\}$

Distributed optimization with the ADMM [Schizas2008]



Distributed Optimization with ADMM

At each clock tick k:

- Every sensor i performs a minimization:

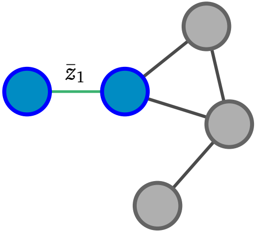
$$x_i^{k+1} = \underset{x}{\operatorname{argmin}} \left\{ f_i(x) + \frac{\rho}{2} \sum_{\ell \in \sigma_i} \left(x_i - \bar{z}_{|\ell}^k + \frac{\lambda_{i,|\ell}^k}{\rho} \right)^2 \right\}$$

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 $\ell \in \sigma_i \triangleq \{l : i \in A_l\}$

Distributed optimization with the ADMM [Schizas2008]



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 $\ell \in \sigma_i \triangleq \{l : i \in A_l\}$

Distributed Optimization with ADMM

At each clock tick k:

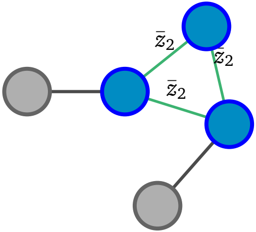
- Every sensor i performs a minimization:

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

- Every subset A_ℓ computes its average:

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

Distributed optimization with the ADMM [Schizas2008]



Distributed Optimization with ADMM

At each clock tick k:

- Every sensor i performs a minimization:

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

- Every subset A_ℓ computes its average:

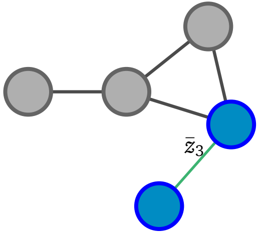
$$\bar{z}^{k+1}_{|\ell} = \frac{1}{|A_\ell|} \sum_{i \in A_\ell} x_i^{k+1}$$

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 $\ell \in \sigma_i \triangleq \{l : i \in A_l\}$

Distributed optimization with the ADMM [Schizas2008]



DISTRIBUTED OPTIMIZATION WITH ADMM

At each clock tick k:

- Every sensor i performs a minimization:

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

- Every subset A_ℓ computes its average:

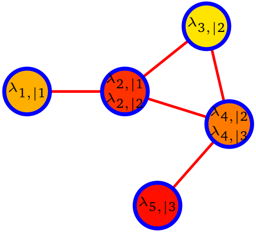
$$\bar{z}_{|\ell}^{k+1} = \frac{1}{|A_\ell|} \sum_{i \in A_\ell} x_i^{k+1}$$

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 $\ell \in \sigma_i \triangleq \{l : i \in A_l\}$

Distributed optimization with the ADMM [Schizas2008]



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 $\ell \in \sigma_i \triangleq \{l : i \in A_l\}$

Distributed Optimization with ADMM

At each clock tick k:

- Every sensor i performs a minimization:

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

- 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:

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

- argmin: costs in computational time
- averaging: costs in networking

Outline

Introducing Monotone Operators

Reasons

- Unified mathematical theory for convex minimization
- Simplicity and Elegance of the proofs
- Intuitive vision that enables to derive new asynchronous algorithms

Introducing Monotone Operators

Definition of a monotone operator

*An **operator** T on $\mathbb{R}^N$ is a set-valued mapping:*

$$\begin{array}{rccl} \mathsf{T}: & \mathbb{R}^N & \to & 2^{\mathbb{R}^N} \\ & x & \mapsto & \mathsf{T}(x) \subset \mathbb{R}^N. \end{array}$$

T *is said to be **monotone** if*

$$\forall (x,y),(x',y') \in \mathsf{T}, \ \ \langle x - x'; y - y' \rangle \geq 0$$

*and **maximal** if it is not strictly contained in any other monotone operator as a subset of* $\mathbb{R}^N \times \mathbb{R}^N$.

- Extension of $\mathbb{R}^N \to \mathbb{R}^N$ monotone functions
- $(x,y) \in \mathsf{T}$ iff $y \in \mathsf{T}(x)$

Example: subdifferential of a convex function h

∂h is a **maximally monotone operator**.
We want to find a **zero** of ∂h.

