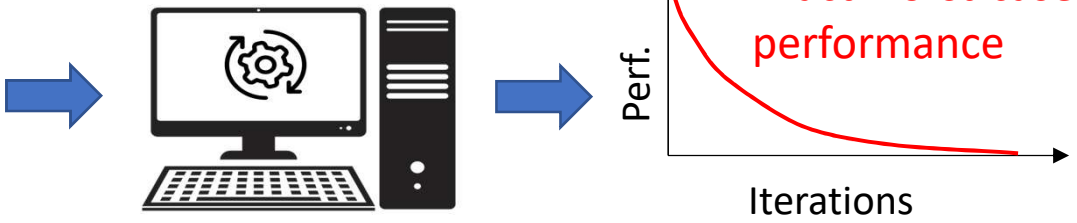
UCLouvain icteam

June, 29 2021

Automatic Performance Estimation for Decentralized Optimization

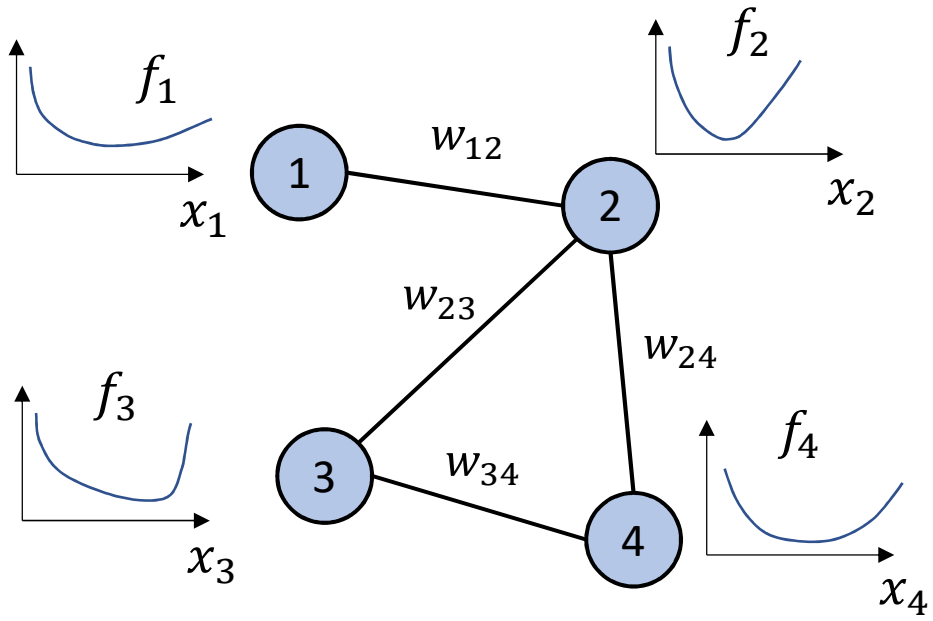
Sébastien Colla, Julien Hendrickx

```
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
```



Decentralized Optimization

$$\min_x f(x) = \sum_i f_i(x)$$

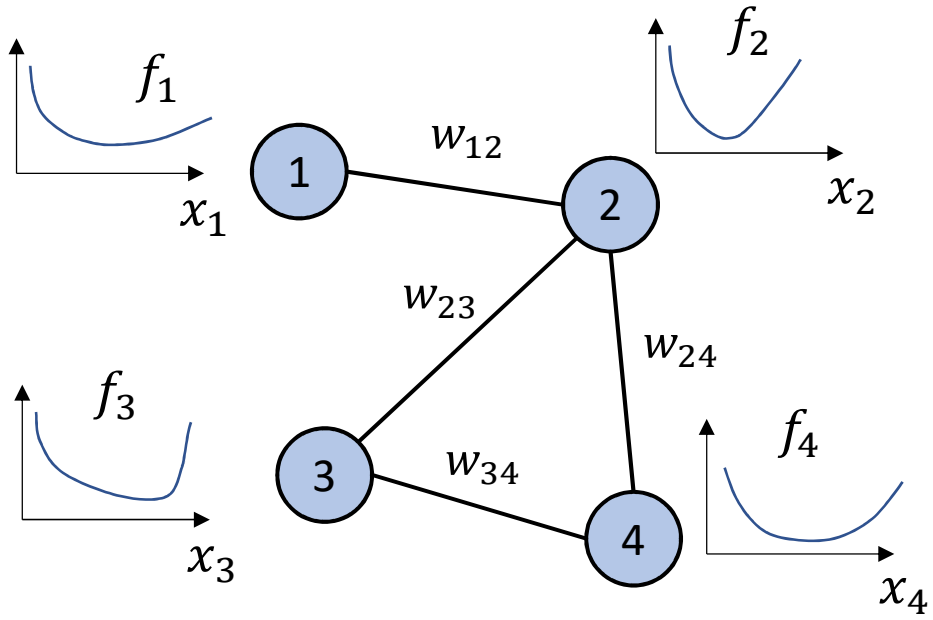


Decentralization

- Local function: f_i

Decentralized Optimization

$$\min_x f(x) = \sum_i f_i(x)$$



Decentralization

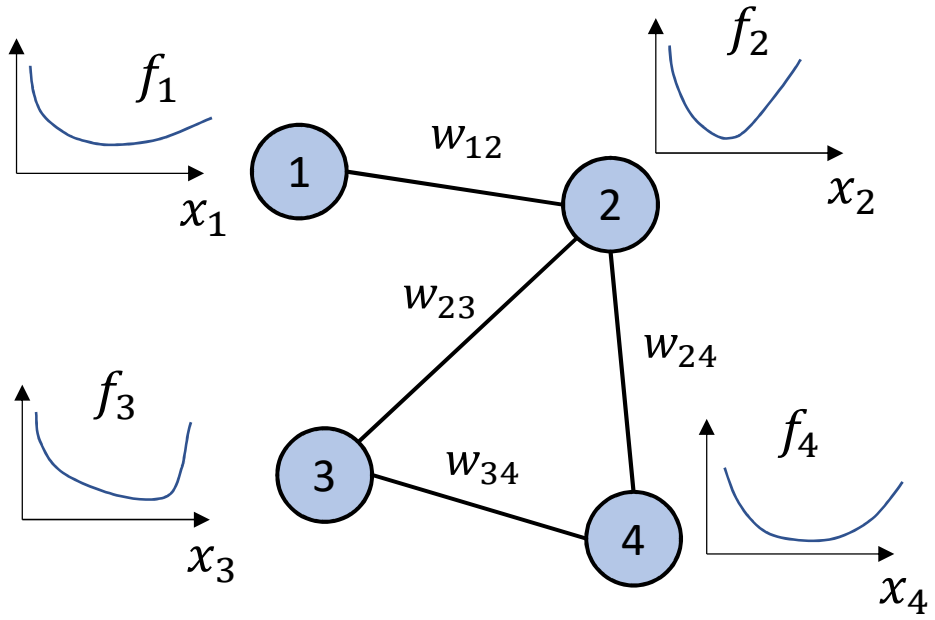
- Local function: f_i

Iterative algorithm

- Local computations

Decentralized Optimization

$$\min_x f(x) = \sum_i f_i(x)$$



Decentralization

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

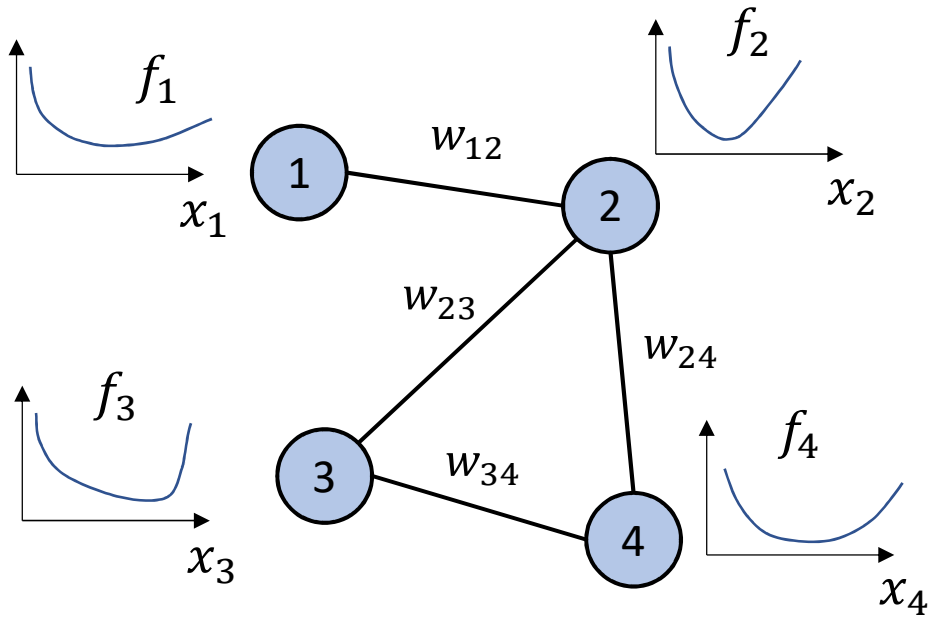
Iterative algorithm

- Local computations

Decentralized Optimization

$$\min_{x} f(x) = \sum_{i} f_i(x)$$

$$\min_{x_1,\dots,x_N} \sum_{i} f_i(x_i)$$
$$\text{s.t. } x_i = x_j \quad \forall\, (i,j) \text{ neighbors}$$



