Optimisation Distribuée Asynchrone et Applications

Franck IUTZELER

Chaire Alcatel-Lucent sur la Radio Flexible – Supélec

Journées MAS – 2014

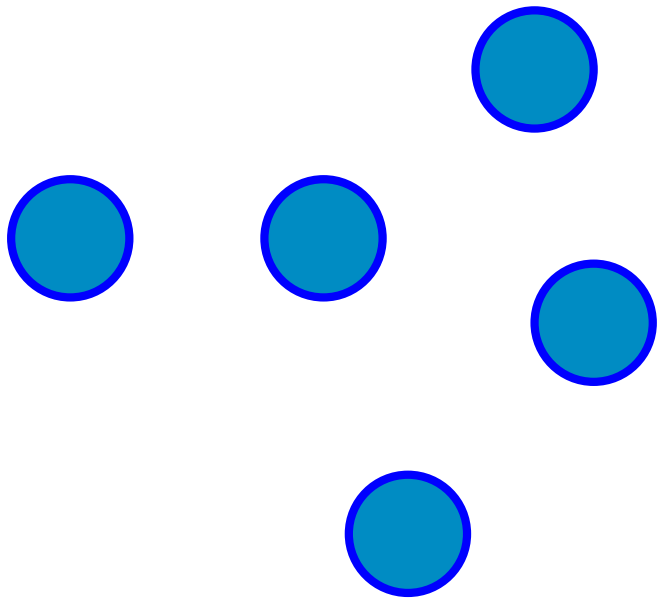
Supélec Flexible Radio
Alcatel Lucent Chair on Flexible Radio

Contexte Général de l'Optimisation Distribuée

- OPTIMISATION

  allocation de ressources,
  apprentissage

Contexte Général de l'Optimisation Distribuée



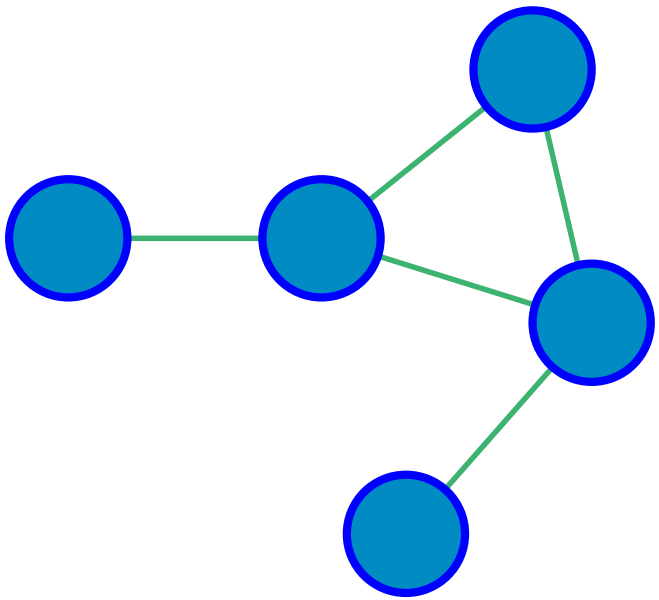
- OPTIMISATION

  allocation de ressources, apprentissage

- AGENTS

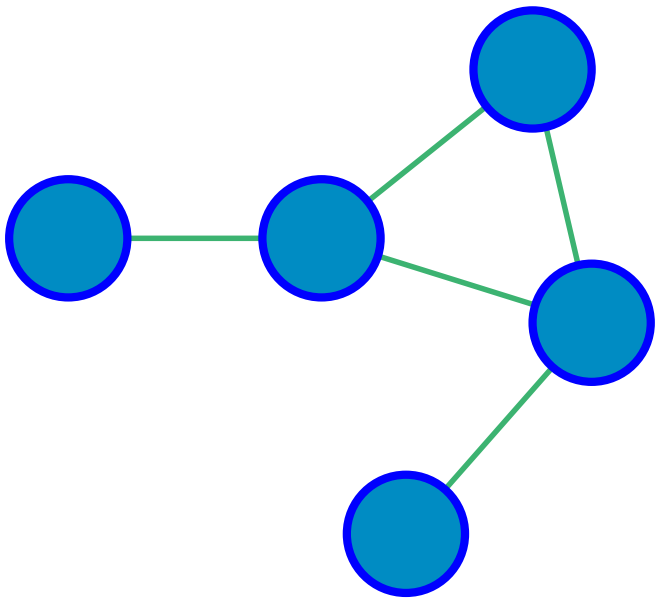
  données/fonctions locales, capacité de calcul

Contexte Général de l'Optimisation Distribuée



- OPTIMISATION
  allocation de ressources, apprentissage
- AGENTS
  données/fonctions locales, capacité de calcul
- RÉSEAU
  communications entre agents

Contexte Général de l'Optimisation Distribuée



- OPTIMISATION
  allocation de ressources, apprentissage
- AGENTS
  données/fonctions locales, capacité de calcul
- RÉSEAU
  communications entre agents

But

Il s'agit d'atteindre de manière **distribuée** un **consensus** autour d'une solution d'un **problème d'optimisation global** en utilisant uniquement des **données et des communications locales**.

Outline

Outline

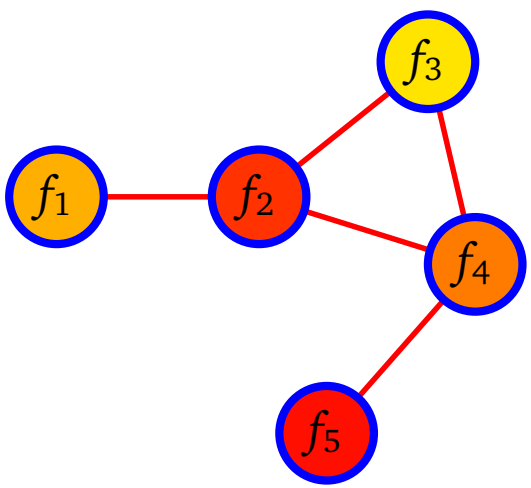
Introduction

Optimisation convexe distribuée

Trouver de manière distribuée un point minimisant la somme des fonctions locales des agents.

Problème :

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



- f_i est une fonction **convexe** et **locale** de l'agent i
- f n'est **connue nulle part**
- Les agents doivent trouver un minimiseur $x^\star$ de f
- réseau non-dirigé, connecté, V est l'ensemble des N agents

Reformulation de notre problème

Formulation distribuée du problème

Le problème de départ ne prend pas en compte :

- le fait qu'un agent ne connait que sa propre fonction ;
- les communications possibles (et impossibles) entre agents.

- Problème de départ

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

Reformulation de notre problème

Formulation distribuée du problème

Le problème de départ ne prend pas en compte :

- le fait qu'un agent ne connait que sa propre fonction ;
- les communications possibles (et impossibles) entre agents.

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

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

$$\text{sous contrainte} \quad x_1 = x_2 = \ldots = x_N$$

- Problème de départ
- Ajoutons le fait que les agents ne connaissent que leur propres fonctions

Reformulation de notre problème

Formulation distribuée du problème

Le problème de départ ne prend pas en compte :

- le fait qu'un agent ne connait que sa propre fonction ;
- les communications possibles (et impossibles) entre agents.

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

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

$$\text{sous contrainte} \quad x_1 = x_2 = \ldots = x_N$$

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

- Problème de départ
- Ajoutons le fait que les agents ne connaissent que leur propres fonctions
- Passons la contrainte dans la fonction à minimiser

où la *fonction indicatrice*

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

Reformulation de notre problème

Formulation distribuée du problème

Le problème de départ ne prend pas en compte :

- le fait qu'un agent ne connait que sa propre fonction ;
- les communications possibles (et impossibles) entre agents.