The resolvent of an operator

Resolvent of an operator T

*The **resolvent** of T is the operator defined as:*

$$\mathsf{J}_{\mathsf{T}} \triangleq (\mathsf{I} + \mathsf{T})^{-1}.$$

- I is the identity operator $\mathsf{I}(x) = x$ and $(x, y) \in \mathsf{T}$ iff $(y, x) \in \mathsf{T}^{-1}$.

Example: subdifferential of a convex function h

Finding a **zero** of $\partial h \Leftrightarrow$ Finding a **fixed point** of $\mathsf{J}_{\partial h}$
Natural fixed-point algorithm:

$$\zeta^{k+1} = \mathsf{J}_{\partial h}(\zeta^{k})$$

We want to know the **contraction properties** of $\mathsf{J}_{\partial h}$.

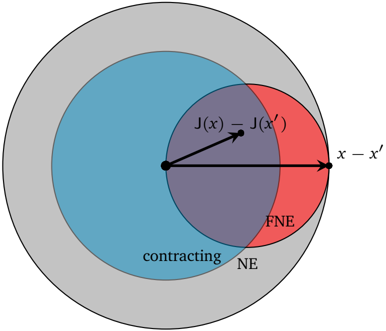
Contraction properties of resolvents

Lemma

J *is the resolvent of a monotone operator iff it is* ***Firmly Non-Expansive (FNE)****:*

$$\forall (x, x'), \quad \langle x - x'; \mathsf{J}(x) - \mathsf{J}(x') \rangle \geq \|\mathsf{J}(x) - \mathsf{J}(x')\|^2.$$

- J is not Banach contracting.



The Proximal point algorithm

Lemma [Rockafellar1976]

Let J *be a FNE operator such that* **fix** $\mathsf{J} \neq \emptyset$*, the sequence generated by*

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

converges to a point of **fix** J.

Example: subdifferential of a convex function h

Iterating $\mathsf{J}_{\partial h}$ leads to the **Proximal Point Algorithm**:

$$x^{k+1} = \mathsf{J}_{\partial h}(x^k) \Leftrightarrow x^{k+1} = \operatorname*{argmin}_x \left\{ h(x) + \frac{1}{2} \left\| x - x^k \right\|^2 \right\}$$

which converges to a zero of ∂h if any.

Solving our problem

Equivalent problem

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

Dual problem

$$\max_{\lambda\in\mathbb{R}^M} \mathcal{D}(\lambda) \triangleq -F^*(-\mathbf{M}^\mathrm{T}\lambda) - G^*(\lambda)$$

Solving our problem with monotone operators

We want to find a **zero** of $\mathsf{U}+\mathsf{V}$

$$-\partial\mathcal{D} = \underbrace{-\mathbf{M}\partial F^*(-\mathbf{M}^\mathrm{T}\cdot)}_{\mathsf{U}} + \underbrace{\partial G^*}_{\mathsf{V}}$$

using the **resolvents** of U and V **separately**.

Lions-Mercier operator and Douglas-Rachford splitting

Lemma [Lions1979]

The resolvent J^{LM} *of Lions-Mercier operator for splitting two operators* U *and* V*:*

- *is defined as* $\mathsf{J}^{LM} \triangleq \mathsf{J}_{\rho\mathsf{U}} \circ (2\mathsf{J}_{\rho\mathsf{V}} - \mathsf{I}) + (\mathsf{I} - \mathsf{J}_{\rho\mathsf{V}})$*;*
- *is FNE if* U *and* V *are monotone;*
- *has a fixed point if* $\mathbf{zer}(\mathsf{U} + \mathsf{V}) \neq \emptyset$.

If $\zeta^\star \in \mathbf{fix}\, \mathsf{J}^{LM}$ *then* $\lambda^\star \triangleq \mathsf{J}_{\rho\mathsf{V}}(\zeta^\star) \in \mathbf{zer}(\mathsf{U} + \mathsf{V})$.

Solving our problem with monotone operators

Iterating J^{LM} with $\mathsf{U} = -\mathbf{M}\partial F^*(-\mathbf{M}^{\mathrm{T}}\cdot)$ and $\mathsf{V} = \partial G^*$ leads to the **ADMM**.

- The iterations of J^{LM} are performed on $\zeta \in \mathbb{R}^M$, problem variables $\lambda \triangleq \mathsf{J}_{\rho\mathsf{V}}(\zeta)$, z and x are intermediate variables.