Decentralization

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

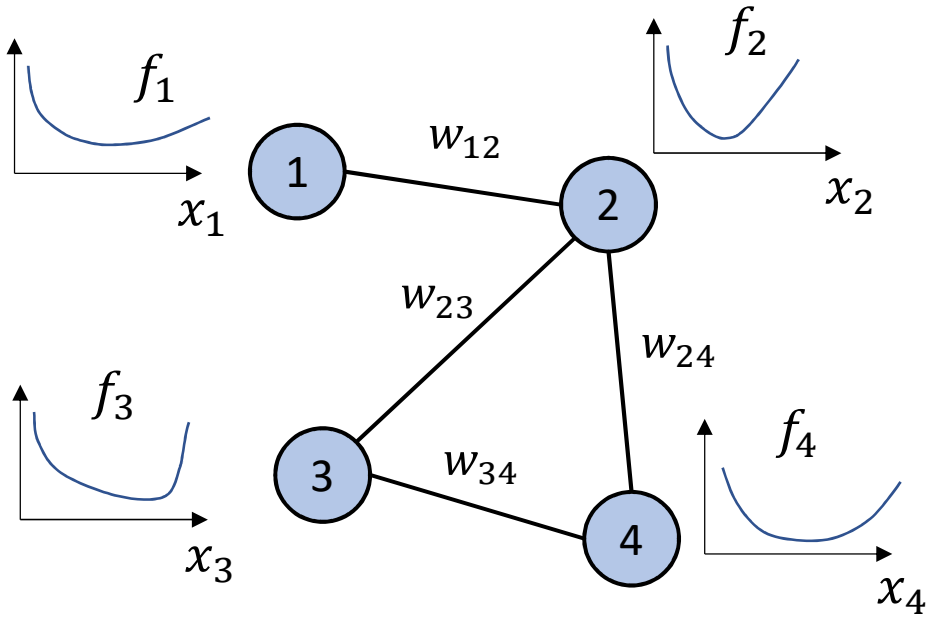
Iterative algorithm

- Local computations

Decentralized Optimization

$$\min_x f(x) = \sum_i f_i(x)$$

$$\min_{x_1,\dots,x_N} \sum_i f_i(x_i) \quad \text{s.t. } x_i = x_j \ \ \forall\,(i,j) \text{ neighbors}$$



Decentralization

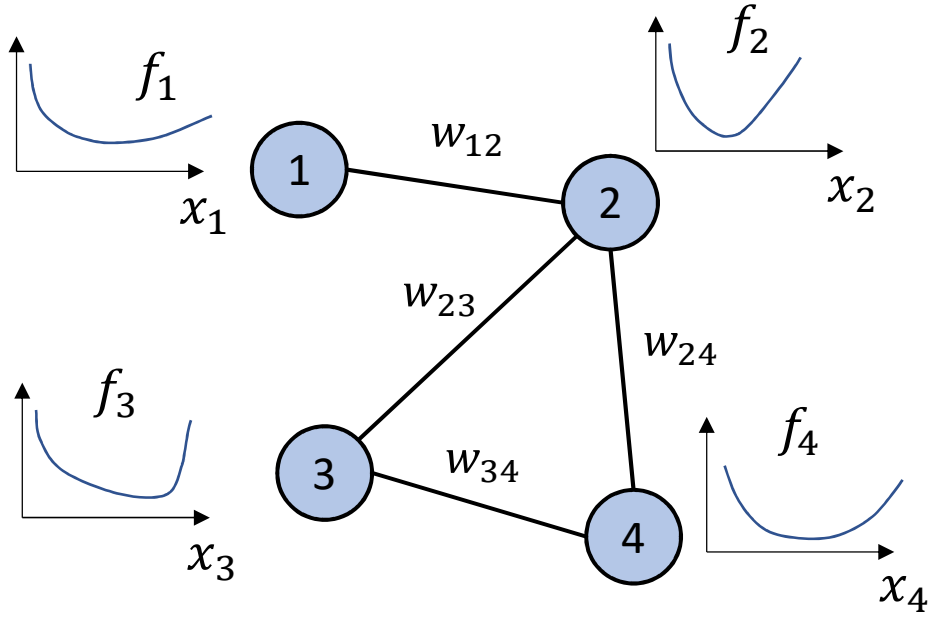
- Local function: f_i
- Local copy of x: x_i

Iterative algorithm

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

Decentralized Gradient Descent (DGD)

$$\min_{x_1,\dots,x_N} \sum_i f_i(x_i)$$
$$\text{s.t. } x_i = x_j \quad \forall\ (i,j) \text{ neighbors}$$



Decentralization

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

Decentralized Gradient Descent (DGD)

For each iteration k

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

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

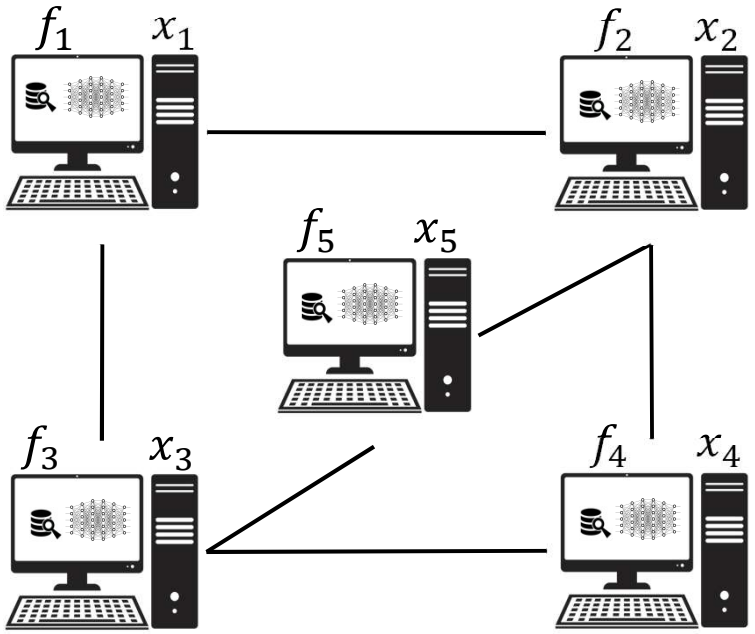
Motivations: Decentralized Machine Learning

Notations

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

Model training

$$\min_x \sum_k \text{Error}(x, d_k) + \text{regul (x)}$$



Decentralization

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

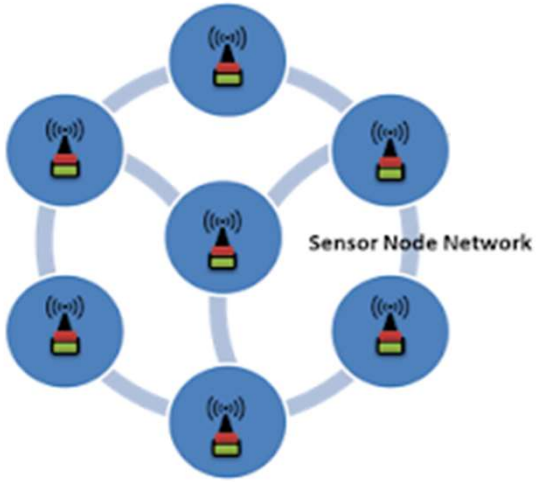
$$f_i(x) = \sum_{k \in \mathcal{D}_i} \text{Error}(x, d_k)$$

- Local copy of x

Motivations Big data – Privacy – Speed Up

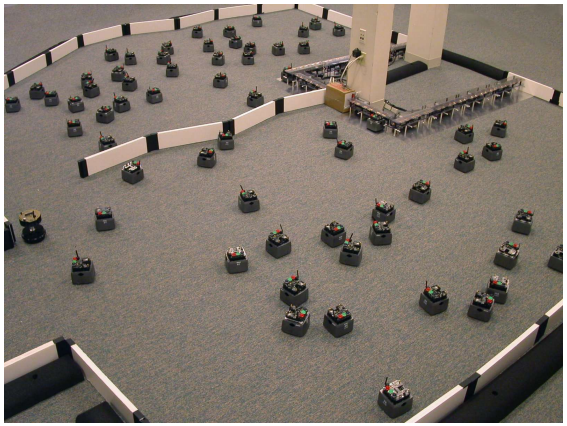
Other applications

Sensor Network



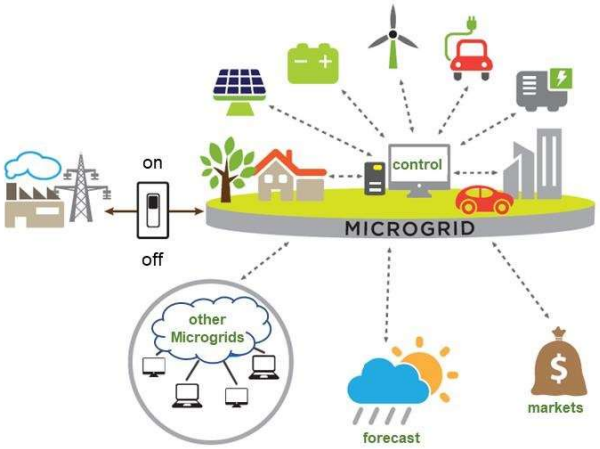
Informatics–Wireless sensor network

Multi-robot systems



Multi-Robot Systems Engineering
MIT , James McLurkin

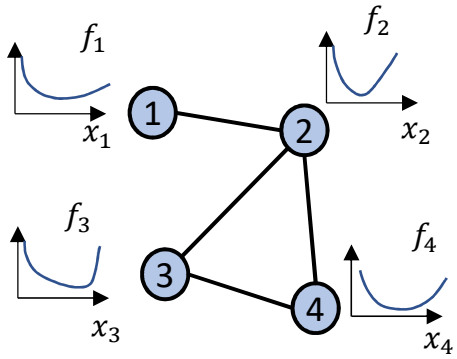
Micro-Grid



ResearchGate, Planning and implementation of bankable microgrids, Michael Stadler

Decentralized Optimization

- Many challenges for better methods

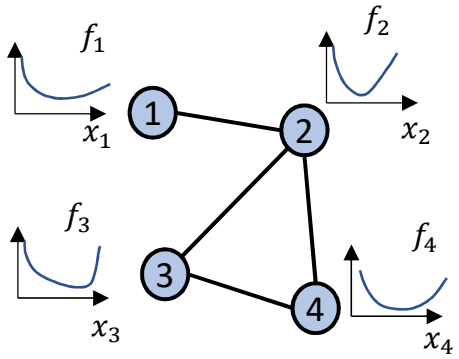


Decentralized Optimization

Many challenges for better methods

BUT

Analysis highly complex

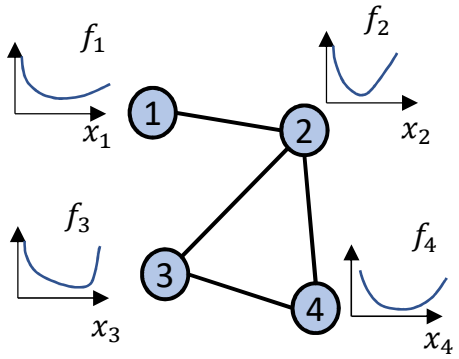


Decentralized Optimization

Many challenges for better methods

BUT

Analysis highly complex



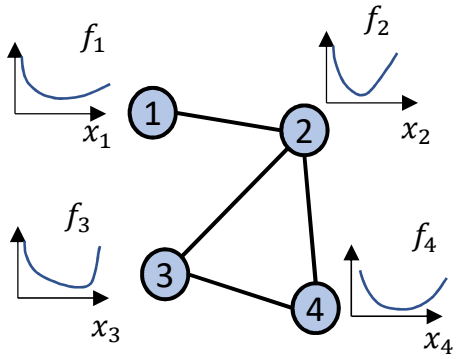
- Design: **long and complex** process
- Performance bounds: complex and **conservative**

