Exploiting Agent Symmetries for Automatic Performance Analysis of Distributed Optimization Methods

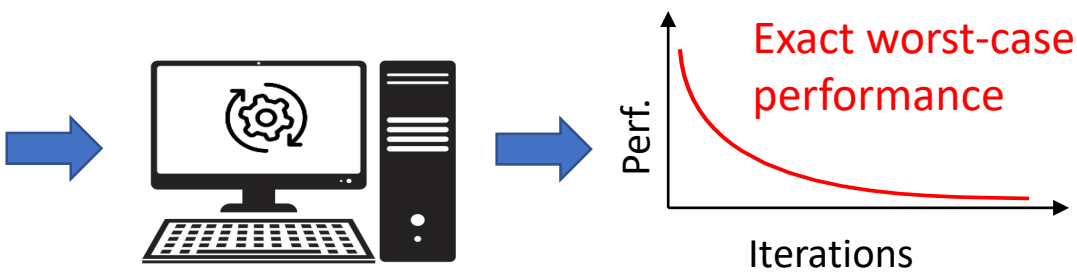
Sébastien Colla, Julien Hendrickx

Mathematical Engineering Department, UCLouvain (Belgium)

EUROpt 2023

```
function MyDecentralizedAlgo()
    N = 10; % number of agents
    x0 = init(N); % Initial point
    x = x0;

    for i=1:niter
        % any local computations
        % and local communications
        x = update(x,N);
    end
end
```

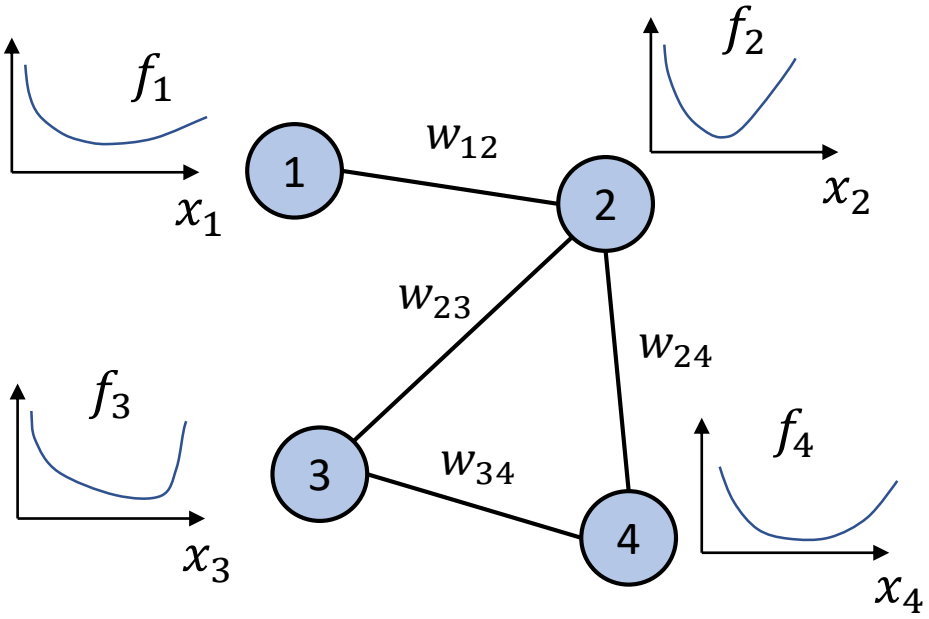


Distributed Optimization

$$\min_x f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$$

$$\min_{x_1,\dots,x_N} F_s(x_1,\dots,x_N) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i)$$

$$\text{s.t. } x_i = x_j \quad \forall\,(i,j) \text{ neighbors}$$



Decentralization

- Local function: f_i
- Local copy of x: x_i

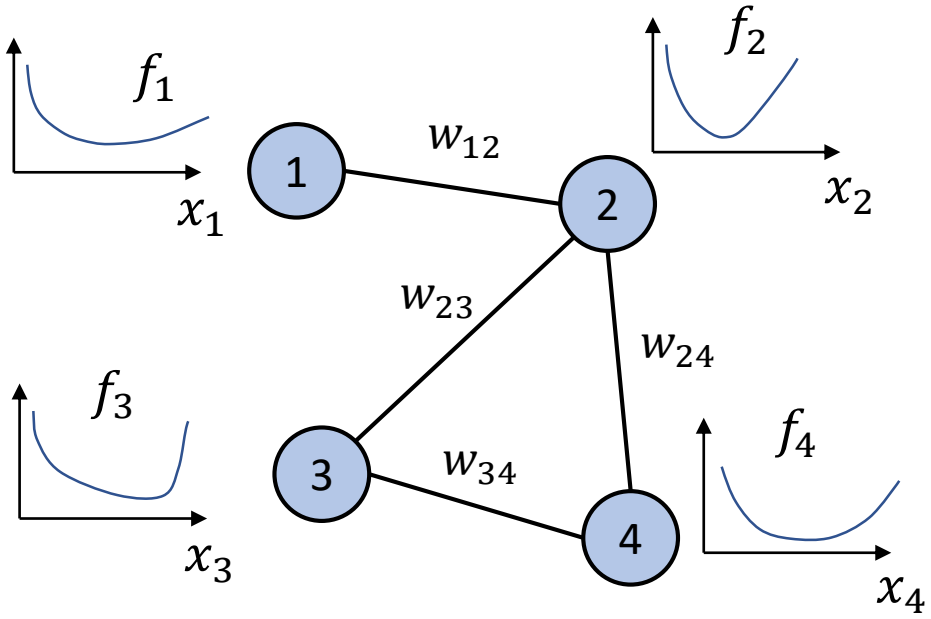
Iterative algorithm

- Local computations
- Local communications (W)
  so that $x_i = x_j$ (eventually)