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

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

$$\text{sous contrainte} \quad x_1 = x_2 = \ldots = x_N$$

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

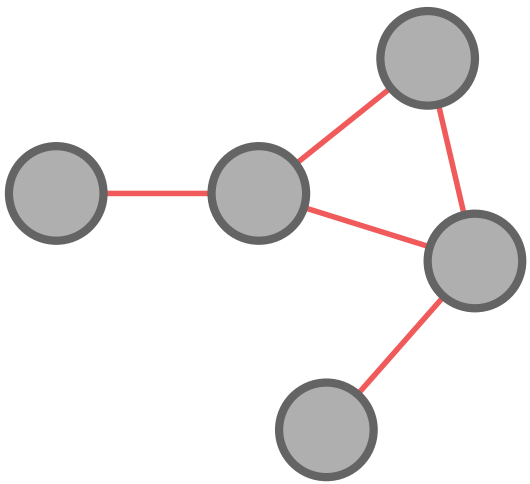
- Problème de départ
- Ajoutons le fait que les agents ne connaissent que leur propres fonctions
- Passons la contrainte dans la fonction à minimiser

où la *fonction indicatrice*

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

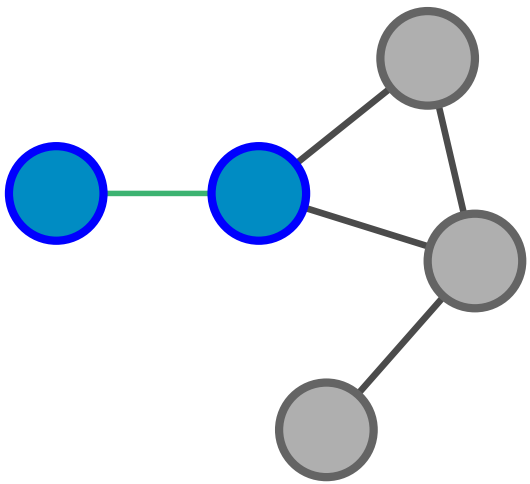
Reformulation de l'indicatrice du consensus

- L'idée est d'imposer le consensus localement sur *L sous-ensembles connectés se recouvrant* [Schizas2008]



Reformulation de l'indicatrice du consensus

- L'idée est d'imposer le consensus localement sur *L sous-ensembles connectés se recouvrant* [Schizas2008]

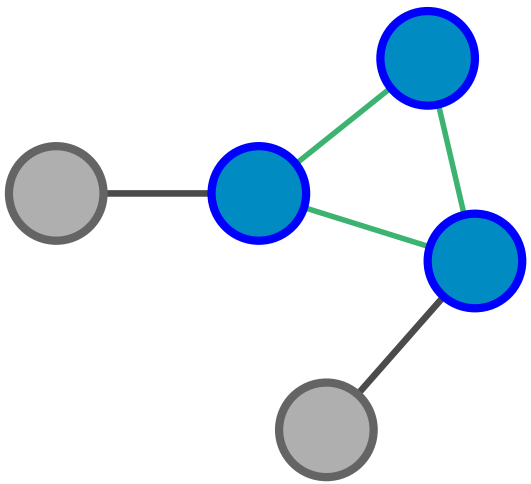


- $A_1 = \{1, 2\}$ $\quad \mathbf{M}_1 x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

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

Reformulation de l'indicatrice du consensus

- L'idée est d'imposer le consensus localement sur *L sous-ensembles connectés se recouvrant* [Schizas2008]

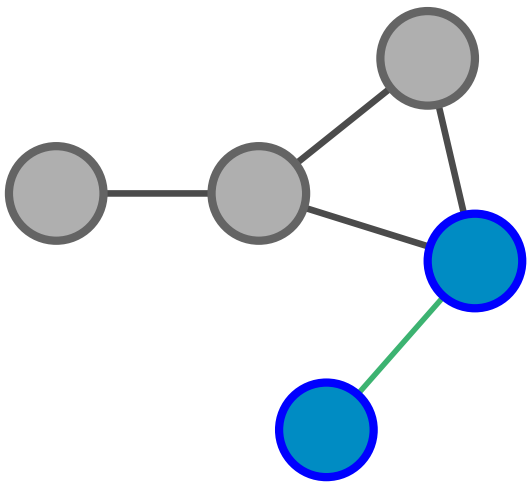


- $A_1 = \{1, 2\}$ $\quad \mathbf{M}_1 x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $A_2 = \{2, 3, 4\}$ $\quad \mathbf{M}_2 x = \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}$

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

Reformulation de l'indicatrice du consensus

- L'idée est d'imposer le consensus localement sur *L sous-ensembles connectés se recouvrant* [Schizas2008]

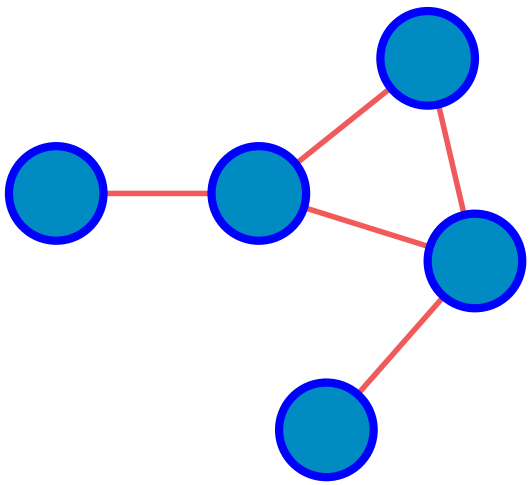


- $A_1 = \{1, 2\}$ $\quad \mathbf{M}_1 x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $A_2 = \{2, 3, 4\}$ $\quad \mathbf{M}_2 x = \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}$
- $A_3 = \{4, 5\}$ $\quad \mathbf{M}_3 x = \begin{bmatrix} x_4 \\ x_5 \end{bmatrix}$

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

Reformulation de l'indicatrice du consensus

- L'idée est d'imposer le consensus localement sur *L sous-ensembles connectés se recouvrant* [Schizas2008]

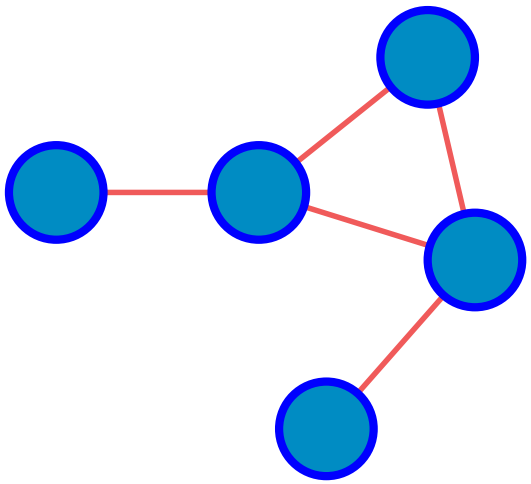


- $A_1 = \{1, 2\}$ $\quad \mathbf{M}_1 x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- $A_2 = \{2, 3, 4\}$ $\quad \mathbf{M}_2 x = \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}$
- $A_3 = \{4, 5\}$ $\quad \mathbf{M}_3 x = \begin{bmatrix} x_4 \\ x_5 \end{bmatrix}$

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

Reformulation de l'indicatrice du consensus

- L'idée est d'imposer le consensus localement sur *L sous-ensembles connectés se recouvrant* [Schizas2008]



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

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

Equivalent problem

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

Optimisation Distribuée par l'ADMM

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

- Problème : somme de deux fonctions
  - F : contient le problème d'origine
  - G et $\mathbf{M}$: contiennent les contraintes liées au réseau

Optimisation Distribuée par l'ADMM

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

- Problème : somme de deux fonctions
  - F : contient le problème d'origine
  - G et $\mathbf{M}$: contiennent les contraintes liées au réseau
- Résoudre directement ne permet pas d'obtenir un algorithme distribué

Optimisation Distribuée par l'ADMM

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

- Problème : somme de deux fonctions
  - F : contient le problème d'origine
  - G et $\mathbf{M}$: contiennent les contraintes liées au réseau
- Résoudre directement ne permet pas d'obtenir un algorithme distribué
- Mise en place d'algorithmes de *splitting*