Decentralized Optimization

Many challenges for better methods

BUT

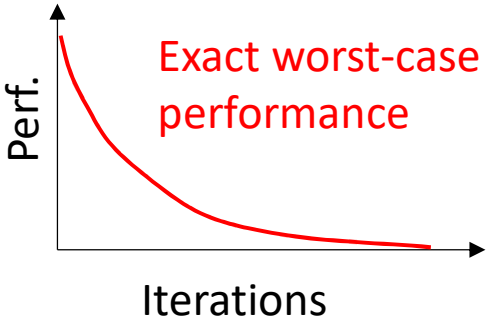
Analysis highly complex



- Design: **long and complex** process
- Performance bounds: complex and **conservative**
- Difficult algorithms comparisons
- Difficult parameters tuning

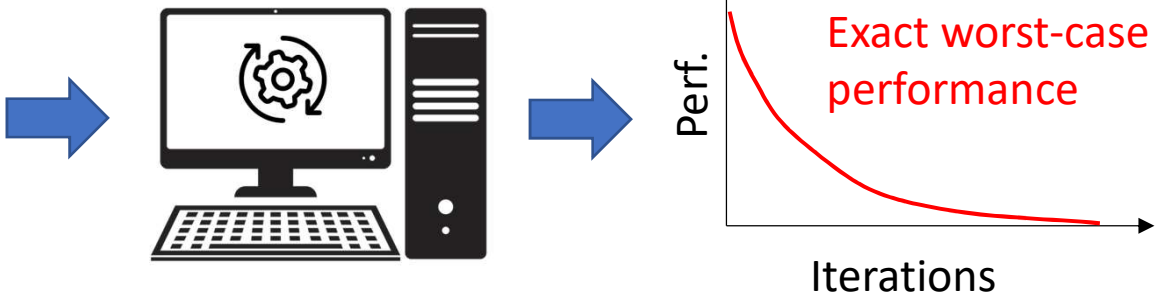
🎯 Objective

```
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
```



🎯 Objective

```
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
```



Impact for decentralized optimization

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

Performance Estimation Problem (PEP)

Idea: Worst-cases are solutions to optimization problems

Performance Estimation Problem (PEP)

Idea: Worst-cases are solutions to optimization problems

$$\max_{f,\, x^0,\ldots,x^K} \quad \mathrm{perf}(f,\, x^0, \ldots, x^K) \overset{\textit{e.g.}}{=} f(x^K) - f(x^*)$$

With $f \in$ class of functions

x_0 initial condition

x^k from the algorithm analyzed

Performance Estimation Problem (PEP)

Idea: Worst-cases are solutions to optimization problems

$$\max_{f,\, x^0,\ldots,x^K} \quad \mathrm{perf}(f,\, x^0,\ldots,x^K) \overset{\textit{e.g.}}{=} f(x^K) - f(x^*)$$

With $f \in$ class of functions

x_0 initial condition

x^k from the algorithm analyzed

Infinite-Dimensional problem

Performance Estimation Problem (PEP)

Idea: Worst-cases are solutions to optimization problems

$$\max_{f,\, x^0,\ldots,x^K} \quad \mathrm{perf}(f, x^0, \ldots, x^K) \overset{e.g.}{=} f(x^K) - f(x^*)$$

With $f \in$ class of functions

x_0 initial condition

x^k from the algorithm analyzed

Infinite-Dimensional problem

Using SDP reformulation

PEP can be solved exactly for a wide class of **centralized** first-order algorithms.

Existing toolbox (PESTO)

[Taylor17]

Performance Estimation Problem (PEP)

Idea: Worst-cases are solutions to optimization problems

$$\max_{f,\, x^0,\ldots,x^K,\, G} \ \mathrm{perf}(f, x^0, \ldots, x^K) \overset{e.g.}{=} f(x^K) - f(x^*)$$

With $f \in$ class of functions

x_0 initial condition

x^k from the algorithm analyzed

$G \in$ class of graphs

Infinite-Dimensional problem

Using SDP reformulation

PEP can be solved exactly for a wide class of **centralized** first-order algorithms.

Existing toolbox (PESTO)

[Taylor17]

*No formulation for **communication networks***

PEP for DGD: network given a priori

PEP for DGD: network given a priori

$$\max_{\substack{f,\, x^0,\ldots,x^K,G \\ y^0,\ldots,y^{K-1}}} \quad \mathrm{perf}(f,\, x^0,\ldots,x^K)$$

With $f \in$ class of functions

x_0 initial condition

$W(G)$ given network matrix

Iterates from DGD

$$\begin{cases} y_i^k = \sum_j w_{ij}\, x_j^k \\ x_i^{k+1} = y_i^k - \alpha \nabla f_i(x_i^k) \end{cases}$$

For all $i = 1 \ldots N$,
For all $k = 0 \ldots K-1$

PEP for DGD: network given a priori

$$\max_{\substack{f,\, x^0,\ldots,x^K,G \\ y^0,\ldots,y^{K-1}}} \quad \mathrm{perf}(f,\, x^0,\ldots,x^K)$$

With $f \in$ class of functions

x_0 initial condition

$W(G)$ given network matrix

Iterates from DGD

$$\begin{cases} y_i^k = \sum_j w_{ij}\, x_j^k \\ x_i^{k+1} = y_i^k - \alpha \nabla f_i(x_i^k) \end{cases}$$

For all $i = 1 \ldots N$,
For all $k = 0 \ldots K-1$

➡ ***Exact Formulation***

PEP for DGD: class of networks

$$\max_{\substack{f,\, x^0,\ldots,x^K,\boldsymbol{G} \\ y^0,\ldots,y^{K-1}}} \quad \mathrm{perf}(f,\, x^0,\ldots,x^K)$$

With $f \in$ class of functions

x_0 initial condition

$\boldsymbol{W(G)}$ **Any symmetric doubly stochastic matrix with given range of eigenvalues $[\lambda^-, \lambda^+]$**

Iterates from DGD

$$\begin{cases} y_i^k = \sum_j w_{ij}\, x_j^k \\ x_i^{k+1} = y_i^k - \alpha \nabla f_i(x_i^k) \end{cases}$$

For all $i = 1 \ldots N$,
For all $k = 0 \ldots K-1$

PEP for DGD: class of networks

$$\max_{\substack{f,\, x^0,\ldots,x^K,\boldsymbol{G} \\ y^0,\ldots,y^{K-1}}} \quad \mathrm{perf}(f,\, x^0,\ldots,x^K)$$

With $f \in$ class of functions

x_0 initial condition

$\boldsymbol{W(G)}$ **Any symmetric doubly stochastic matrix with given range of eigenvalues $[\lambda^-, \lambda^+]$**

Iterates from DGD

$$\begin{cases} y_i^k = \sum_j w_{ij} x_j^k \\ x_i^{k+1} = y_i^k - \alpha \nabla f_i(x_i^k) \end{cases}$$

?

For all $i = 1 \ldots N$,
For all $k = 0 \ldots K-1$

PEP for DGD: class of networks

$$\max_{\substack{f,\, x^0,\ldots,x^K,G \\ y^0,\ldots,y^{K-1}}} \quad \mathrm{perf}(f, x^0, \ldots, x^K)$$

With $f \in$ class of functions

x_0 initial condition

$\boldsymbol{W(G)}$ **Any symmetric doubly stochastic matrix with given range of eigenvalues $[\lambda^-, \lambda^+]$**

Find constraints between y_i^k and x_i^k

Iterates from DGD

$$\begin{cases} y_i^k = \sum_j w_{ij} x_j^k \\ x_i^{k+1} = y_i^k - \alpha \nabla f_i(x_i^k) \end{cases}$$

?

For all $i = 1 \ldots N$,
For all $k = 0 \ldots K-1$

Constraints for consensus steps

- Search Space for x_i^k and y_i^k

(C1) $$y_i^k = \sum_{j=1}^{N} w_{ij}\, x_j^k$$

For each agent $i = 1 \dots N$,
For each consensus step $k = 0 \dots K-1$

(C2) $W = [w_{ij}]$ is a symmetric and doubly-stochastic matrix with a given range of eigenvalues $[\lambda^-, \lambda^+]$

Constraints for consensus steps

- Search Space for X and Y

(C1) $$y_i^k = \sum_{j=1}^{N} w_{ij}\, x_j^k$$

For each agent $i = 1 \dots N$,
For each consensus step $k = 0 \dots K-1$

compact notation $Y = WX$ with $Y_{ik} = y_i^k,\ X_{ik} = x_i^k.$

(C2) $W = [w_{ij}]$ is a symmetric and doubly-stochastic matrix with a given range of eigenvalues $[\lambda^-, \lambda^+]$

Constraints for consensus steps

- Search Space for X and Y

(C1) $$y_i^k = \sum_{j=1}^{N} w_{ij}\, x_j^k$$

For each agent $i = 1 \dots N$,
For each consensus step $k = 0 \dots K-1$

compact notation $Y = WX$ with $Y_{ik} = y_i^k,\ X_{ik} = x_i^k.$

(C2) $W = [w_{ij}]$ is a symmetric and doubly-stochastic matrix with a given range of eigenvalues $[\lambda^-, \lambda^+]$

- **Necessary constraints** for describing (C1) and (C2)

$\bar{X}, \bar{Y}$: agents average vectors

$X_\perp, Y_\perp$: centered matrices

$$X_\perp = X - \mathbf{1}\bar{X}^T,\ Y_\perp = Y - \mathbf{1}\bar{Y}^{\mathrm{T}}$$

$$\bar{X} = \bar{Y} \quad (1)$$

$$\lambda^- X_\perp^T X_\perp \preccurlyeq X_\perp^T Y_\perp \preccurlyeq \lambda^+ X_\perp^T X_\perp \quad (2)$$

$$(Y_\perp - \lambda^- X_\perp)^T (Y_\perp - \lambda^+ X_\perp) \preccurlyeq 0 \quad (3)$$