Distributed Gradient Descent (DGD)

$$\min_{x_1,\dots,x_N} F_s(x_1,\dots,x_N) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i)$$

$$\text{s.t. } x_i = x_j \quad \forall\,(i,j) \text{ neighbors}$$



Decentralization

- Local function: f_i
- Local copy of x: x_i

Distributed Gradient Descent (DGD)

For each iteration k

$$y_i^k = \sum_j w_{ij}\, x_j^k \qquad \textit{Consensus step}$$

$$x_i^{k+1} = y_i^k - \alpha \nabla f_i(x_i^k) \qquad \textit{Local gradient step}$$

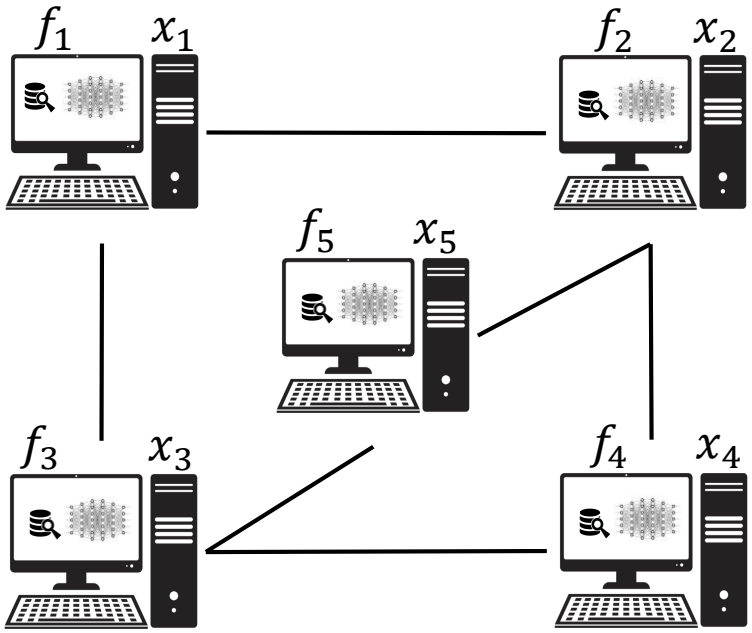
Motivations: Decentralized Machine Learning

Notations

- Model parameters x
- Data set $\{d \in \mathcal{D}\}$

Model training

$$\min_x \sum_{d \in D} \text{Error}(x, d)$$



Decentralization

- Part of the data $\mathcal{D}_i$
- Local function

$$f_i(x) = \sum_{d \in \mathcal{D}_i} \text{Error}(x, d)$$

- Local copy of x

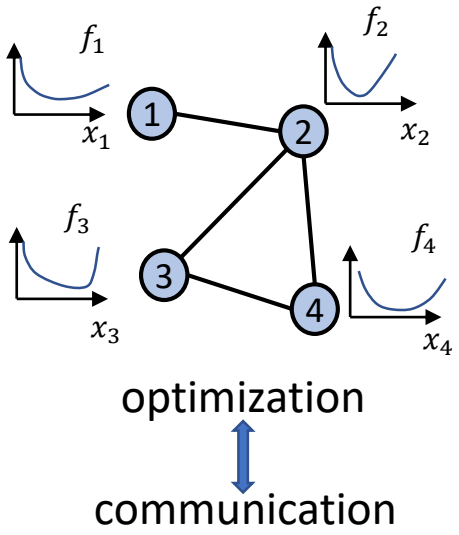
➡ **Motivations** Big data – Privacy – Speed Up

Decentralized Optimization

Many challenges for better methods

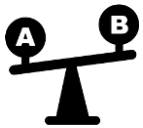
BUT

Analysis highly complex



- Performance bounds: complex and **conservative**

- Difficult algorithms comparisons

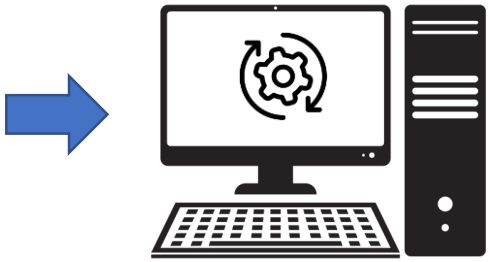


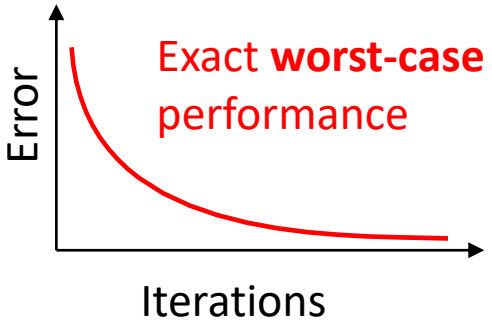
- Difficult parameters tuning

Objective

```
function MyDecentralizedAlgo()
    N = 10;         % number of agents
    x0 = init(N);  % initial points
    x = x0;

    for k=1:niter
        % any local computations
        % any local communications
        x = update(x,N);
    end
end
```





Impact for decentralized optimization

- Access to **accurate performance** of methods
- Easier **comparison and tuning** of algorithms
- **Rapid exploration** of new algorithms.

Overview

- Performance Estimation Problem (PEP) for distributed optimization
- Agents independent performance using symmetric solutions
- Subsets of interchangeable agents and the performance of the worst agent.