Alternating Direction Method of Multipliers :

$$\begin{aligned}
x^{k+1} &= \operatorname*{argmin}_x \left\{ F(x) + \frac{\rho}{2} \left\| \mathbf{M}x - z^k + \lambda^k/\rho \right\|^2 \right\} \\
z^{k+1} &= \operatorname*{argmin}_z \left\{ G(z) + \frac{\rho}{2} \left\| \mathbf{M}x^{k+1} - z + \lambda^k/\rho \right\|^2 \right\} \\
\lambda^{k+1} &= \lambda^k + \rho \left(\mathbf{M}x^{k+1} - z^{k+1} \right)
\end{aligned}$$

Optimisation Distribuée par l'ADMM

L'ADMM appliqué à notre problème :

- $x^{k+1} = \underset{x \in \mathbb{R}^N}{\operatorname{argmin}} \left\{ F(x) + \frac{\rho}{2} \left\| \mathbf{M}x - z^k + \lambda^k/\rho \right\|_2^2 \right\}$
- $z^{k+1} = \underset{z \in \mathbb{R}^M}{\operatorname{argmin}} \left\{ G(z) + \frac{\rho}{2} \left\| \mathbf{M}x^{k+1} - z + \lambda^k/\rho \right\|_2^2 \right\}$
- $\lambda^{k+1} = \lambda^k + \rho \left(\mathbf{M}x^{k+1} - z^{k+1} \right)$

Optimisation Distribuée par l'ADMM

L'ADMM appliqué à notre problème :

- $$x^{k+1} = \underset{x \in \mathbb{R}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} f_i(x_i) + \frac{\rho}{2} \left\| \mathbf{M}x - z^k + \lambda^k/\rho \right\|_2^2 \right\}$$

- $$z^{k+1} = \underset{z \in \mathbb{R}^M}{\operatorname{argmin}} \left\{ \sum_{\ell=1}^{L} \iota_{\mathrm{Span}(\mathbb{1})}(z_{|\ell}) + \frac{\rho}{2} \left\| \mathbf{M}x^{k+1} - z + \lambda^k/\rho \right\|_2^2 \right\}$$

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

Optimisation Distribuée par l'ADMM

L'ADMM appliqué à notre problème :

- Pour tout agent i, $x_i^{k+1} = \underset{x}{\operatorname{argmin}} \left\{ f_i(x) + \frac{\rho}{2} \sum_{\ell \in \sigma_i} \left(x_i - z_{i,|\ell}^k + \frac{\lambda_{i,|\ell}^k}{\rho} \right)^2 \right\}$
- Pour tout bloc ℓ, $z_{|\ell}^{k+1} = \underset{z \in \mathbb{R}^{|A_\ell|}}{\operatorname{argmin}} \left\{ \iota_{\mathrm{Span}(\mathbb{1})}(z) + \frac{\rho}{2} \left\| \mathbf{M}_\ell x^{k+1} - z + \frac{\lambda_{|\ell}^k}{\rho} \right\|_2^2 \right\}$
- Pour tout agent i et bloc $\ell \in \sigma_i$, $\lambda_{i,|\ell}^{k+1} = \lambda_{i,|\ell}^k + \rho(x_i^{k+1} - \bar{z}_{|\ell}^{k+1})$

Chaque étape se déroule localement au niveau agent/bloc

· $z_{|\ell}$: bloc de taille $|A_\ell|$, correspond au sous-ensemble ℓ

· $\lambda_{i,|\ell}$: scalaire, correspond à l'agent i dans l'ensemble $\ell \in \sigma_i \triangleq \{l : i \in A_l\}$

Optimisation Distribuée par l'ADMM

L'ADMM appliqué à notre problème :

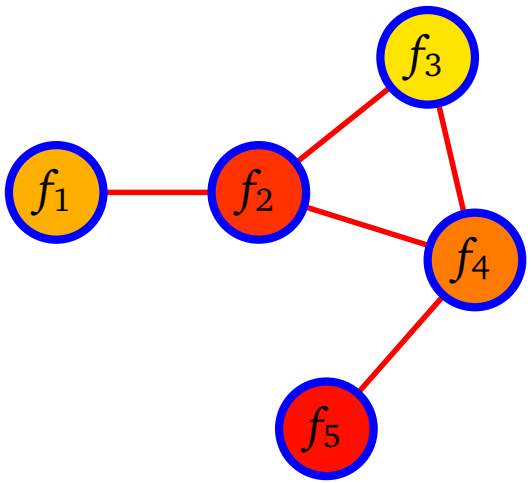
- Pour tout agent i, $x_i^{k+1} = \underset{x}{\operatorname{argmin}} \left\{ f_i(x) + \frac{\rho}{2} \sum_{\ell \in \sigma_i} \left(x_i - z_{i,|\ell}^k + \frac{\lambda_{i,|\ell}^k}{\rho} \right)^2 \right\}$
- Pour tout bloc ℓ, $z_{|\ell}^{k+1} = \frac{1}{|A_\ell|} \sum_{i \in A_\ell} x_i^{k+1} \mathbb{1} = \bar{z}_{|\ell}^{k+1} \mathbb{1}$
- Pour tout agent i et bloc $\ell \in \sigma_i$, $\lambda_{i,|\ell}^{k+1} = \lambda_{i,|\ell}^k + \rho(x_i^{k+1} - \bar{z}_{|\ell}^{k+1})$

Chaque étape se déroule localement au niveau agent/bloc

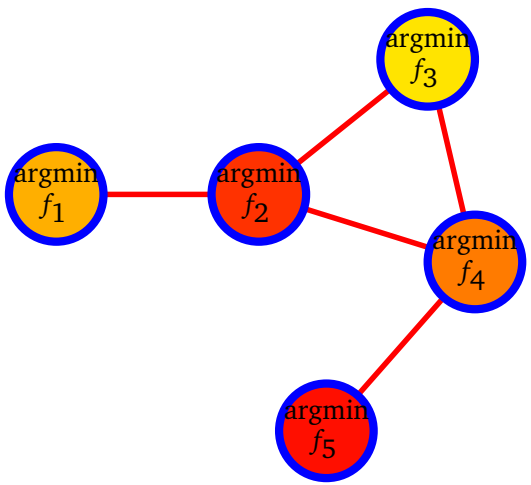
· $z_{|\ell}$: bloc de taille $|A_\ell|$, correspond au sous-ensemble ℓ

· $\lambda_{i,|\ell}$: scalaire, correspond à l'agent i dans l'ensemble $\ell \in \sigma_i \triangleq \{l : i \in A_l\}$

Optimisation Distribuée par l'ADMM



Optimisation Distribuée par l'ADMM



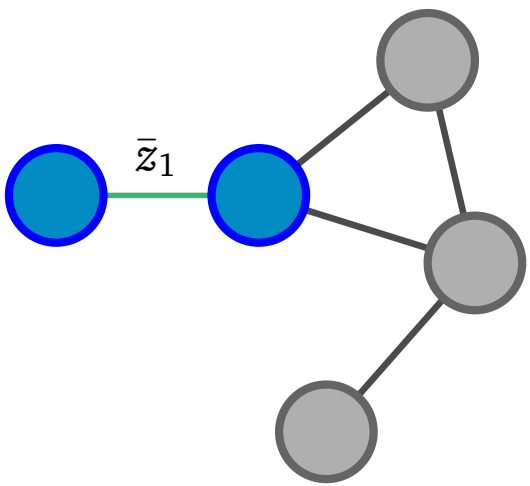
Optimisation Distribuée par l'ADMM

A l'itération k :

- Chaque agent i calcule :

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

Optimisation Distribuée par l'ADMM



Optimisation Distribuée par l'ADMM

A l'itération k :