Constraints for consensus steps

- Search Space for X and Y

(C1) $$y_i^k = \sum_{j=1}^{N} w_{ij}\, x_j^k$$

For each agent $i = 1 \dots N$,
For each consensus step $k = 0 \dots K-1$

compact notation $Y = WX$ with $Y_{ik} = y_i^k,\ X_{ik} = x_i^k.$

(C2) $W = [w_{ij}]$ is a symmetric and doubly-stochastic matrix with a given range of eigenvalues $[\lambda^-, \lambda^+]$

- **Necessary constraints** for describing (C1) and (C2)

$\bar{X}, \bar{Y}$: agents average vectors

$X_\perp, Y_\perp$: centered matrices

$$X_\perp = X - \mathbf{1}\bar{X}^T,\ Y_\perp = Y - \mathbf{1}\bar{Y}^{\mathrm{T}}$$

$$\bar{X} = \bar{Y} \quad (1)$$

$$\lambda^- X_\perp^T X_\perp \preccurlyeq X_\perp^T Y_\perp \preccurlyeq \lambda^+ X_\perp^T X_\perp \quad (2)$$

$$(Y_\perp - \lambda^- X_\perp)^T (Y_\perp - \lambda^+ X_\perp) \preccurlyeq 0 \quad (3)$$

Simplification when $-\lambda^- = \lambda^+ = \lambda$: $Y_\perp^T Y_\perp \preccurlyeq \lambda^2 X_\perp^T X_\perp$

Constraints for consensus steps

Summary of the constraints for **consensus steps between Y and X**

$$\bar{X} = \bar{Y} \quad (1)$$

$$\lambda^{-} X_{\perp}^{T} X_{\perp} \preccurlyeq X_{\perp}^{T} Y_{\perp} \preccurlyeq \lambda^{+} X_{\perp}^{T} X_{\perp} \quad (2)$$

$$(Y_{\perp} - \lambda^{-} X_{\perp})^{T} (Y_{\perp} - \lambda^{+} X_{\perp}) \preccurlyeq 0 \quad (3)$$

Advantages of our constraints

- ✓ **Independent** of the algorithm
- ✓ Link different consensus steps that **use the same matrix**
- ✓ Can be incorporated into SDP formulation of PEP, which can be **solved efficiently**

PEP for DGD: class of networks

$$\max_{\substack{f,\, x^0,\ldots,x^K,G \\ y^0,\ldots,y^{K-1}}} \quad \mathrm{perf}(f,\, x^0,\ldots,x^K)$$

With $f \in$ class of functions

x_0 initial condition

$\mathrm{W(G)}$ Any symmetric doubly stochastic matrix with given range of eigenvalues $[\lambda^-, \lambda^+]$

Iterates from DGD
$$\begin{cases} y_i^k = \sum_j w_{ij}\, x_j^k \\ x_i^{k+1} = y_i^k - \alpha \nabla f_i(x_i^k) \end{cases}$$

For all $i = 1 \ldots N$,
For all $k = 0 \ldots K-1$

PEP for DGD: class of networks

$$\max_{\substack{f,\, x^0,\ldots,x^K,\, G \\ y^0,\ldots,y^{K-1}}} \quad \mathrm{perf}(f, x^0, \ldots, x^K)$$

With $f \in$ class of functions

x_0 initial condition

$W(G)$ ~~Any symmetric doubly stochastic matrix with given range of eigenvalues $[\lambda^-, \lambda^+]$~~

Iterates from DGD

$$\begin{cases} y_i^k = \sum_j w_{ij}\, x_j^k \\ x_i^{k+1} = y_i^k - \alpha \nabla f_i(x_i^k) \end{cases}$$

For all $i = 1 \ldots N$,
For all $k = 0 \ldots K-1$

PEP for DGD: Spectral formulation

(Relaxation)

$$\max_{\substack{f,\, x^0,\ldots,x^K \\ y^0,\ldots,y^{K-1}}} \quad \mathrm{perf}(f, x^0, \ldots, x^K)$$

With $f \in$ class of functions

x_0 initial condition

Iterates from DGD $\begin{cases} x_i^{k+1} = y_i^k - \alpha \nabla f_i(x_i^k) \end{cases}$ For all $i = 1 \ldots N$, For all $k = 0 \ldots K-1$

Consensus steps

$Y = WX$

W symmetric, doubly stochastic, $\lambda(W) \in [\lambda^-, \lambda^+]$

$$\begin{cases} \bar{X} = \bar{Y} & (1) \\ \lambda^- X_\perp^T X_\perp \preccurlyeq X_\perp^T Y_\perp \preccurlyeq \lambda^+ X_\perp^T X_\perp & (2) \\ (Y_\perp - \lambda^- X_\perp)^T (Y_\perp - \lambda^+ X_\perp) \preccurlyeq 0 & (3) \end{cases}$$

PEP for DGD: Spectral formulation

(Relaxation)

$$\max_{\substack{f,\, x^0,\ldots,x^K \\ y^0,\ldots,y^{K-1}}} \quad \mathrm{perf}(f, x^0, \ldots, x^K)$$

With $f \in$ class of functions

x_0 initial condition

Iterates from DGD $\begin{cases} x_i^{k+1} = y_i^k - \alpha \nabla f_i(x_i^k) \end{cases}$ For all $i = 1 \ldots N$, For all $k = 0 \ldots K-1$

Consensus steps

$Y = WX$

W symmetric doubly stochastic $\lambda(W) \in [\lambda^-, \lambda^+]$

$$\begin{cases} \bar{X} = \bar{Y} & (1) \\ \lambda^- X_\perp^T X_\perp \preccurlyeq X_\perp^T Y_\perp \preccurlyeq \lambda^+ X_\perp^T X_\perp & (2) \\ (Y_\perp - \lambda^- X_\perp)^T (Y_\perp - \lambda^+ X_\perp) \preccurlyeq 0 & (3) \end{cases}$$

⇒ *Upper bounds for the worst-case performance of DGD*

Theoretical guarantee for DGD

[NOR17]

After K steps of DGD with $\alpha = \frac{h}{\sqrt{K}}$, for solving $\min_x f(x) = \sum_i f_i(x)$

With optimal solution x^*

If

(i) f_i convex for each i and (sub)gradients bounded by B;

(ii) Identical starting points: $x_i^0 = x^0$ for each i, s.t. $\|x^0 - x^*\|^2 \leq R^2$

(iii) Communication matrix W: symmetric, doubly stochastic,
$\lambda(W) \in [-\lambda, \lambda]$ (except for $\lambda_1(W) = 1$)

Then

$$f(x_{av}) - f(x^*) \leq RB\left(\frac{h^{-1} + h}{2\sqrt{K}} + \frac{2h}{\sqrt{K}(1-\lambda)}\right) \quad \text{where} \quad x_{av} = \frac{1}{K}\sum_k \frac{1}{N}\sum_i x_i^k$$

Theoretical guarantee for DGD

[NOR17]

After K steps of DGD with $\alpha = \frac{h}{\sqrt{K}}$, for solving $\min_x f(x) = \sum_i f_i(x)$

With optimal solution x^*

If

(i) f_i convex for each i and (sub)gradients bounded by B;

(ii) Identical starting points: $x_i^0 = x^0$ for each i, s.t. $\|x^0 - x^*\|^2 \leq R^2$

(iii) Communication matrix W: symmetric, doubly stochastic,
$\lambda(W) \in [-\lambda, \lambda]$ (except for $\lambda_1(W) = 1$)

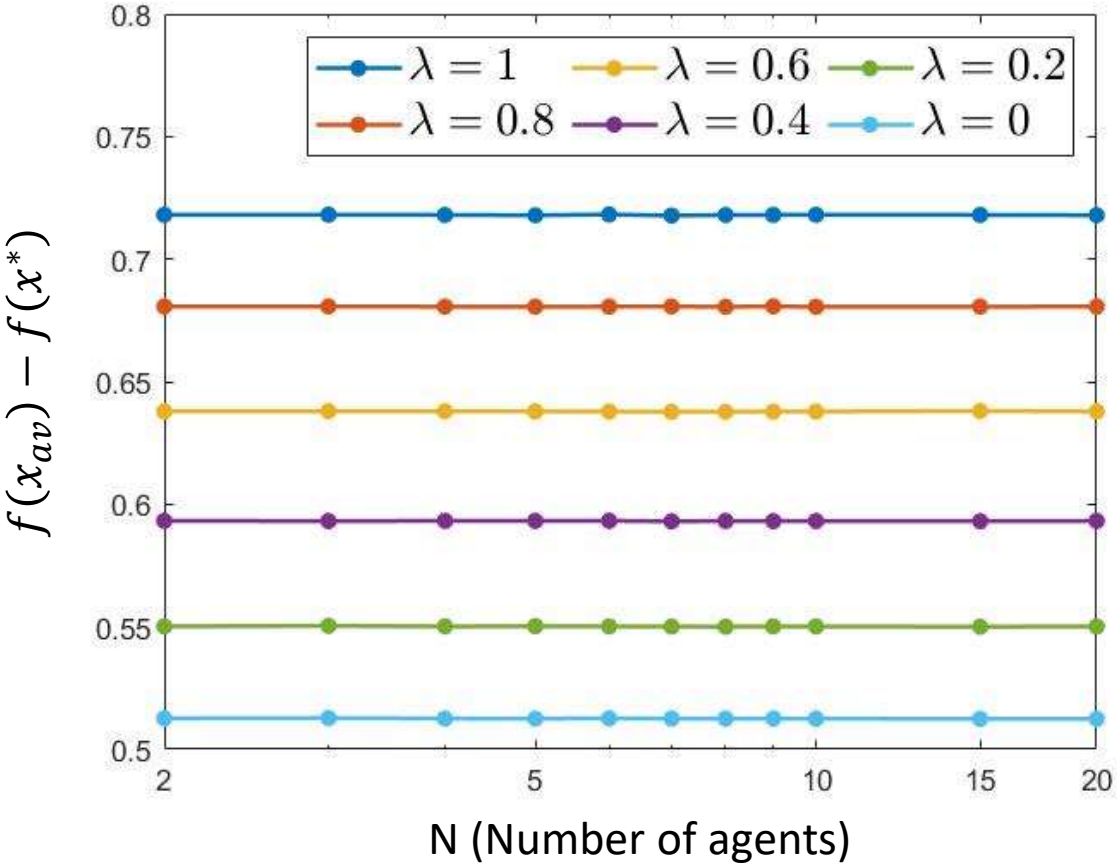
Then

$$f(x_{av}) - f(x^*) \leq RB\left(\frac{h^{-1} + h}{2\sqrt{K}} + \frac{2h}{\sqrt{K}(1-\lambda)}\right) \quad \text{where} \quad x_{av} = \frac{1}{K}\sum_k \frac{1}{N}\sum_i x_i^k$$