Performance Estimation Problem (PEP)

idea

Worst-cases are solutions to optimization problems

[Drori and Teboulle 2014]

Typical decentralized optimization result

Provided that

- The local functions are in a given class $\mathrm{f_i} \in \mathcal{F}$
- The network (matrix) is in $\mathcal{W}$
- The initial iterates are in some set $\mathcal{X}_0$

Then after K iterations, some quality measure P of the estimate solutions *always satisfy* (…)

PEP : **compute** a tight worst-case bound

Performance Estimation Problem (PEP)

idea

Worst-cases are solutions to optimization problems

[Drori and Teboulle 2014]

Typical decentralized optimization result

Provided that

- The local functions are in a given class $\mathrm{f_i} \in \mathcal{F}$
- The network (matrix) is in $\mathcal{W}$
- The initial iterates are in some set $\mathcal{X}_0$

Then after K iterations, some quality measure P of the estimate solutions *always satisfy* (…)

PEP : **compute** a tight worst-case bound

$$\max_{f_i,\, \mathrm{W},\, x_i^k,\, y_i^k} \quad P(f_i, x_i^0, \dots, x_i^K)$$

s.t.

$$f_\mathrm{i} \in \mathcal{F} \qquad \text{for } i = 1 \dots N$$
$$\mathrm{W} \in \mathcal{W}$$
$$x_i^0 \in \mathcal{X}_0$$
$$y_i^k = W x_i^k$$
$$x_i^0, \dots, x_i^K$$
$$y_i^0, \dots, y_i^{K-1}$$

satisfy the algorithm

Performance Estimation Problem (PEP)

idea

Worst-cases are solutions to optimization problems

[Drori and Teboulle 2014]

Typical decentralized optimization result

Provided that

- The local functions are in a given class $\mathrm{f_i} \in \mathcal{F}$
- The network (matrix) is in $\mathcal{W}$
- The initial iterates are in some set $\mathcal{X}_0$

Then after K iterations, some quality measure P of the estimate solutions *always satisfy* (…)

PEP : **compute** a tight worst-case bound

$$\max_{f_i,\, \mathrm{W},\, x_i^k,\, y_i^k} \quad P(f_i, x_i^0, \dots, x_i^K)$$

s.t.

$$f_i \in \mathcal{F} \qquad \text{for } i = 1 \dots N$$
$$\mathrm{W} \in \mathcal{W}$$
$$x_i^0 \in \mathcal{X}_0$$
$$y_i^k = W x_i^k$$
$$x_i^0, \dots, x_i^K$$
$$y_i^0, \dots, y_i^{K-1}$$

satisfy the algorithm

Problems:

- **Infinite dimensional sets**
- **Highly nonlinear**

Finite dimensional reformulation

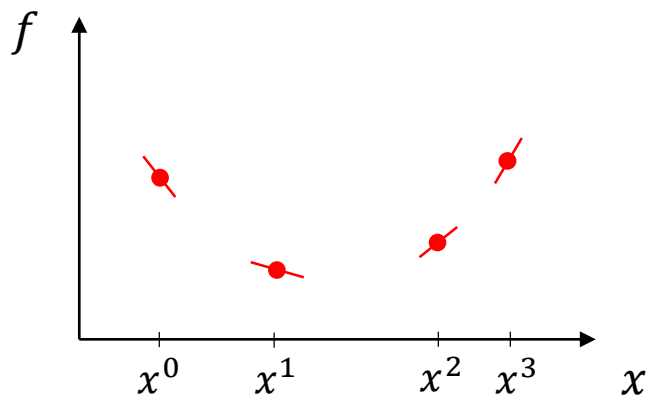
Finite dimension $y_i^k, x_i^k, g_i^k, f_i^k$

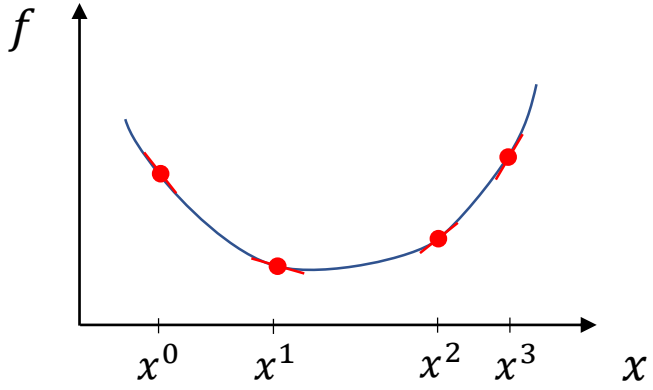
Class of functions

Interpolation conditions on $\{x_i^k, g_i^k, f_i^k\}_{k=1\dots K}$

$\Leftrightarrow$

There is $f_i \in \mathcal{F}$ s.t. $f_i^k = f_i(x_i^k)$ and $g_i^k = \nabla f_i(x_i^k)$





Interpolation conditions for many classical function classes [Taylor et al. 2017]

e.g. L-smooth and μ-strongly convex, convex bounded gradient,…

Finite dimensional reformulation

Finite dimension $y_i^k, x_i^k, g_i^k, f_i^k$

Class of functions

Interpolation conditions on $\{x_i^k, g_i^k, f_i^k\}_{k=0\dots K}$ $\Leftrightarrow$ There is $f_i \in \mathcal{F}$ s.t. $f_i^k = f_i(x_i^k)$ and $g_i^k = \nabla f_i(x_i^k)$

Class of network matrices

Interpolation conditions on $\{x_i^k, y_i^k\}_{k=0\dots K-1}$ $\Leftrightarrow$ There is $W \in \mathcal{W}$ s.t. $y_i^k = W x_i^k$