About the linear convergence rate of Distributed ADMM

- This formalism guided us onto the study of the linear rate

Assumptions

The functions $f_i \in \Gamma_0(\mathbb{R})$. Furthermore, the infimum of our problem is attained at a point $x^\star$ such that the functions f_i are twice differentiable at $x^\star$ and

$$\sum_{i=1}^{N} \nabla^2 f_i(x^\star) > 0.$$

$$\text{Define } \ \mathbf{Q} = \rho\mathbf{M}\left(\begin{bmatrix}\nabla^2 f_1(x_\star) & & \\ & \ddots & \\ & & \nabla^2 f_N(x_\star)\end{bmatrix} + \rho\mathbf{M}^{\mathrm{T}}\mathbf{M}\right)^{-1} \mathbf{M}^\star,$$

$$\mathbf{P} = \begin{bmatrix}\mathbf{J}_{|A_1|} & & \\ & \ddots & \\ & & \mathbf{J}_{|A_L|}\end{bmatrix} \text{ and } \ \mathbf{R} = (\Pi_{span(\mathbf{P}+\mathbf{Q})} - (\mathbf{P}+\mathbf{Q}))(\mathbf{I} - 2\mathbf{P}).$$

About the linear convergence rate of Distributed ADMM

- In the quadratic case, we have $\zeta^{k+1} = \mathbf{R}\zeta^{k+1} + d$ and thus $x^k - x^\star \propto \mathbf{R}^k(\zeta^0 - \zeta^\star)$ so the spectral radius of $\mathbf{R}$ controls the convergence rate.
- In the general case, we can prove that $\mathbf{r}(\mathbf{R})$ still controls the convergence rate.

Theorem

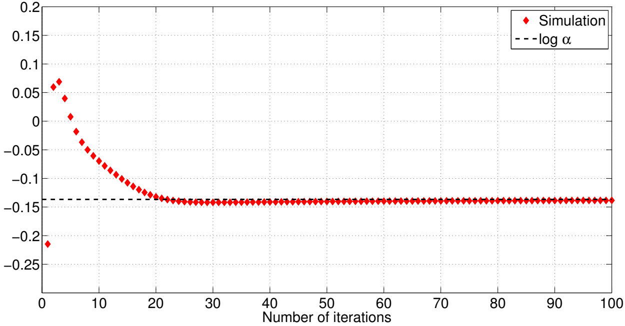
Under the previous assumption, $\alpha = \mathbf{r}(\mathbf{R}) < 1$ and for any initial value (z_0, λ_0) of the ADMM algorithm,

$$\limsup_{k\to\infty} \frac{1}{k} \log \|x_k - 1_N \otimes x_\star\| \leq \log \alpha.$$

F. Iutzeler, P. Bianchi, Ph. Ciblat, W. Hachem, "*Explicit Convergence Rate of a Distributed Alternating Direction Method of Multipliers*," `http://arxiv.org/abs/1312.1085`, Dec. 2013.

Numerical illustrations

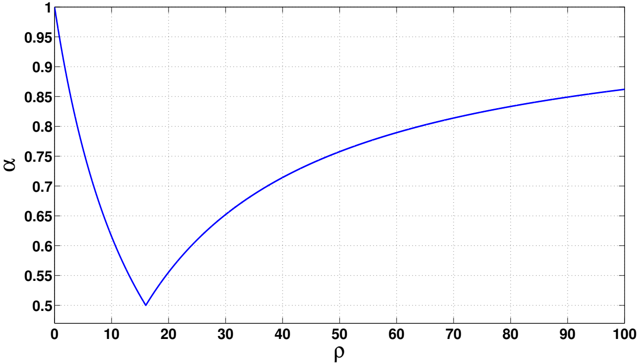
5-node graph separated as before, exponential functions



- Our rate is tight

Numerical illustrations

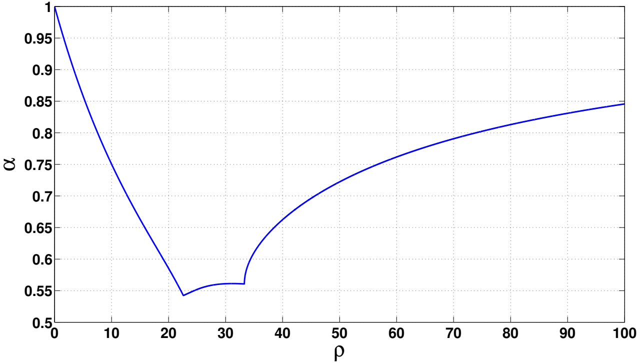
Centralized 5-node graph, quadratic functions with identic second order derivatives $\sigma_2 = 16$



- Optimal rate is $1/2$ obtained for $\rho = \sigma_2$.