$f(x_{av}) - f(x^*)$: Performance measure

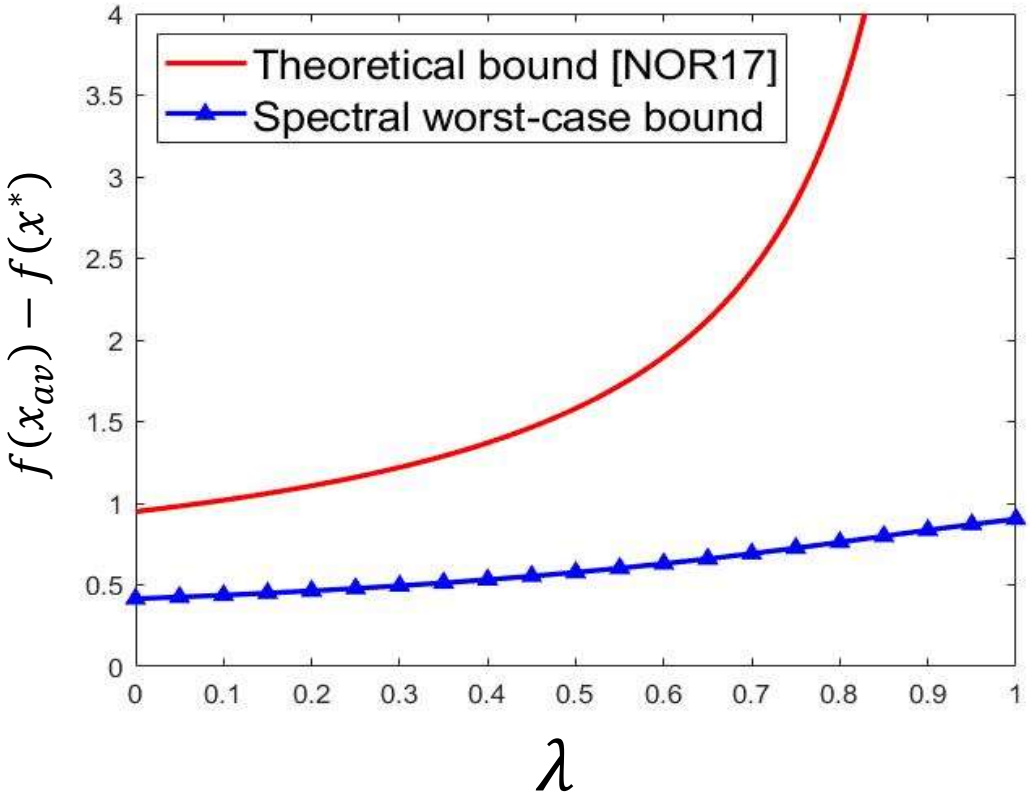
Same **settings** in our experiments, with $R = 1, B = 1$ and $h = 1$.

DGD – Spectral worst-case evolution with N



For $K = 5$ iterations and $\lambda(W) \in [-\lambda, \lambda]$

DGD – Spectral worst-case vs Theoretical bound

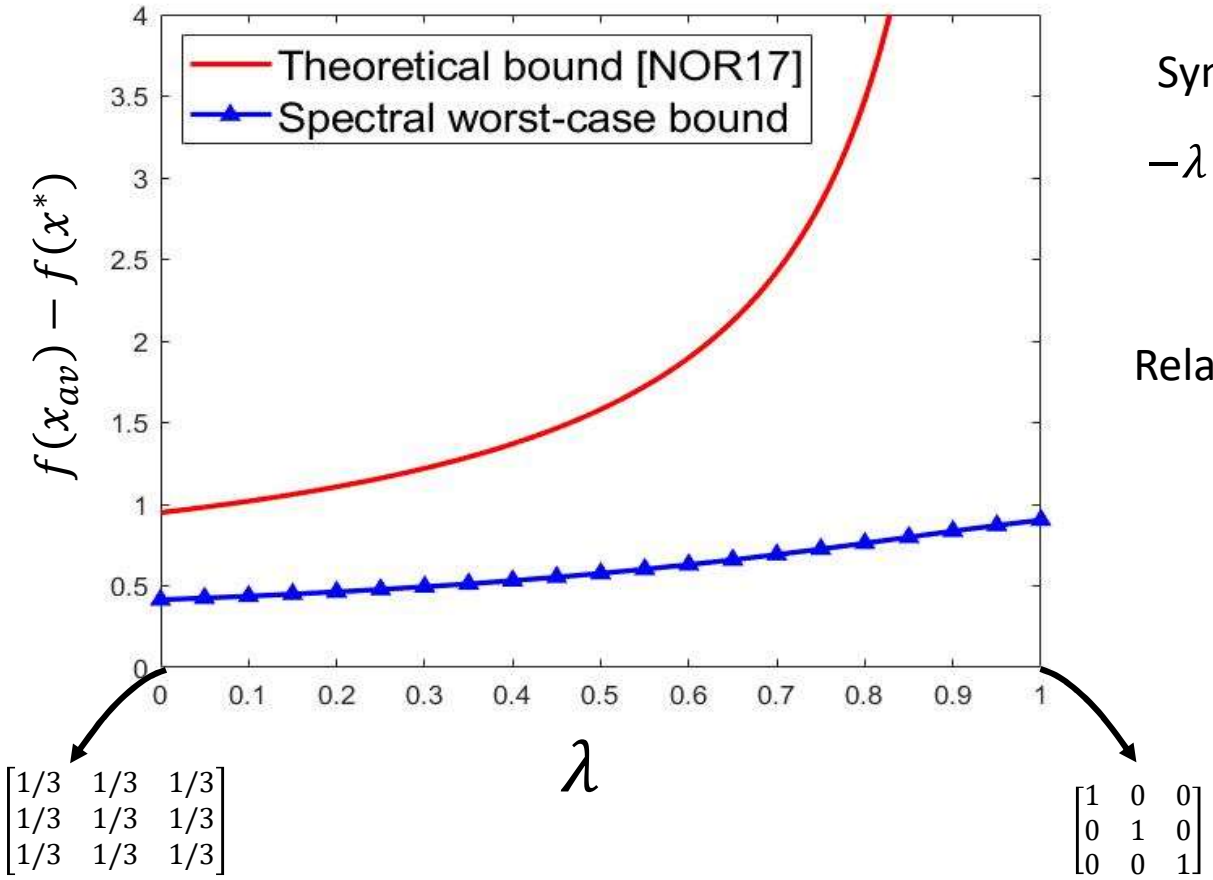


Symmetric range of eigenvalues

$$-\lambda \leq \lambda_n(W) \leq \cdots \leq \lambda_2(W) \leq \lambda$$

For $K = 10$ iterations, $N = 3$ agents.

DGD – Spectral worst-case vs Theoretical bound



Symmetric range of eigenvalues

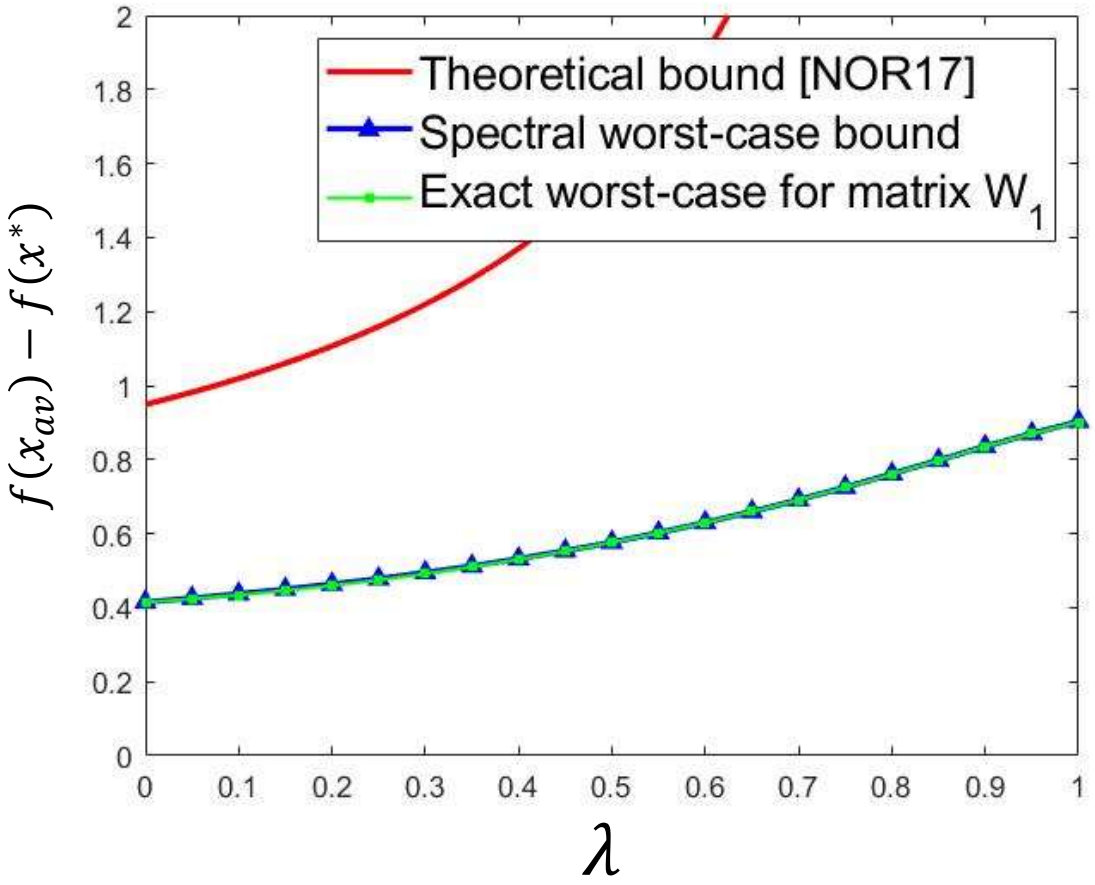
$$-\lambda \leq \lambda_n(W) \leq \cdots \leq \lambda_2(W) \leq \lambda$$

Related to algebraic connectivity σ

$$\lambda = 1 - \sigma$$

For $K = 10$ iterations, $N = 3$ agents.