Common class

$\mathcal{W}(\lambda^-, \lambda^+)$: symmetric, doubly stochastic, and $\lambda(W) \in [\lambda^-, \lambda^+]$

➡ necessary conditions available (relaxed PEP)

[Colla 2023]

PEP for Distributed Optimization

$$\max_{y_i^k, x_i^k, g_i^k, f_i^k} \quad P(y_i^k, x_i^k, g_i^k, f_i^k)$$

s.t.

$$\left.\begin{array}{c} f_i \in \mathcal{F} \\ W \in \mathcal{W} \\ y_i^k = W x_i^k \end{array}\right\}$$ Interpolation conditions

$$x_i^0 \in \mathcal{X}_0$$

$$x_i^0, \dots, x_i^K$$

$$y_i^0, \dots, y_i^{K-1}$$ satisfy the algorithm

PEP for Distributed Optimization

$$\max_{y_i^k, x_i^k, g_i^k, f_i^k} \quad P(y_i^k, x_i^k, g_i^k, f_i^k)$$

s.t.
$$\left.\begin{aligned} f_i &\in \mathcal{F} \\ W &\in \mathcal{W} \\ y_i^k &= W x_i^k \end{aligned}\right\} \text{ Interpolation conditions}$$
$$x_i^0 \in \mathcal{X}_0$$
$$x_i^0, \dots, x_i^K$$
$$y_i^0, \dots, y_i^{K-1} \quad \text{satisfy the algorithm}$$

PEP constraints may be quadratic and non-convex in y_i^k, x_i^k, or g_i^k

$$e.g. \qquad \frac{1}{N}\sum_{i=1}^{N} \left\| x_i^0 - x^* \right\|^2 \leq 1$$

↳ Linear in the scalar product between the x_i^0 and x^*

Change of variable: Gram Matrix

Variables Function values and Gram Matrix of scalar products

$F = [f_1 \dots f_N]$ where $f_i = \left[f_i^0 \dots f_i^K\right]$

Gram Matrix

$G = P^T P$ $P = [P_1 \dots P_N]$ $P_i = \left[y_i^0 \dots \; y_i^{K-1} \; x_i^0 \dots \; x_i^K \; g_i^0 \dots g_i^K\right]$

$$G = P^T P = \begin{bmatrix} P_1^T P_1 & \cdots & P_1^T P_N \\ \vdots & \ddots & \vdots \\ P_N^T P_1 & \cdots & P_1^T P_1 \end{bmatrix} = \begin{bmatrix} G_{11} & \cdots & G_{1N} \\ \vdots & \ddots & \vdots \\ G_{N1} & \cdots & G_{11} \end{bmatrix}$$

Change of variable: Gram Matrix

Variables Function values and Gram Matrix of scalar products

$F = [f_1 \dots f_N]$ where $f_i = \left[f_i^0 \dots f_i^K\right]$

Gram Matrix

$G = P^T P$ $\quad P = [P_1 \dots P_N]$ $\quad P_i = \left[y_i^0 \dots \; y_i^{K-1} \; x_i^0 \dots \; x_i^K \; g_i^0 \dots g_i^K\right]$

$$G = P^T P = \begin{bmatrix} P_1^T P_1 & \cdots & P_1^T P_N \\ \vdots & \ddots & \vdots \\ P_N^T P_1 & \cdots & P_1^T P_1 \end{bmatrix} = \begin{bmatrix} G_{11} & \cdots & G_{1N} \\ \vdots & \ddots & \vdots \\ G_{N1} & \cdots & G_{11} \end{bmatrix}$$

PEP

$$\max_{F,\,G} \quad \mathrm{perf}(\mathrm{F}, \mathrm{G})$$

s.t. $G \succcurlyeq 0$

Interpolation
Initial
Algorithm

constraints linear in G and F

Change of variable: Gram Matrix

Variables Function values and Gram Matrix of scalar products

$F = [f_1 \dots f_N]$ where $f_i = [f_i^0 \dots f_i^K]$

Gram Matrix

$G = P^T P$ $\quad P = [P_1 \dots P_N]$ $\quad P_i = [y_i^0 \dots y_i^{K-1}\ x_i^0 \dots x_i^K\ g_i^0 \dots g_i^K]$

$$G = P^T P = \begin{bmatrix} P_1^T P_1 & \cdots & P_1^T P_N \\ \vdots & \ddots & \vdots \\ P_N^T P_1 & \cdots & P_1^T P_1 \end{bmatrix} = \begin{bmatrix} G_{11} & \cdots & G_{1N} \\ \vdots & \ddots & \vdots \\ G_{N1} & \cdots & G_{11} \end{bmatrix}$$