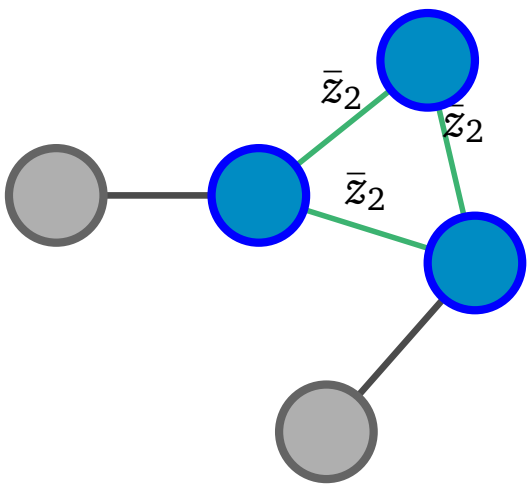
- Chaque agent i calcule :

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

- Chaque sous-ensemble A_ℓ échange :

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

Optimisation Distribuée par l'ADMM



Optimisation Distribuée par l'ADMM

A l'itération k :

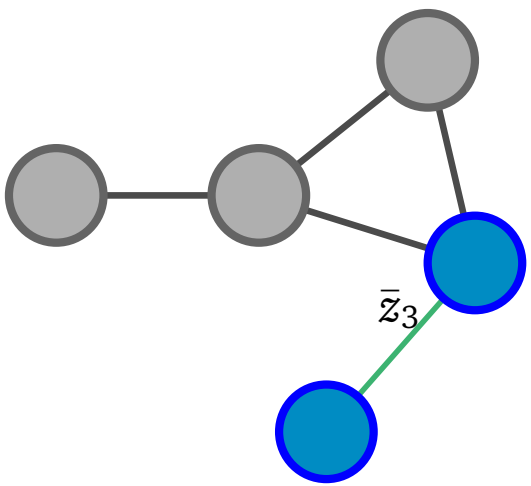
- Chaque agent i calcule :

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

- Chaque sous-ensemble A_ℓ échange :

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

Optimisation Distribuée par l'ADMM



Optimisation Distribuée par l'ADMM

A l'itération k :

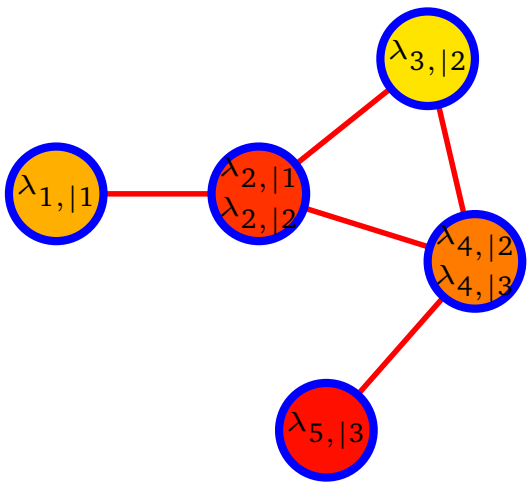
- Chaque agent i calcule :

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

- Chaque sous-ensemble A_ℓ échange :

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

Optimisation Distribuée par l'ADMM



Optimisation Distribuée par l'ADMM

A l'itération k :

- Chaque agent i calcule :

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

- Chaque sous-ensemble A_ℓ échange :

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

- Chaque agent i se met à jour :

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

- étape 1 – argmin : couteux en calcul
- étape 2 – échange : couteux en comm.

Outline

Introduction aux Opérateurs Monotones

Raisons

- Outils et Théorie mathématique unifiée pour la minimisation convexe
- Simplicité et Élégance des preuves
- Formulation intuitive qui permet la création d'algorithmes asynchrones

Introduction aux Opérateurs Monotones

Définition

*Un **opérateur** T sur $\mathbb{R}^N$ est une fonction multivaluée :*

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

*T est dit **monotone** si*

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

- Extension des fonctions croissantes aux fonctions multivaluées
- $(x,y) \in \mathsf{T}$ ssi $y \in \mathsf{T}(x)$

Exemple : sous-différentielle d'une fonction convexe h

∂h est un **opérateur monotone**.
On veut donc trouver les **zéros** de ∂h.

Résolvante d'un opérateur

Résolvante d'un opérateur T

*La **résolvante** de T est l'opérateur :*

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

- où I est l'opérateur identité $\mathsf{I}(x) = x$ et $(x, y) \in \mathsf{T}$ ssi $(y, x) \in \mathsf{T}^{-1}$.

Exemple : sous-différentielle d'une fonction convexe h

Trouver un **zéro** de $\partial h \Leftrightarrow$ Trouver un **point fixe** de $\mathsf{J}_{\partial h}$
Algorithme du point fixe :

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

Quelles sont les **propriétés de contraction** de $\mathsf{J}_{\partial h}$.

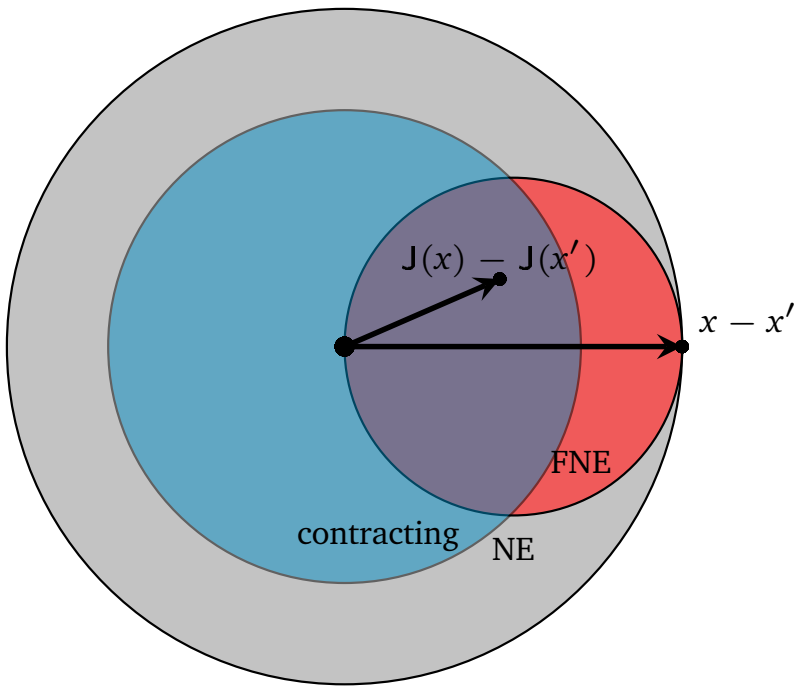
Propriétés de Contraction des Résolventes

Lemme

*J est la résolvante d'un opérateur montone ssi il est **Fermement Non-Expansif (FNE)** :*

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

- J n'est pas une contraction de Banach/Picard.



L'algorithme du Point Proximal

Lemme [Rockafellar1976]

Soit J *un opérateur FNE tel que* **fix** $\mathsf{J} \neq \emptyset$*, alors la séquence générée par*

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

converge vers une point de **fix** J.

Exemple : sous-différentielle d'une fonction convexe h

Itérer $\mathsf{J}_{\partial h}$ conduit à l'**Algorithme du Point Proximal** :

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

qui converge vers un zéro de ∂h.

Notre Problème

Problème Equivalent

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

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

Résolution de notre problème avec les opérateurs monotones

On veut trouver un **zéro** de $\mathsf{U} + \mathsf{V}$

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

en utilisant U et V **séparément** afin de découpler F et G.

L'opérateur de Lions-Mercier et l'ADMM

Lemme [Lions1979]

La résolvante J^{LM} *de l'opérateur de Lions-Mercier pour deux opérateurs* U *et* V *:*

- *est définie comme* $\mathsf{J}^{LM} \triangleq \mathsf{J}_{\rho\mathsf{U}} \circ (2\mathsf{J}_{\rho\mathsf{V}} - \mathsf{I}) + (\mathsf{I} - \mathsf{J}_{\rho\mathsf{V}})$ *;*
- *est FNE si* U *et* V *sont monotones ;*
- *a un point fixe si* $\mathbf{zer}(\mathsf{U} + \mathsf{V}) \neq \emptyset$.