Is the spectral formulation accurate ?



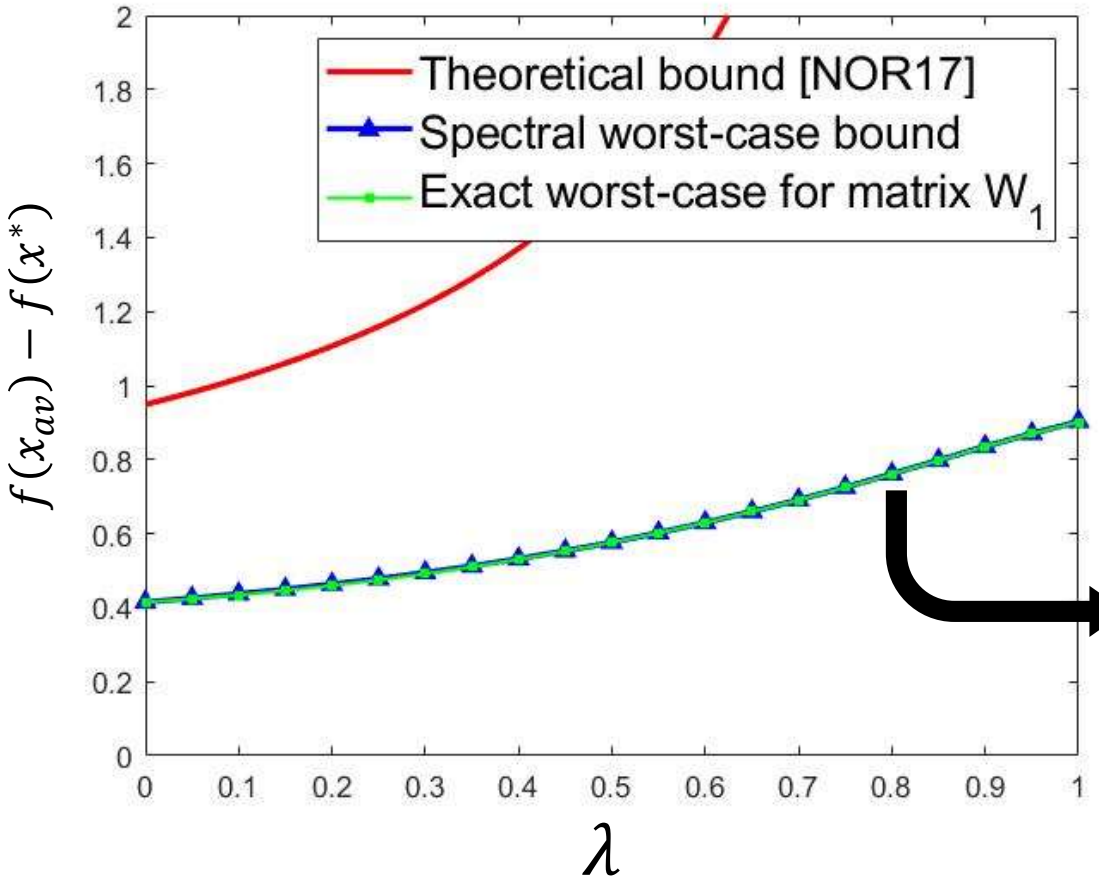
$$W_1 = \begin{bmatrix} d & c & c \\ c & d & c \\ c & c & d \end{bmatrix}$$

with $c = \frac{1+\lambda}{3}$, $d = 1 - 2c$

➡ $\lambda(W_1) = \{1, -\lambda, -\lambda\}$

For $K = 10$ iterations, $N = 3$ agents and $\lambda(W) \in [-\lambda, \lambda]$

Is the spectral formulation accurate ?



$$W_1 = \begin{bmatrix} d & c & c \\ c & d & c \\ c & c & d \end{bmatrix}$$

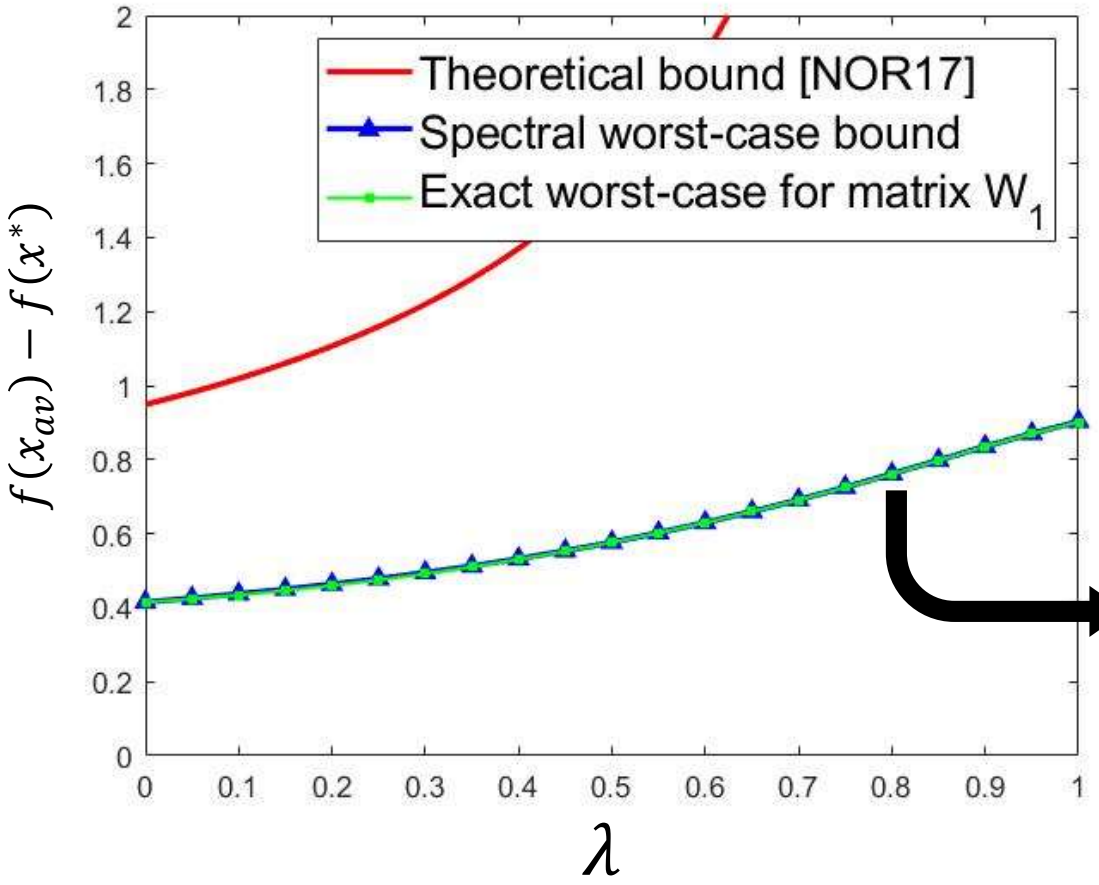
with $c = \frac{1+\lambda}{3}$, $d = 1 - 2c$

➡ $\lambda(W_1) = \{1, -\lambda, -\lambda\}$

EXACT for *spectrally* doubly-stochastic matrices

For $K = 10$ iterations, $N = 3$ agents and $\lambda(W) \in [-\lambda, \lambda]$

Is the spectral formulation accurate ?



$$W_1 = \begin{bmatrix} d & c & c \\ c & d & c \\ c & c & d \end{bmatrix}$$

with $c = \frac{1+\lambda}{3}$, $d = 1 - 2c$

➡ $\lambda(W_1) = \{1, -\lambda, -\lambda\}$

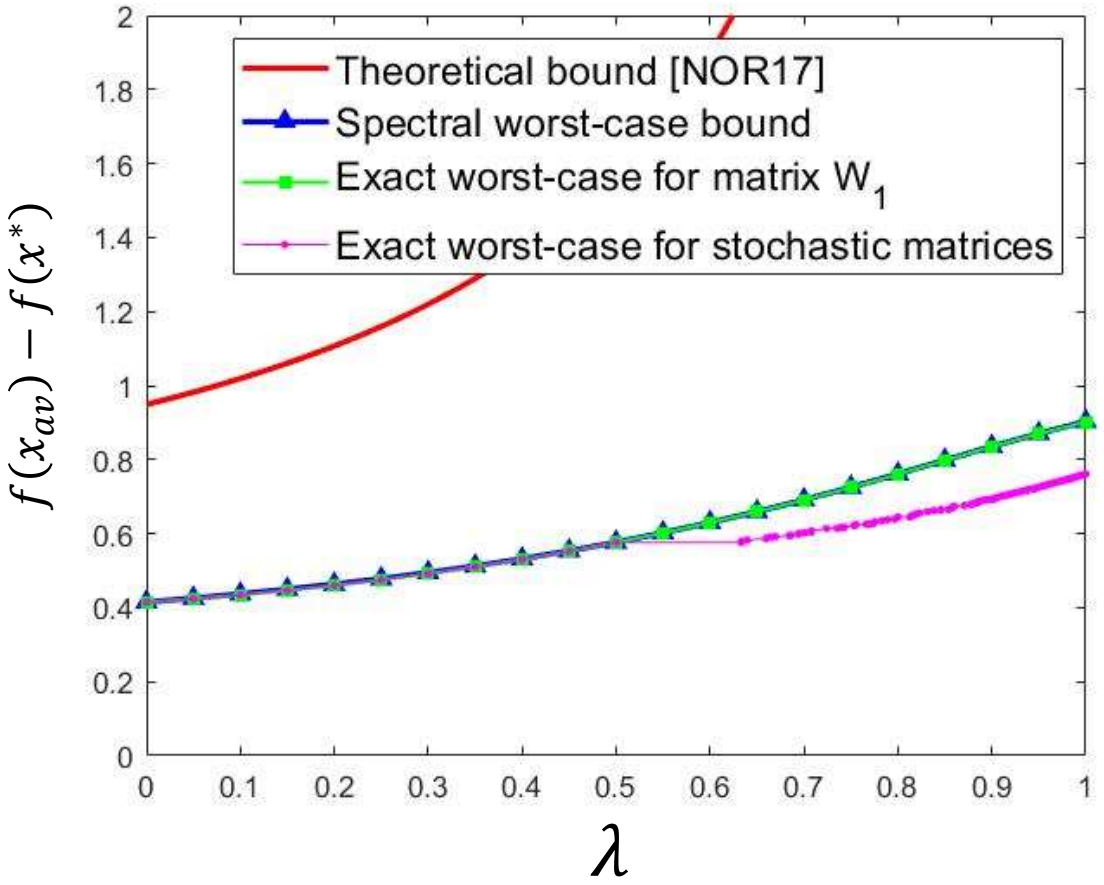
EXACT for *spectrally* doubly-stochastic matrices

Observation:

$W_1 = \widehat{W} = YX^+$

For $K = 10$ iterations, $N = 3$ agents and $\lambda(W) \in [-\lambda, \lambda]$

Is the spectral formulation accurate ?