Numerical illustrations

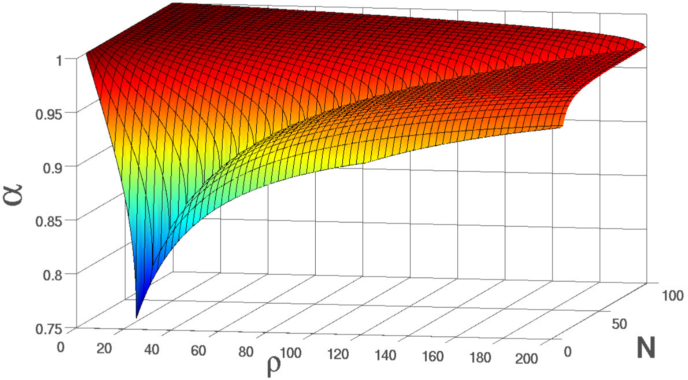
Centralized 5-node graph, quadratic functions with different second order derivatives $4, 9, 16, 25$ and 39



- Optimal rate is $> 1/2$.

Numerical illustrations

Ring graph, quadratic functions with identic second order derivatives $\sigma_2 = 16$



- The optimal parameter ρ grows linearly with N

Outline

Block-random updates and Asynchronous optimization

One can remark that updating the block ℓ in ζ is equivalent to update subset ℓ.

$$\zeta^{k+1} = \begin{bmatrix} \zeta_1^{k+1} \\ \vdots \\ \zeta_\ell^{k+1} \\ \vdots \\ \zeta_L^{k+1} \end{bmatrix} = \begin{bmatrix} \mathsf{J}_{|1}^{\mathrm{LM}}(\zeta^k) \\ \vdots \\ \mathsf{J}_{|\ell}^{\mathrm{LM}}(\zeta^k) \\ \vdots \\ \mathsf{J}_{|L}^{\mathrm{LM}}(\zeta^k) \end{bmatrix} = \mathsf{J}^{\mathrm{LM}}(\zeta^k)$$

Block-random updates and Asynchronous optimization

One can remark that updating the block ℓ in ζ is equivalent to update subset ℓ.
Let us try to update **only one block chosen at random**.

$$\zeta^{k+1} = \begin{bmatrix} \zeta_{|1}^{k+1} \\ \vdots \\ \zeta_{|\ell}^{k+1} \\ \vdots \\ \zeta_{|L}^{k+1} \end{bmatrix} = \begin{bmatrix} \zeta_{|1}^{k} \\ \vdots \\ \mathsf{J}_{|\ell}^{\mathrm{LM}}(\zeta^k) \\ \vdots \\ \zeta_{|L}^{k} \end{bmatrix} \triangleq \hat{\mathsf{J}}_{|\ell}^{\mathrm{LM}}(\zeta^k)$$

Block-random updates and Asynchronous optimization

One can remark that updating the block ℓ in ζ is equivalent to update subset ℓ. Let us try to update **only one block chosen at random**.

$$\zeta^{k+1} = \begin{bmatrix} \zeta_{|1}^{k+1} \\ \vdots \\ \zeta_{|\ell}^{k+1} \\ \vdots \\ \zeta_{|L}^{k+1} \end{bmatrix} = \begin{bmatrix} \zeta_{|1}^{k} \\ \vdots \\ \mathsf{J}_{|\ell}^{\mathrm{LM}}(\zeta^k) \\ \vdots \\ \zeta_{|L}^{k} \end{bmatrix} \triangleq \hat{\mathsf{J}}_{|\ell}^{\mathrm{LM}}(\zeta^k)$$

Problem: J^{LM} is FNE but $\hat{\mathsf{J}}_{|\ell}^{\mathrm{LM}}$ **is not**.