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

Résolution de notre problème avec les opérateurs monotones

Itérer J^{LM} où $\mathsf{U} = -\mathbf{M}\partial F^*(-\mathbf{M}^{\mathrm{T}}\cdot)$ et $\mathsf{V} = \partial G^*$ conduit à l'**ADMM.**

L'opérateur de Lions-Mercier et l'ADMM

Lemme [Lions1979]

La résolvante J^{LM} *de l'opérateur de Lions-Mercier pour deux opérateurs* U *et* V *:*

- *est définie comme* $\mathsf{J}^{LM} \triangleq \mathsf{J}_{\rho\mathsf{U}} \circ (2\mathsf{J}_{\rho\mathsf{V}} - \mathsf{I}) + (\mathsf{I} - \mathsf{J}_{\rho\mathsf{V}})$ *;*
- *est FNE si* U *et* V *sont monotones ;*
- *a un point fixe si* $\mathbf{zer}(\mathsf{U} + \mathsf{V}) \neq \emptyset$.

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

Résolution de notre problème avec les opérateurs monotones

Itérer J^{LM} où $\mathsf{U} = -\mathbf{M}\partial F^*(-\mathbf{M}^{\mathrm{T}}\cdot)$ et $\mathsf{V} = \partial G^*$ conduit à l'**ADMM**.

Remarques :

- Les itérations de J^{LM} se font sur $\zeta = \lambda + \rho z \in \mathbb{R}^M$, les autres variables du problème (x par exemple) sont intermédiaires.
- Travailler sur $\zeta = \lambda + \rho z$ directement plutôt que sur x permet généralement de simplifier les preuves.

Convergence linéaire (exponentielle) de l'ADMM

Pour certaines fonctions F et G, le formalisme des opérateurs monotones permet l'étude de la convergence linéaire de l'ADMM (à travers ζ). C'est notamment le cas avec la fonction G définie précedemment.

Théorème

Supposons que les fonctions f_i sont convexes, propres, l.s.c., et qu'au point $x^\star$ où l'infimum du problème est atteint, les f_i sont deux fois différentiables et

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

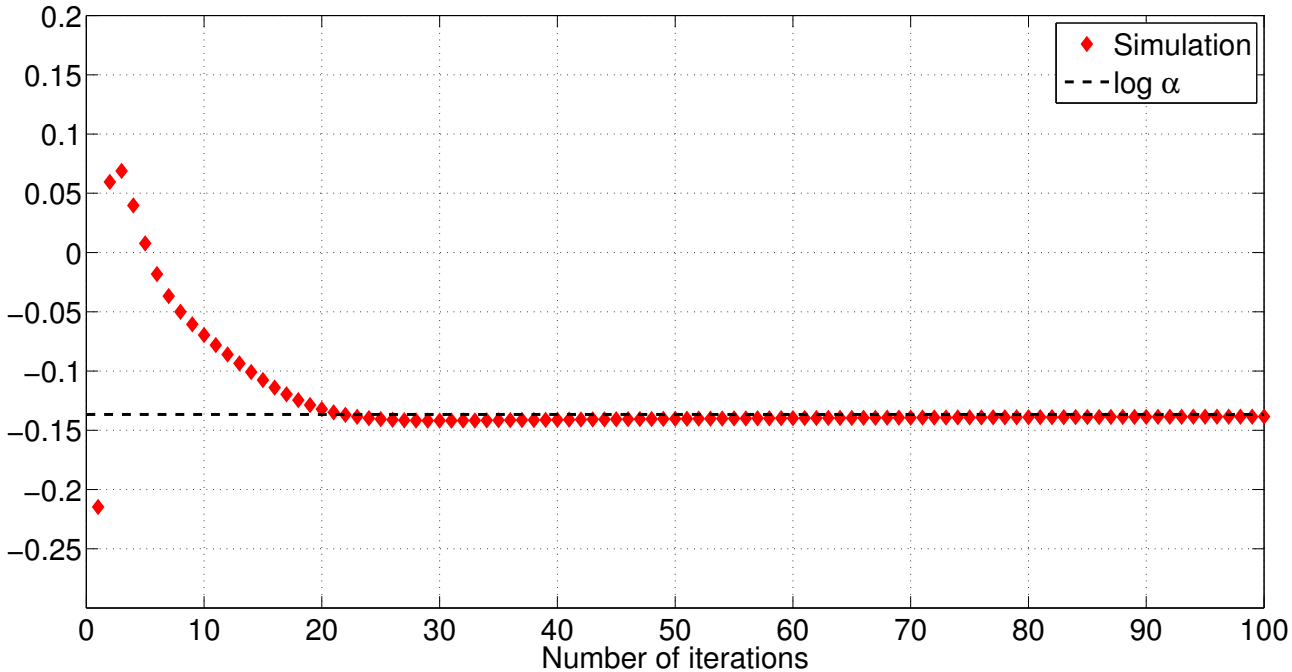
Alors, on peut identifier une matrice $\mathbf{R}$ dépendant des paramètres du problème telle que $\alpha = \mathbf{r}(\mathbf{R}) < 1$ et que pour tout (z_0, λ_0), l'ADMM distribué vu précédemment vérifie,

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

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.

Convergence linéaire (exponentielle) de l'ADMM

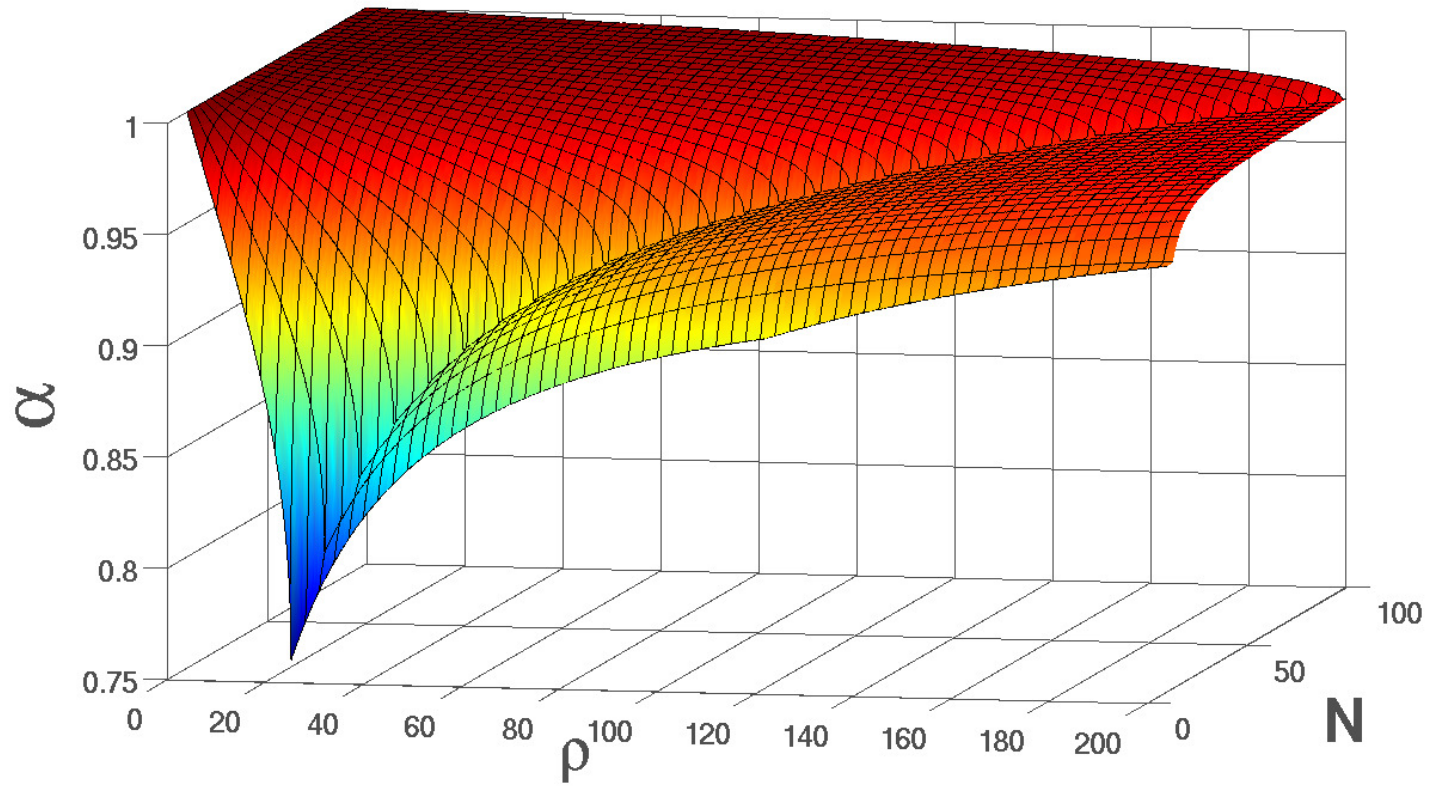
Graphe de 5 nœuds vu précédemment , f_i exponentielles