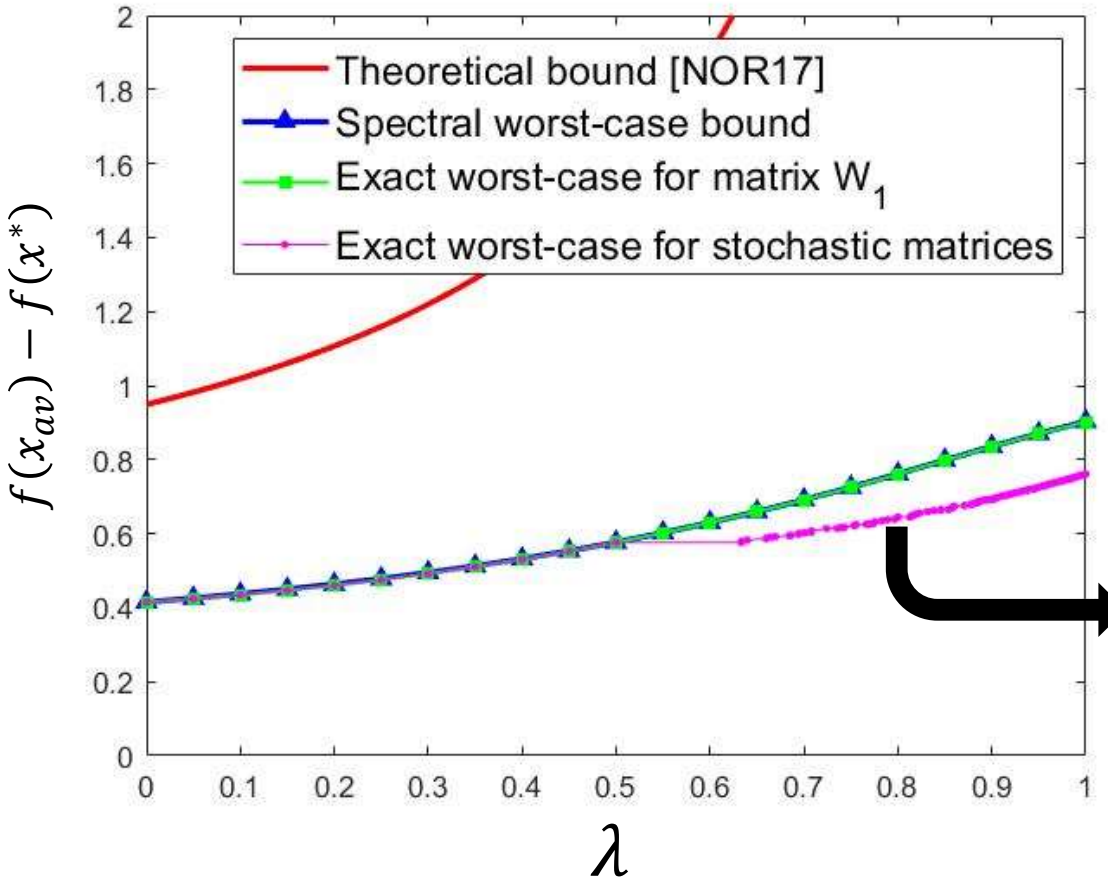
$$W_1 = \begin{bmatrix} d & c & c \\ c & d & c \\ c & c & d \end{bmatrix}$$

with $c = \frac{1+\lambda}{3}$, $d = 1 - 2c$

➡ $\lambda(W_1) = \{1, -\lambda, -\lambda\}$

For $K = 10$ iterations, $N = 3$ agents and $\lambda(W) \in [-\lambda, \lambda]$

Is the spectral formulation accurate ?



$$W_1 = \begin{bmatrix} d & c & c \\ c & d & c \\ c & c & d \end{bmatrix}$$

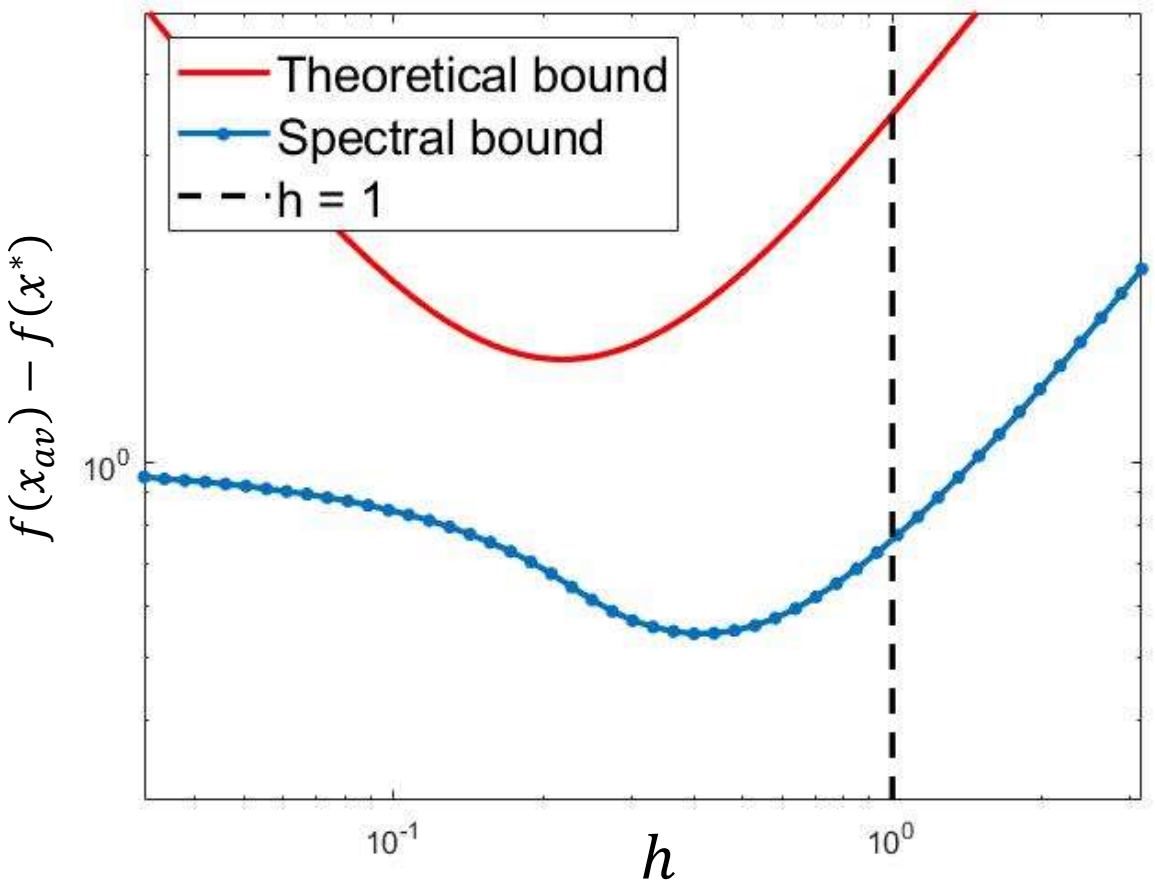
with $c = \frac{1+\lambda}{3}$, $d = 1 - 2c$