Block-random updates and Asynchronous optimization

One can remark that updating the block ℓ in ζ is equivalent to update subset ℓ. Let us try to update **only one block chosen at random**.

$$\zeta^{k+1} = \begin{bmatrix} \zeta^{k+1}_{|1} \\ \vdots \\ \zeta^{k+1}_{|\ell} \\ \vdots \\ \zeta^{k+1}_{|L} \end{bmatrix} = \begin{bmatrix} \zeta^{k}_{|1} \\ \vdots \\ \mathsf{J}^{\mathrm{LM}}_{|\ell}(\zeta^k) \\ \vdots \\ \zeta^{k}_{|L} \end{bmatrix} \triangleq \hat{\mathsf{J}}^{\mathrm{LM}}_{|\ell}(\zeta^k)$$

Problem: J^{LM} is FNE but $\hat{\mathsf{J}}^{\mathrm{LM}}_{|\ell}$ **is not.**

Theorem

Let J be a FNE operator and $\{\xi^k\}$ be an i.i.d. process valued in $\{1,\ldots,L\}$ such that $\mathbb{P}[\xi = \ell] > 0 \ \forall \ell$. Then, the iterations

$$\zeta^{k+1} = \hat{\mathsf{J}}_{|\xi^k}(\zeta^k)$$

produce a sequence converging almost surely to a fixed point of J if any.

F. Iutzeler, P. Bianchi, Ph. Ciblat, W. Hachem, "*Asynchronous Distributed Optimization using a Randomized Alternating Direction Method of Multipliers*," CDC, Dec. 2013.

Block-random updates and Asynchronous optimization

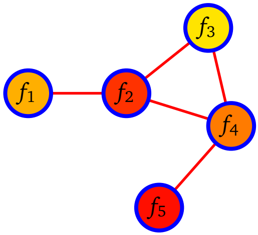
One can remark that updating the block ℓ in ζ is equivalent to update subset ℓ. Let us try to update **only one block chosen at random**.

$$\zeta^{k+1} = \begin{bmatrix} \zeta_{|1}^{k+1} \\ \vdots \\ \zeta_{|\ell}^{k+1} \\ \vdots \\ \zeta_{|L}^{k+1} \end{bmatrix} = \begin{bmatrix} \zeta_{|1}^{k} \\ \vdots \\ \mathsf{J}_{|\ell}^{\mathrm{LM}}(\zeta^k) \\ \vdots \\ \zeta_{|L}^{k} \end{bmatrix} \triangleq \hat{\mathsf{J}}_{|\ell}^{\mathrm{LM}}(\zeta^k)$$

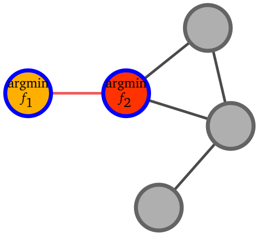
Consequences

Our block-random fixed point **converges almost surely**.
But which are the iterations of this algorithm applied to J^{LM}?

Asynchronous optimization with a randomized ADMM



Asynchronous optimization with a randomized ADMM



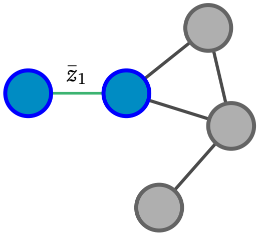
Asynchronous Optimization w/ ADMM

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 performs a proximal operation:

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

Asynchronous optimization with a randomized ADMM



ASYNCHRONOUS OPTIMIZATION W/ ADMM

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

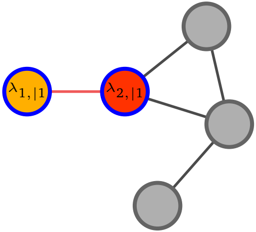
- Every sensor $i \in A_{\xi^{k+1}}$ of the block performs a proximal operation:

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

- The block computes its average:

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

Asynchronous optimization with a randomized ADMM



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 performs a proximal operation:

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

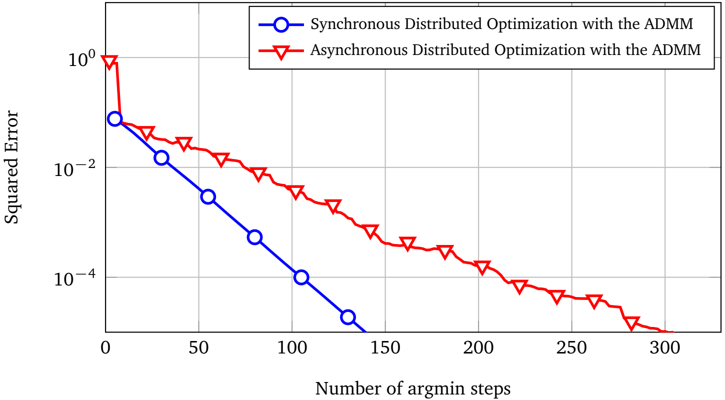
- The block computes its average:

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

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

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

Numerical illustrations



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

About the linear convergence of this asynchronous optimization algorithm

Theorem

Under the same assumptions as in the synchronous case and with i.i.d. choices of the blocks, there is $\alpha < 1$ such that for any initial value (z_0, λ_0) of the ADMM algorithm,

$$\limsup_{k\to\infty} \frac{1}{k} \log \|x_k - 1_N \otimes x_\star\| \leq \log \alpha \text{ with probability 1.}$$

To appear, May 2014.

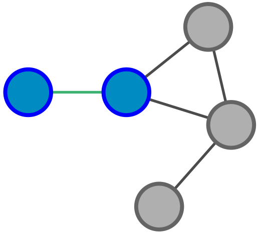
- The asynchronous fashion does not change the convergence mode
- The rate α is closely linked to the one in the synchronous case

Applications

- Our asynchronous setup enables us to deal with a large variety of situations

Applications

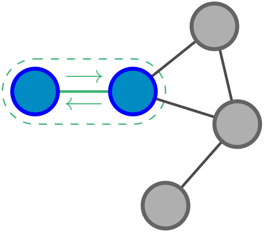
- Our asynchronous setup enables us to deal with a large variety of situations



- Distributed Optimization using local coordinators

Applications

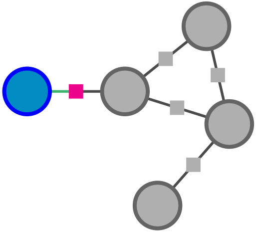
- Our asynchronous setup enables us to deal with a large variety of situations



- Distributed Optimization using local coordinators

Applications

- Our asynchronous setup enables us to deal with a large variety of situations

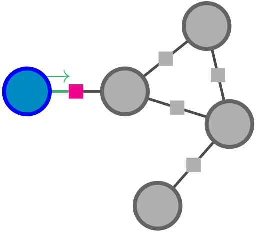


- Distributed Optimization using local coordinators
- Distributed Optimization with *One-Way* communications

By adding dummy nodes with constant functions
This also enables us to deal with network failures

Applications

- Our asynchronous setup enables us to deal with a large variety of situations

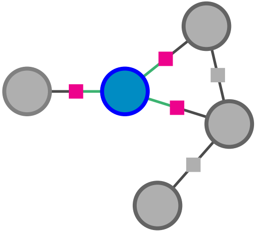


- Distributed Optimization using local coordinators
- Distributed Optimization with *One-Way* communications

By adding dummy nodes with constant functions
This also enables us to deal with network failures

Applications

- Our asynchronous setup enables us to deal with a large variety of situations

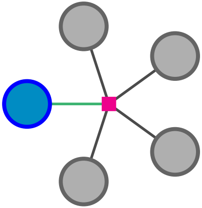


- Distributed Optimization using local coordinators
- Distributed Optimization with *One-Way* communications

By adding dummy nodes with constant functions
This also enables us to deal with network failures

Applications

- Our asynchronous setup enables us to deal with a large variety of situations

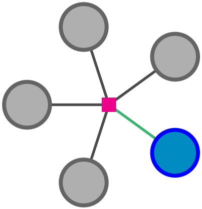


- Distributed Optimization using local coordinators
- Distributed Optimization with *One-Way* communications
- Mini-batch optimization/learning

The network is then just an artifact

Applications

- Our asynchronous setup enables us 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

Outline

General Conclusion & Perspectives

Conclusions

- Bringing randomness enables to deal with a large variety of situations
- Linear convergence and precise rates can be obtained

Perspectives

- Deriving techniques to obtain *good* values of parameter ρ
- Implementing asynchronous optimization for machine learning in a practical big data network