PEP

$$\max_{F,\,G} \quad \text{perf}(F, G)$$

s.t. $G \succcurlyeq 0$

Interpolation
Initial
Algorithm

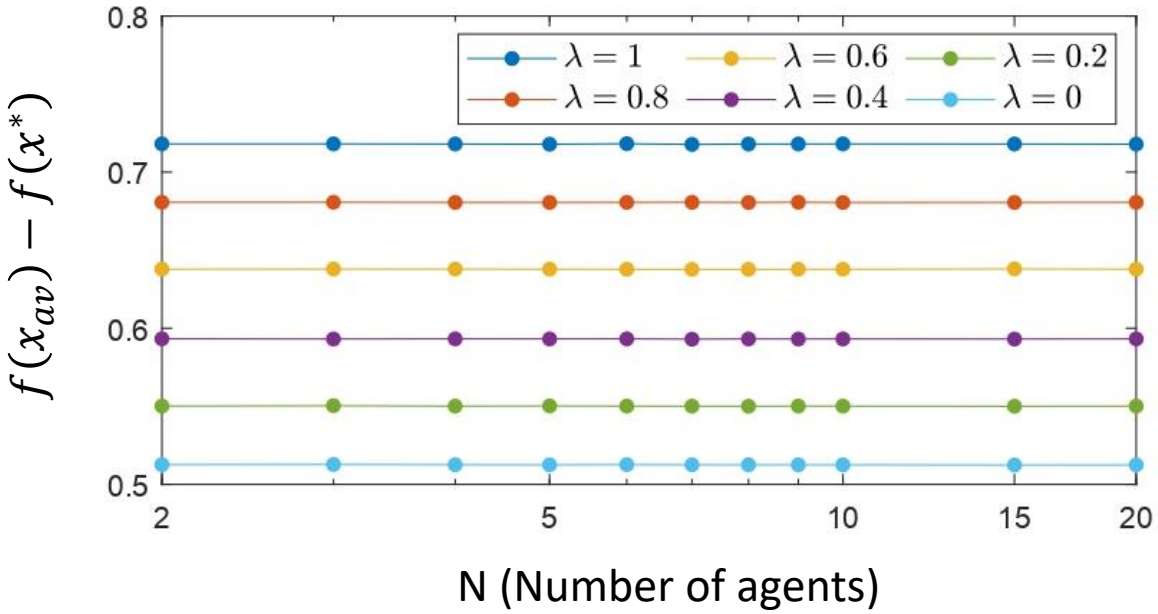
constraints linear in G and F

Note

- $y_i^k, x_i^k, g_i^k, f_i^k$ can be recovered if $G \succcurlyeq 0$
- dimension independent formulation ($d = \text{rank}\, G$)

SDP ➡ *Efficient resolution*

Agent Independent Performance



DGD – $\mathrm{K} = 5$ iterations

$\lambda(\mathrm{W}) \in [-\lambda, \lambda]$

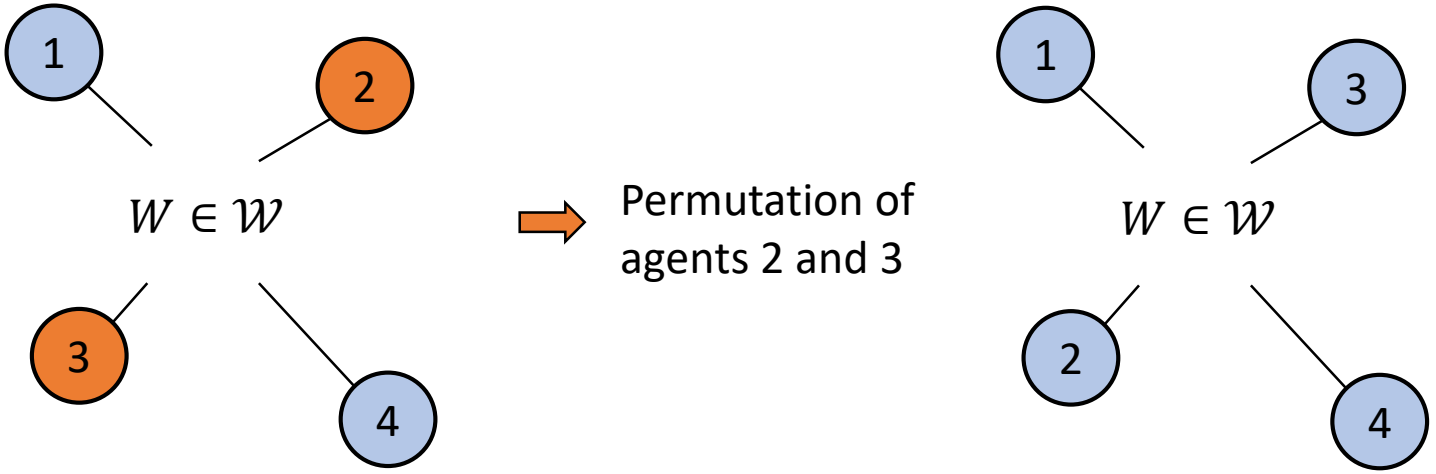
Questions

- Can we build an equivalent PEP formulation computing agent-independent bound (by construction) ?
- When does this independence occur ?

Agents are Interchangeable

Assumption

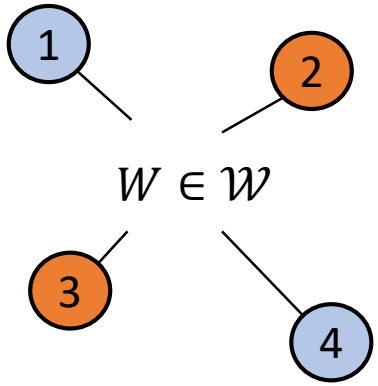
- **no agent plays a specific role** in the algorithm, the performance measure or the initial conditions.
- **any permutation** of agents lead to the same worst-case performance