- Notre rate asymptotique est exact
- Il est atteint assez rapidement
- Il permet notamment d'avoir une idée de la pertinence des connexions

Convergence linéaire (exponentielle) de l'ADMM

Graphe en anneau, f_i quadratiques avec mêmes dérivées secondes $\sigma_2 = 16$



- Le choix du paramètre ρ est important (la forme pointue est caractéristique)
- Le ρ optimal dépend de σ_2, de N, de la forme du graphe, ...
- Dans le cas ci-dessus, le ρ optimal est $\frac{\sigma_2}{2\sin(2\pi/N)}$
- Dans le cas centralisé, le ρ optimal est σ_2

Outline

Mise à jour aléatoire des blocs et optimisation distribuée

On a remarqué que dans l'ADMM distribué, les ℓ-ièmes blocs de z et λ se calculaient de manière locale au ℓ-ième sous-ensemble d'agents.

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

Mise à jour aléatoire des blocs et optimisation distribuée

On a remarqué que dans l'ADMM distribué, les ℓ-ièmes blocs de z et λ se calculaient de manière locale au ℓ-ième sous-ensemble d'agents.
$\rightarrow$ Tentons de mettre à jour **uniquement un bloc choisi aléatoirement**.

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

Mise à jour aléatoire des blocs et optimisation distribuée

On a remarqué que dans l'ADMM distribué, les ℓ-ièmes blocs de z et λ se calculaient de manière locale au ℓ-ième sous-ensemble d'agents.
$\rightarrow$ Tentons de mettre à jour **uniquement un bloc choisi aléatoirement**.

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

Problème : J^{LM} est FNE mais $\hat{\mathsf{J}}_{|\ell}^{\mathrm{LM}}$ **ne l'est pas**.

Mise à jour aléatoire des blocs et optimisation distribuée

On a remarqué que dans l'ADMM distribué, les ℓ-ièmes blocs de z et λ se calculaient de manière locale au ℓ-ième sous-ensemble d'agents.
$\rightarrow$ Tentons de mettre à jour **uniquement un bloc choisi aléatoirement**.

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

Problème : J^{LM} est FNE mais $\hat{\mathsf{J}}_{|\ell}^{\mathrm{LM}}$ **ne l'est pas**.

Théorème

Soit J *un opérateur FNE et* $\{\xi^k\}$ *un processus i.i.d. à valeur dans* $\{1,\ldots,L\}$ *tel que* $\mathbb{P}[\xi = \ell] > 0 \ \ \forall \ell$. *Alors, les itérations*

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

convergent presque-surement vers un point fixe de J.

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

Mise à jour aléatoire des blocs et optimisation distribuée

On a remarqué que dans l'ADMM distribué, les ℓ-ièmes blocs de z et λ se calculaient de manière locale au ℓ-ième sous-ensemble d'agents.
$\rightarrow$ Tentons de mettre à jour **uniquement un bloc choisi aléatoirement.**

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

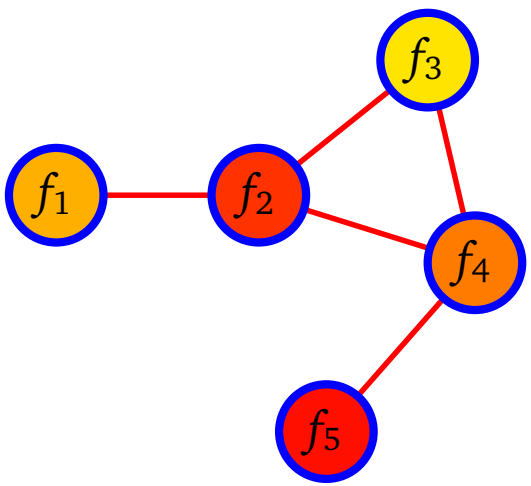
Conséquences

Notre algorithme du point fixe aléatoire converge presque-surement.
Quel sont les itérations de cet algorithme appliqué à J^{LM} ?

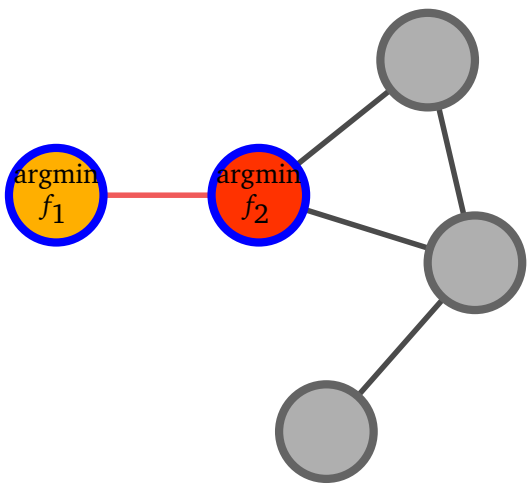
Extensions

dim. infinie, opérateurs α-moyennés, relaxation Krasnoselskii–Mann, ...
P.L. Combettes, J.-C. Pesquet, "*Stochastic Quasi-Fejér Block-Coordinate Fixed Point Iterations with Random Sweeping*," arXiv :1404.7536, 2014.
P. Bianchi, W. Hachem, F. Iutzeler, "*A Stochastic Coordinate Descent Primal-Dual Algorithm and Applications to Large-Scale Composite Optimization*," arXiv :1407.0898 , 2014.

Optimisation Distribuée Asynchrone par l'ADMM



Optimisation Distribuée Asynchrone par l'ADMM



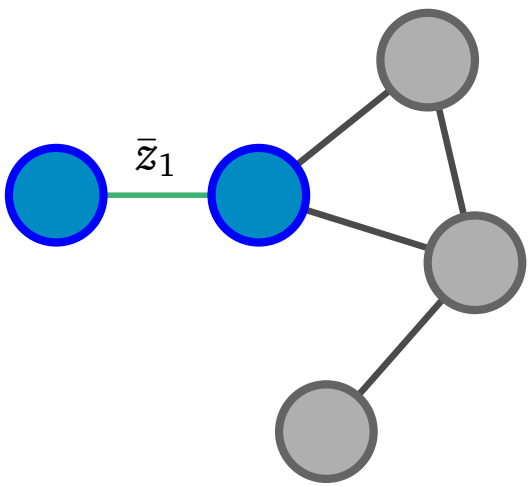
Optimisation Distribuée Asynchrone par l'ADMM

A chaque itération k, soit ξ^{k+1} l'indice du bloc choisi :

- Chaque agent $i \in A_{\xi^{k+1}}$ du bloc effectue un prox. :

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

Optimisation Distribuée Asynchrone par l'ADMM



Optimisation Distribuée Asynchrone par l'ADMM

A chaque itération k, soit ξ^{k+1} l'indice du bloc choisi :