➡ $\lambda(W_1) = \{1, -\lambda, -\lambda\}$

ACCURATE for doubly-stochastic matrices

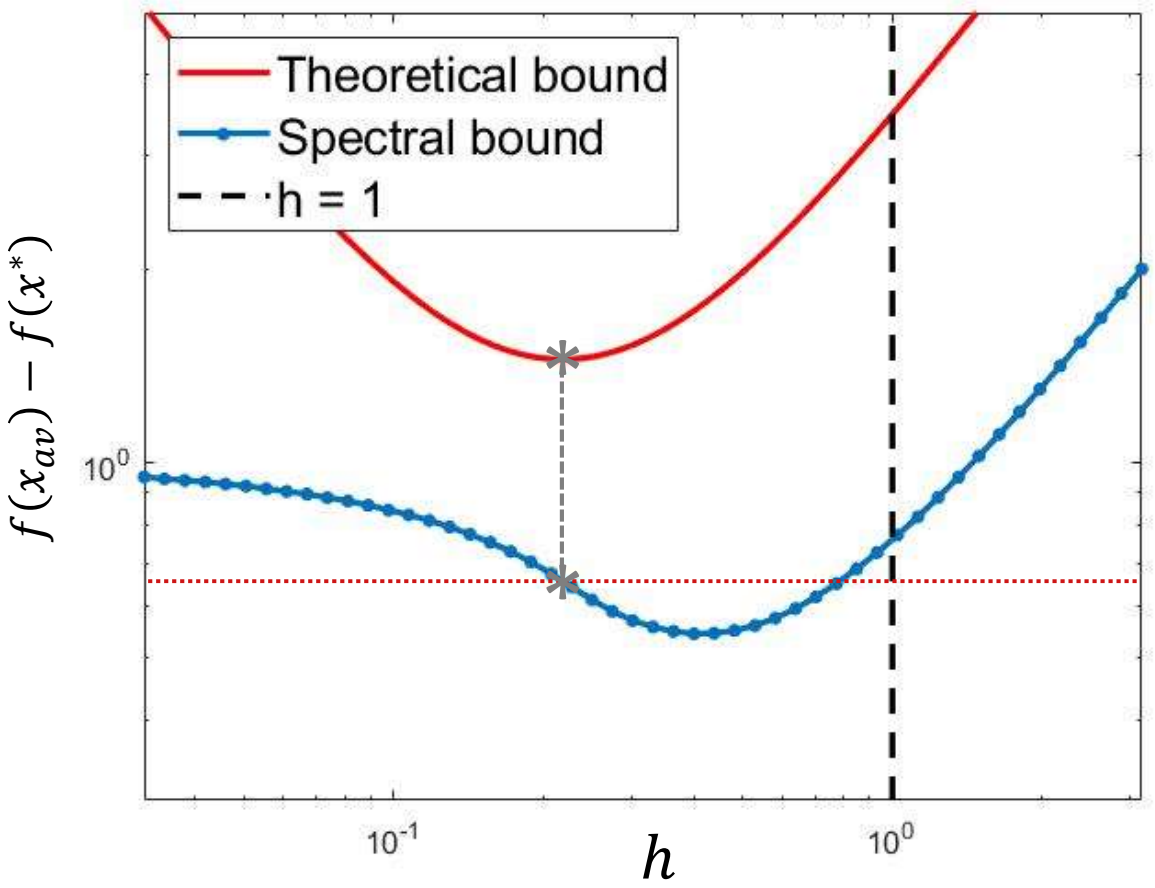
For $K = 10$ iterations, $N = 3$ agents and $\lambda(W) \in [-\lambda, \lambda]$

DGD: Step-Size tuning



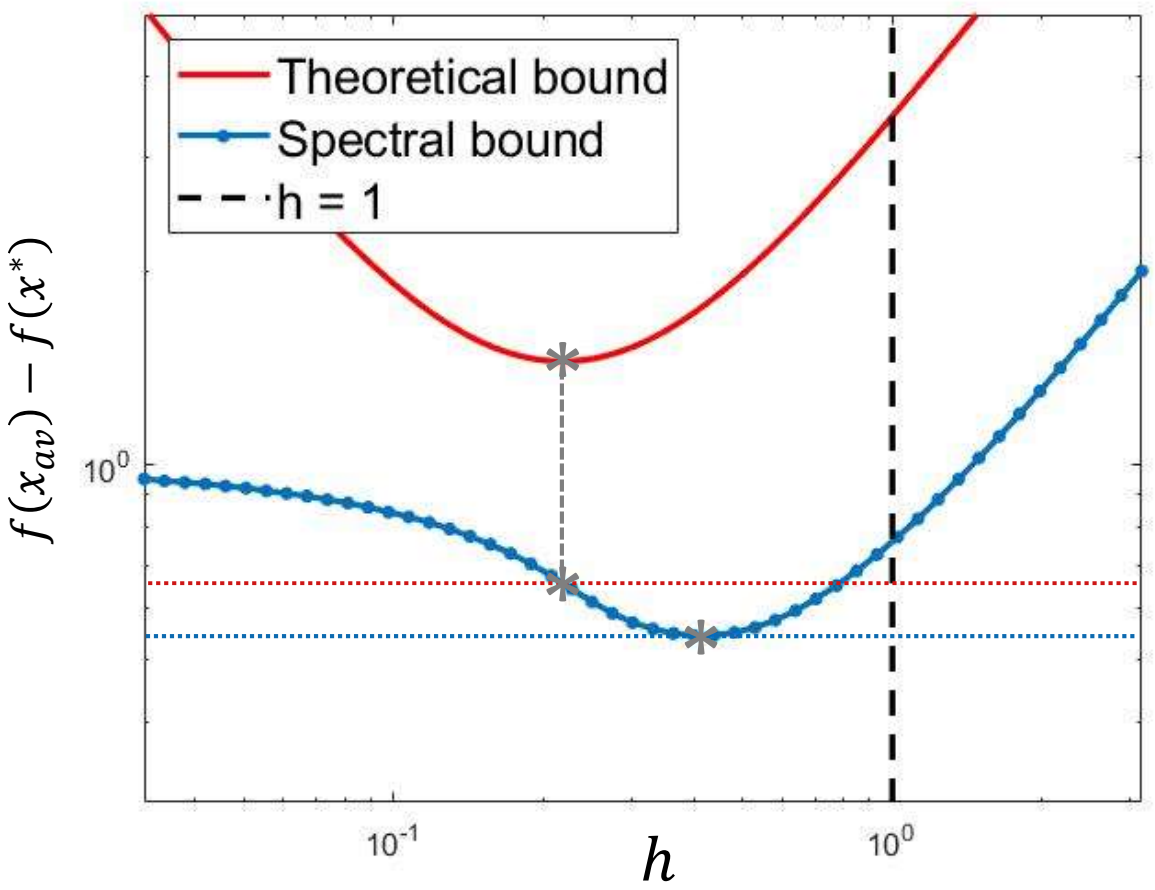
For $K = 10$ iterations, $N = 3$ agents with $\lambda = 0.8$ and $\alpha = \frac{h}{\sqrt{K}}$

DGD: Step-Size tuning



For $K = 10$ iterations, $N = 3$ agents with $\lambda = 0.8$ and $\alpha = \frac{h}{\sqrt{K}}$

DGD: Step-Size tuning

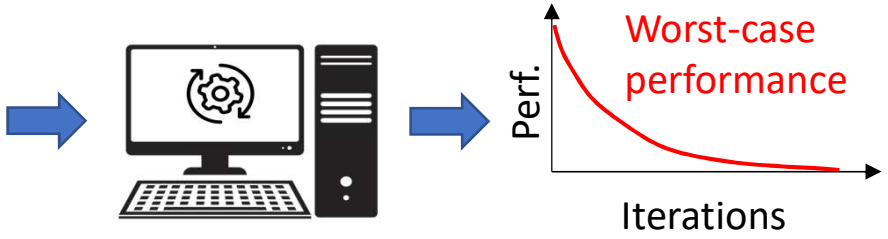


For $K = 10$ iterations, $N = 3$ agents with $\lambda = 0.8$ and $\alpha = \frac{h}{\sqrt{K}}$

Conclusion

Automatic tool for accurate **performance estimation** of decentralized optimization methods

```
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
```

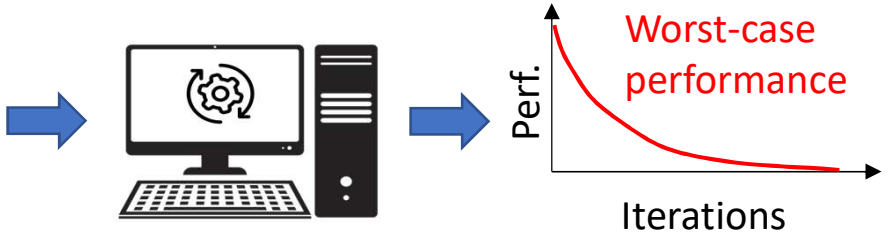


PEP idea: worst-cases are solutions of optimization problems

Conclusion

Automatic tool for accurate **performance estimation** of decentralized optimization methods

```
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
```



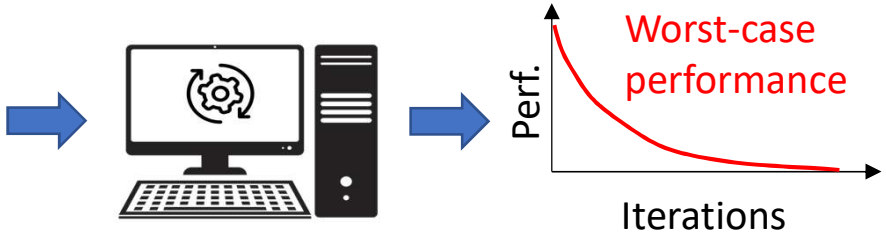
PEP idea: worst-cases are solutions of optimization problems

SPECTRAL formulation	EXACT formulation
Spectral class of matrices	Given network matrix W
Relaxation of PEP	ALWAYS exact

Conclusion

Automatic tool for accurate **performance estimation** of decentralized optimization methods

```
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
```



PEP idea: worst-cases are solutions of optimization problems

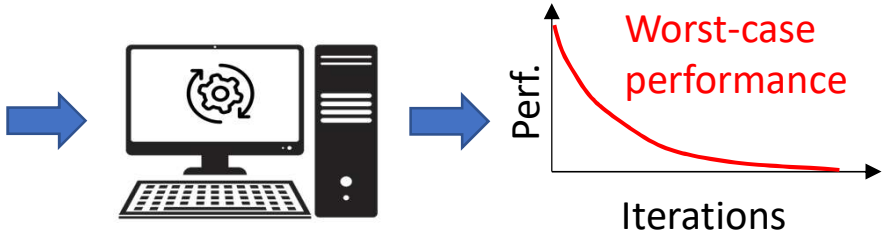
SPECTRAL formulation	EXACT formulation
Spectral class of matrices	Given network matrix W
Relaxation of PEP	ALWAYS exact

For DGD:
- ✓ Independent of N
- ✓ Tight with potential negative weights
- ✓ Accurate with nonnegative weights
- ✓ Improve on the literature bound

Conclusion

Automatic tool for accurate **performance estimation** of decentralized optimization methods

```
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
```



PEP idea: worst-cases are solutions of optimization problems

Future works

- ❑ Implementation of the formulation in **PESTO toolbox** (in progress)
- ❑ Strong theoretical understanding of our formulation
- ❑ Analyze **other decentralized algorithms** using our tool

References

[CH21] S. Colla, J. M. Hendrickx, “Automated Worst-Case Performance Analysis of Decentralized Gradient Descent”, 2021.

[Taylor17] A. B. Taylor, “Convex interpolation and performance estimation of first-order methods for convex optimization”, PhD, UCLouvain, Louvain-la-Neuve, Belgium, 2017.

[NOR17] A. Nedic, A. Olshevsky, and M. G. Rabbat, “Network topology and communication computation tradeoffs in decentralized optimization”, 2017.