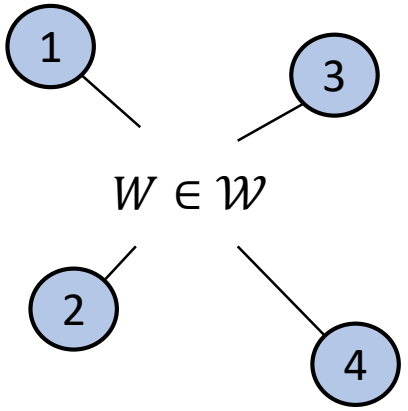
worst-case value :
$wc(1,2,3,4) = P$

worst-case value :
$wc(1,3,2,4) = P$

Agents are Interchangeable



Permutation of agents 2 and 3



Worst-case value

$$wc(1,2,3,4) = P$$

PEP solution

$$F_{\Pi_1} = [f_1\ f_2\ f_3\ f_4]$$

$$G_{\Pi_1} = P_{\Pi_1}^T P_{\Pi_1}$$

$$P_{\Pi_1} = [P_1\ P_2\ P_3\ P_4\ ]$$

Worst-case value

$$wc(1,3,2,4) = P$$

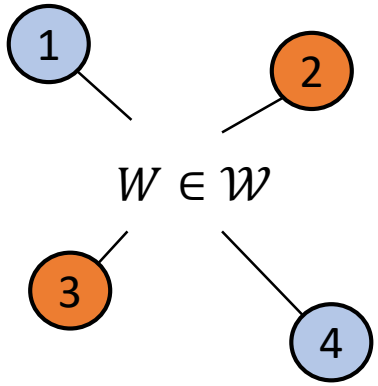
PEP solution

$$F_{\Pi_2} = [f_1\ f_3\ f_2\ f_4]$$

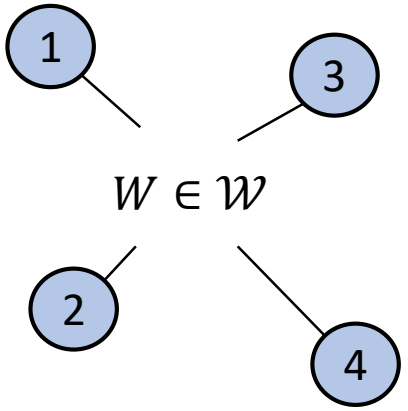
$$G_{\Pi_2} = P_{\Pi_2}^T P_{\Pi_2}$$

$$P_{\Pi_2} = [P_1\ P_3\ P_2\ P_4\ ]$$

Agents are Interchangeable



Permutation of agents 2 and 3



Worst-case value

$$wc(1,2,3,4) = P$$

PEP solution

$$F_{\Pi_1} = [f_1\ f_2\ f_3\ f_4]$$

$$G_{\Pi_1} = P_{\Pi_1}^T P_{\Pi_1}$$

$$P_{\Pi_1} = [P_1\ P_2\ P_3\ P_4]$$

Worst-case value

$$wc(1,3,2,4) = P$$

PEP solution

$$F_{\Pi_2} = [f_1\ f_3\ f_2\ f_4]$$

$$G_{\Pi_2} = P_{\Pi_2}^T P_{\Pi_2}$$

$$P_{\Pi_2} = [P_1\ P_3\ P_2\ P_4]$$

PEP objective is linear in G and F

Solution $\frac{1}{2}\left(F_{\Pi_1} + F_{\Pi_2}\right)\ ;\ \frac{1}{2}\left(G_{\Pi_1} + G_{\Pi_2}\right)$ has the **same worst-case P**

Average permuted PEP solutions

$$F_{\Pi_1} = [f_1\ f_2\ f_3] \qquad G_{\Pi_1} = P_{\Pi_1}^T P_{\Pi_1} \qquad P_{\Pi_1} = [P_1\ P_2\ P_3]$$
$$F_{\Pi_2} = [f_1\ f_3\ f_2] \qquad G_{\Pi_2} = P_{\Pi_2}^T P_{\Pi_2} \qquad P_{\Pi_2} = [P_1\ P_3\ P_2]$$
$$F_{\Pi_3} = [f_2\ f_1\ f_3] \qquad G_{\Pi_3} = P_{\Pi_3}^T P_{\Pi_3} \qquad P_{\Pi_3} = [P_2\ P_1\ P_3]$$
$$F_{\Pi_4} = [f_2\ f_3\ f_1] \qquad G_{\Pi_4} = P_{\Pi_4}^T P_{\Pi_4} \qquad P_{\Pi_4} = [P_2\ P_3\ P_1]$$
$$F_{\Pi_5} = [f_3\ f_1\ f_2] \qquad G_{\Pi_5} = P_{\Pi_5}^T P_{\Pi_5} \qquad P_{\Pi_5} = [P_3\ P_1\ P_2]$$
$$F_{\Pi_6} = [f_3\ f_2\ f_1] \qquad G_{\Pi_6} = P_{\Pi_6}^T P_{\Pi_6} \qquad P_{\Pi_6} = [P_3\ P_2\ P_1]$$

avg

$$F^S = [f_A\ f_A\ f_A]$$

avg

$$G^S = \begin{bmatrix} G_A & \cdots & G_B \\ \vdots & \ddots & \vdots \\ G_B & \cdots & G_A \end{bmatrix}$$

with $f_A = \dfrac{(N-1)!}{N!}\sum_{i=1}^{N} f_i = \dfrac{1}{N}\sum_{i=1}^{N} f_i$

with $G_A = \dfrac{(N-1)!}{N!}\sum_{i=1}^{N} G_{ii} = \dfrac{1}{N}\sum_{i=1}^{N} G_{ii}$

$$G_B = \frac{(N-2)!}{N!}\sum_{i=1}^{N}\sum_{j\neq i}^{N} G_{ij} = \frac{1}{N(N-1)}\sum_{i=1}^{N}\sum_{j\neq i}^{N} G_{ij}$$

$\boldsymbol{F^S}$ and $\boldsymbol{G^S}$ are valid PEP solution with the **same worst-case value**

Agent-Symmetric PEP solution

Proposition (Existence of an agents-symmetric solution of a PEP).