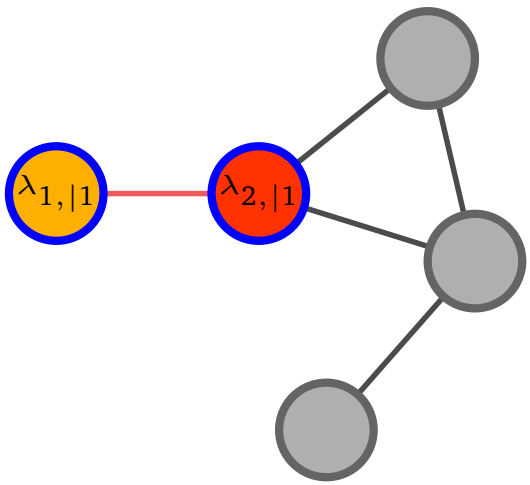
- Chaque agent $i \in A_{\xi^{k+1}}$ du bloc effectue un prox. :

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

- Le bloc calcule sa moyenne :

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

Optimisation Distribuée Asynchrone par l'ADMM



Optimisation Distribuée Asynchrone par l'ADMM

A chaque itération k, soit ξ^{k+1} l'indice du bloc choisi :

- Chaque agent $i \in A_{\xi^{k+1}}$ du bloc effectue un prox. :

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

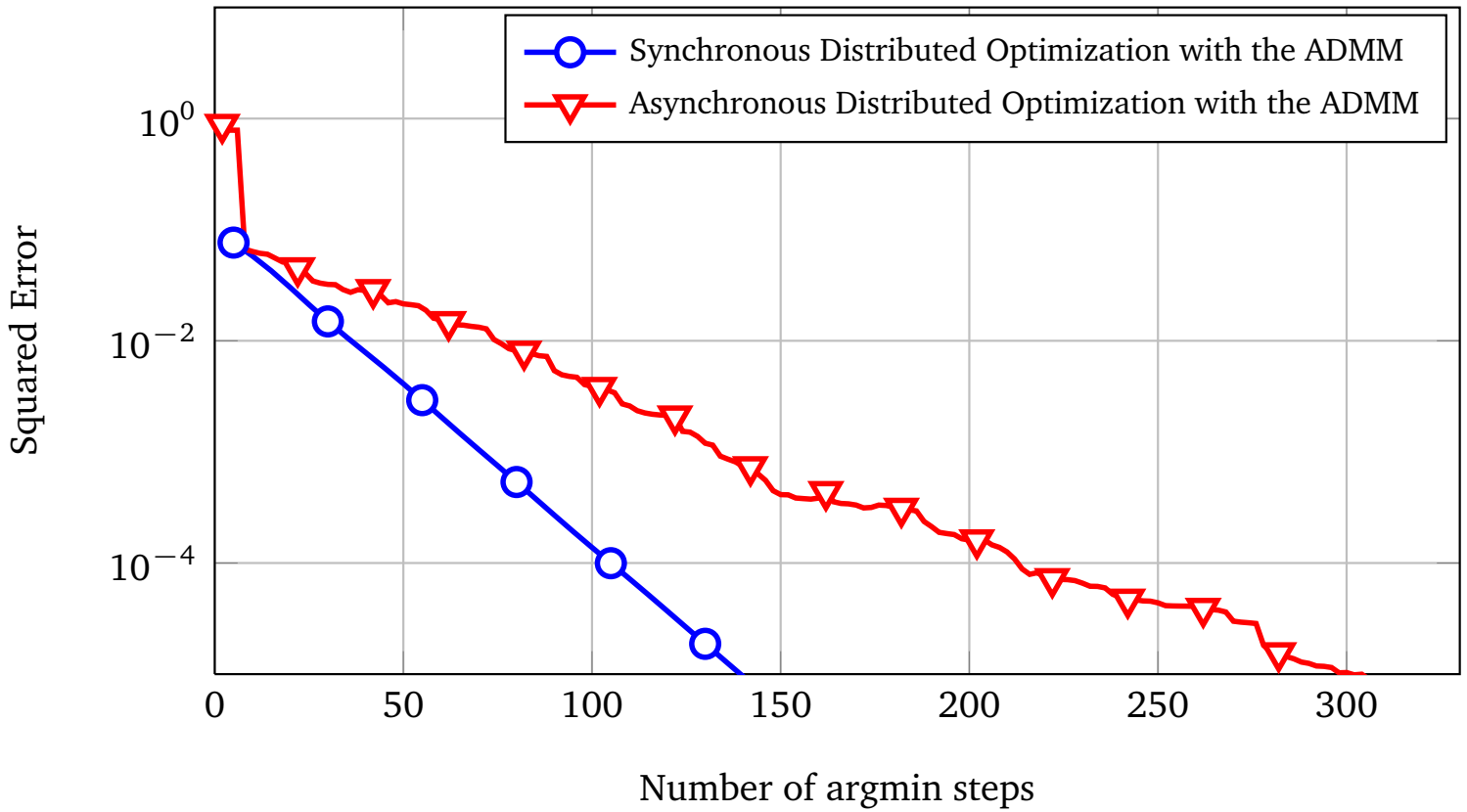
- Le bloc calcule sa moyenne :

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

- Chaque agent du bloc $i \in A_{\xi^{k+1}}$ se met à jour :

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

Illustration Numériques



- ADMM Synchrone : 1 itération = N argmin + L moyenne-bloc
- ADMM Asynchrone : 1 itération = $|A_{\xi^k}|$ argmin + 1 moyenne-bloc
- Convergence linéaire de l'EQM sous des mêmes hypothèses similaires au cas synchrone (to appear)

Outline

Bilan

Nous avons vu une méthode permettant de rendre un problème d'optimisation :

- Distribué

$$F(x)+G(\mathbf{M}x) \qquad \text{et splitting}$$

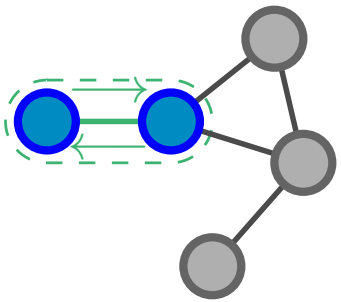
- Asynchrone

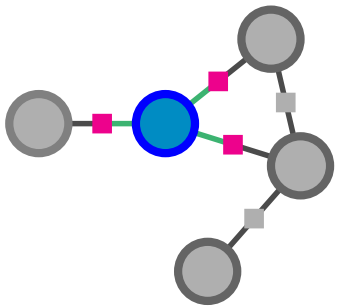
$$\zeta^{k+1} = \begin{bmatrix} \zeta_{|1}^{k+1} \\ \vdots \\ \zeta_{|\ell}^{k+1} \\ \vdots \\ \zeta_{|L}^{k+1} \end{bmatrix} = \begin{bmatrix} \zeta_{|1}^{k} \\ \vdots \\ \mathsf{J}_{|\ell}(\zeta^k) \\ \vdots \\ \zeta_{|L}^{k} \end{bmatrix} \triangleq \hat{\mathsf{J}}_{|\ell}(\zeta^k)$$

$\rightarrow$ Ces méthodes s'appliquent à des problèmes/algorithmes très divers.

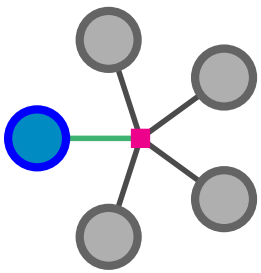
ADMM Distribué pour l'optimisation dans les réseaux

- ADMM Distribué ⇒ Optimisation dans les réseaux, comparaison de structures
  - Smart Grids
  - Réseaux de Capteurs
  - Calcul Haute Performance





- Asynchronisme ⇒ Erreurs
  - Pertes de Liens
  - Échec de calcul



Apprentissage en Grandes Dimensions

- Grande masse de donnés réparties (volontairement ou non) dans un réseau
- Problème : Fonction de coût = attache aux données (fortement convexe et lisse, adaptée aux gradients) + d'une régularisation (adaptée aux prox.) :

$$\min_{x \in \mathbb{R}^N} \underbrace{\sum_{i=1}^{N} l_i(x_i, \text{data}_i)}_{l(x)} + r(x) + G(\mathbf{M}x)$$