Let F and G be solution of an agent dependent PEP formulation for distributed optimization. If **all the N agents are interchangeable** in the PEP, then the symmetrized solution $\boldsymbol{F^s}$ **and** $\boldsymbol{G^s}$ provides another valid solution for the PEP, with the **same worst-case value**:

$$F^s = \frac{1}{N!}\sum_{\Pi} F_{\Pi} = [f_A^T \dots f_A^T] \quad \text{with } f_A = \frac{1}{N}\sum_{i=1}^{N} f_i$$

$$G^s = \begin{bmatrix} G_A & \cdots & G_B \\ \vdots & \ddots & \vdots \\ G_B & \cdots & G_A \end{bmatrix} \quad \text{with } G_A = \frac{1}{N}\sum_{i=1}^{N} G_{ii}\,, \quad G_B = \frac{1}{N(N-1)}\sum_{i=1}^{N}\sum_{j\neq i}^{N} G_{ij}$$

➡ $\mathrm{PEP}(F^s, G^s)$ can be written using only f_A, G_A and G_B as variables

Size depends on N — Size independent of N

Agent Independent Performance

$(N \geq 2)$

If

- ✓ the algorithm and its performance evaluation settings are Gram-representable

  SDP PEP ✓

  → *can be expressed with scalar products of iterates and gradients*

- ✓ The agents are interchangeable in the PEP

  compact SDP PEP ✓

  Problem size independent of N

- ✓ The compact PEP formulation can be expressed without any dependence on the number of agents N

  agent-independent SDP PEP ✓

Then

- ➢ The worst-case performance in the given setting is independent of the number of agents N
- ➢ General worst-case guarantee can simply be derived using $N = 2$

Agent Independent Performance

$(N \geq 2)$

If

✓ The compact PEP formulation can be expressed without any dependence on the number of agents N

agent-independent SDP PEP ✓

When is it satisfied ? → Often

Satisfied (at least) by any PEP involving combinations of

- **Same agent** scalar product : $x_i^T y_i$
- **Pair of agents average** of the scalar products $\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} x_i^T y_j$
- Agent average of the scalar products of two **centered variables** associated with the **same agents** $\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^T(y_i - \bar{y})$

Subsets of interchangeable agents

Assumption

- the set of agents can be decomposed in distinct subsets $\mathcal{V}_1, \dots, \mathcal{V}_J$ containing interchangeable agents
- **any permutation** of agents **from the same subsets** lead to the same worst-case performance

symmetrize the PEP for each subset of agents and write it in a compact form whose **size only depends on the number of subsets J**.

Example

One agent	All the others
$\mathcal{V}_1 = \{1\}$	$\mathcal{V}_2 = \{2, \dots, N\}$

Performance of the worst agent

Performance measure

$$\max_{i \in \{1,\dots,N\}} \; f\left(x_i^K\right) - f(x^*)$$

Distributed optimization problem

$$\min_x f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$$

PEP

$$\max_{F,G} \quad f(x_1^K) - f(x^*)$$

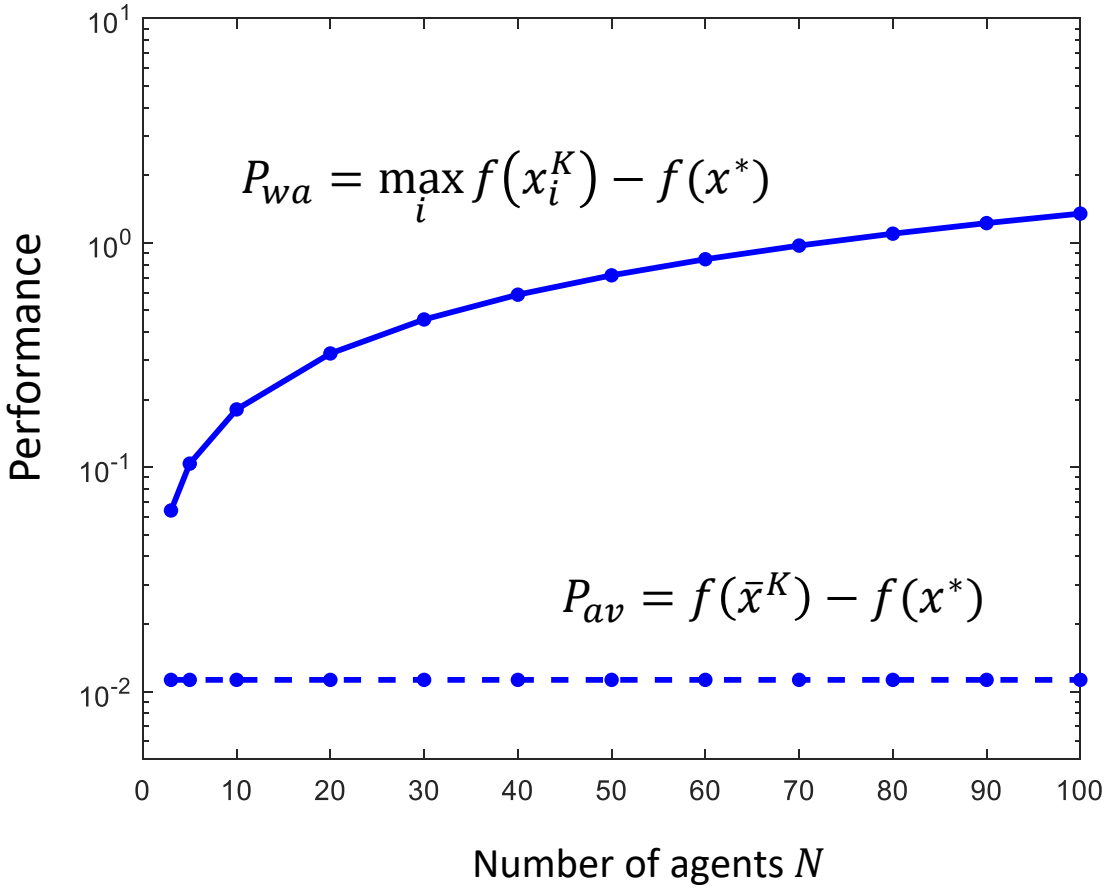
Optimal PEP solution F^*, G^* gives a specific place to agent 1