Apprentissage en Grandes Dimensions

- Grande masse de donnés réparties (volontairement ou non) dans un réseau
- Problème : Fonction de coût = attache aux données (fortement convexe et lisse, adaptée aux gradients) + d'une régularisation (adaptée aux prox.) :

$$\min_{x\in\mathbb{R}^N} \underbrace{\sum_{i=1}^{N} l_i(x_i, \text{data}_i)}_{l(x)} + r(x) + G(\mathbf{M}x)$$

- ADMM+ : Traite un pb. de type $\inf_x f(x) + g(x) + h(\mathbf{M}x)$ avec un gradient sur f (découle de l'algorithme Primal-Dual de Vu/Condat)

$$\begin{aligned}
z^{k+1} &= \operatorname*{argmin}_z \left\{ G(z) + \frac{1}{2\rho} \left\| \mathbf{M}x^k - z + \rho\lambda^k \right\|^2 \right\} \\
\lambda^{k+1} &= \lambda^k + \rho^{-1} \left(\mathbf{M}x^k - z^{k+1} \right) \\
u^{k+1} &= (1 - \tau\rho^{-1})\mathbf{M}x^k + \tau\rho^{-1} z^{k+1} \\
x^{k+1} &= \operatorname*{argmin}_x \left\{ r(x) + \langle \nabla l(x^k); x \rangle + \frac{1}{2\tau} \left\| \mathbf{M}x - u^{k+1} + \tau\lambda^{k+1} \right\|^2 \right\}
\end{aligned}$$

P. Bianchi, W. Hachem, F. Iutzeler, *"A Stochastic Coordinate Descent Primal-Dual Algorithm and Applications to Large-Scale Composite Optimization"*, MLSP, Sept. 2014 & http ://arxiv.org/abs/1407.0898.

Apprentissage en Grandes Dimensions

- Grande masse de donnés réparties (volontairement ou non) dans un réseau
- Problème : Fonction de coût = attache aux données (fortement convexe et lisse, adaptée aux gradients) + d'une régularisation (adaptée aux prox.) :

$$\min_{x \in \mathbb{R}^N} \underbrace{\sum_{i=1}^{N} l_i(x_i, \text{data}_i)}_{l(x)} + r(x) + G(\mathbf{M}x)$$

- ADMM+ :
  - Apprentissage Distribué
  - Les échanges sont de taille # de caractéristiques (evt. parcimonieuses)
  - L'asynchronisme permet des erreurs ou un apprentissage aléatoire
  - Communications unidirectionnelles possibles
  - Compliqué à écrire

Apprentissage en Grandes Dimensions par minibatch

- Réseau centralisé : $G(Mx) = \iota_{\mathrm{span}(\mathbb{1})}(x) \triangleq \iota(x)$
- Régularisation ℓ_1 pour simplifier

$$\min_{x \in \mathbb{R}^N} \sum_{i=1}^{N} l_i(x_i) + \mu \|x\|_1 + \iota(x)$$

Apprentissage en Grandes Dimensions par minibatch

- Réseau centralisé : $G(Mx) = \iota_{\text{span}(\mathbb{1})}(x) \triangleq \iota(x)$
- Régularisation ℓ_1 pour simplifier

$$\min_{x \in \mathbb{R}^N} \sum_{i=1}^{N} l_i(x_i) + \mu \|x\|_1 + \iota(x)$$

Exemple 1 :

- Gradient Proximal sur $\sum_{i=1}^{N} l_i$ et $\mu \| \cdot \|_1 + \iota$
  - Asynchronisme $\Rightarrow$ MISO [Mairal2013]

$$x^{k+1} = \text{SoftThres}\left\{ \frac{1}{N} \sum_{i=1}^{N} x^{\kappa_i} - \gamma \nabla l_i(x^{\kappa_i}); \frac{\mu}{\gamma N} \right\}$$

où $\kappa_i = k$ si i est choisi, inchangé sinon.
J. Mairal, *"Incremental Majoration-Minimization Optimization with Application to Large Scale Learning"*, NIPS , 2013 & http ://arxiv.org/abs/1402.4419.

Apprentissage en Grandes Dimensions par minibatch

- Réseau centralisé : $G(Mx) = \iota_{\mathrm{span}(\mathbb{1})}(x) \triangleq \iota(x)$
- Régularisation ℓ_1 pour simplifier

$$\min_{x \in \mathbb{R}^N} \sum_{i=1}^{N} l_i(x_i) + \mu \|x\|_1 + \iota(x)$$

Exemple 2 :

- ADMM+ sur $\sum_{i=1}^{N} l_i$, $\mu\|\cdot\|_1$, et ι
  - Asynchronisme $\Rightarrow$ SMPD

$$\bar{x}^k = \frac{1}{N}\sum_{i=1}^{N} x_i^k, \quad \bar{\lambda}^k = \frac{1}{N}\sum_{i=1}^{N} \lambda_i^k$$

Pour le batch n choisi :

$$\lambda_n^{k+1} = \lambda_n^k - \bar{\lambda}^k + \rho^{-1}(x_n^k - \bar{x}^k)$$

$$x^{k+1} = \mathrm{SoftThres}\left\{(1 - 2\tau\rho^{-1})x_n^k - \tau\nabla l_n(x_n^k) - \tau\lambda_n^k + 2\tau(\rho^{-1}\bar{x}^k + \bar{\lambda}^k);\ \frac{\tau\mu}{N}\right\}$$

Pour les autres, les variables restent identiques

P. Bianchi, W. Hachem, F. Iutzeler, *"A Stochastic Coordinate Descent Primal-Dual Algorithm and Applications to Large-Scale Composite Optimization"*, MLSP, Sept. 2014 & http ://arxiv.org/abs/1407.0898.

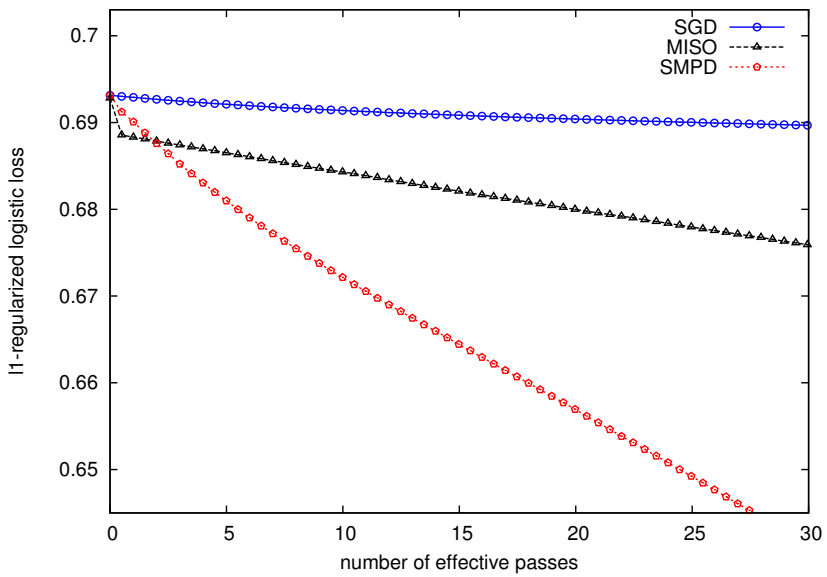
Apprentissage en Grandes Dimensions par minibatch

- Réseau centralisé : $G(Mx) = \iota_{\mathrm{span}(\mathbb{1})}(x) \triangleq \iota(x)$
- Régularisation ℓ_1 pour simplifier

$$\min_{x \in \mathbb{R}^N} \sum_{i=1}^{N} l_i(x_i) + \mu \|x\|_1 + \iota(x)$$

Comparaison :

- Jeu de données `covtype` : m = 581012 observations et p = 54 caractéristiques
- Traité en 581 batchs de 1000 observations



Outline

Conclusion & Perspectives

Conclusions

- L'ajout d'une fonction et un bon choix d'algorithme permet de distribuer un algorithme d'optimisation
- L'ajout de l'aléa permet de traiter une grande variété de situations (erreurs, minibatch, ...)

Perspectives

- Comment trouver de *bonnes* valeurs des paramètres (ρ, τ)
- Fonctions stochastiques