➡ Agent 1 cannot be permuted with the others in F^*, G^*

Sets of interchangeable agents

One agent	All the others
$\mathcal{V}_1 = \{1\}$	$\mathcal{V}_2 = \{2, \dots, N\}$

Performance evolution with N

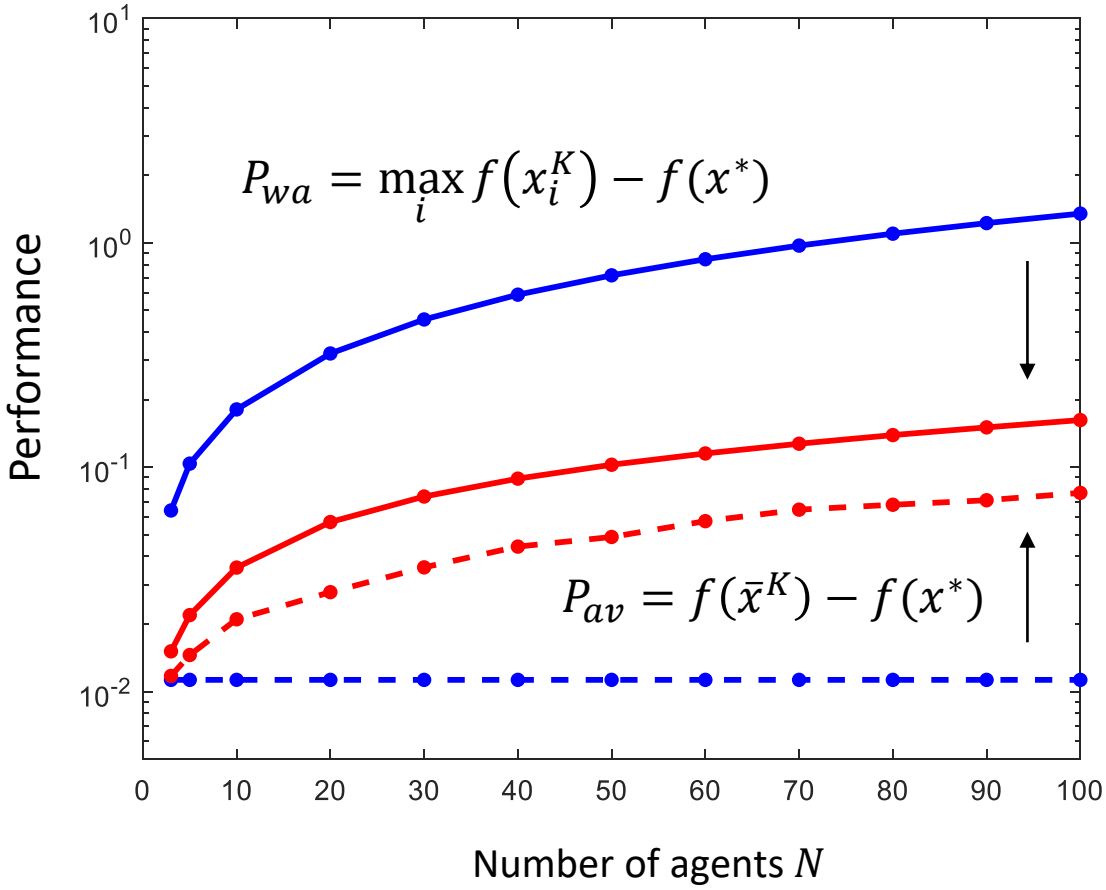


step−size optimized for P_{av}

➢ Performance of the worst-agent scale sublinearly with N

EXTRA – $\mathrm{K} = 15$ iterations, $\lambda(\mathrm{W}) \in [-0.5, 0.5]$

Performance evolution with N



step−sizes optimized for P_{wa} *(diminishing with N)*

step−size optimized for P_{av}

➢ Performance of the worst-agent scale sublinearly with N

EXTRA – $\mathrm{K} = 15$ iterations, $\lambda(\mathrm{W}) \in [-0.5, 0.5]$

Conclusion

Sébastien Colla

Toolbox
PESTO
PEPit

- PEP for distributed optimization

  PEP idea: worst-cases are solutions of optimization problems

- Equivalent agent-compact PEP formulation

  → Often independent of the number of agents

  Exploits agents symmetries and interchangeability in the PEP

- Subsets of interchangeable agents

Compact PEP formulation for many settings in distributed optimization
→ *e.g.* performance of the worst agent (for any N)

Future works

- ❑ Leverage the *subsets of interchangeable agents* approach to analyze new settings (malicious agents, topology constraints, etc)

References

[Colla 2023] S. Colla, J. M. Hendrickx, “Automatic Performance Estimation for Decentralized Optimization”, IEEE Transactions on Automatic Control, 2023.

[to appear] S. Colla, J. M. Hendrickx, “Exploiting Agent Symmetries for Performance Analysis of Distributed Optimization Methods”, *preprint to appear soon.*