Automatic Performance Estimation for Decentralized Optimization

Sebastien Colla, Julien M. Hendrickx

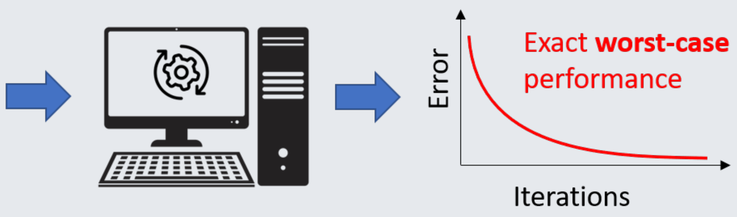
ICTEAM, UCLouvain, Belgium

Goal

Automatic computation of the worst-case **performance** of any decentralized optimization method.

```
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
```



- Better understanding, tuning and comparison of methods,
- Rapid exploration of new methods.

Performance Estimation Problem (PEP)

Computing worst-case performances as optimization problems

$$\sup_{\{f_i\},\{x_i^k\},W} \quad \mathcal{P}\big(\{f_i\},\{x_i^k\},W\big)$$
$$\text{s.t.} \quad f_i \in \mathcal{F},$$
$$x^0 \in \mathcal{X}_0,$$
$$x^k \text{ from method } \mathcal{M},$$
$$W \in \mathcal{W}.$$

- $\mathcal{P}$ is a performance criterion, e.g. $f(x^K) - f(x^*)$.
- $\mathcal{F}$ is a class of functions, e.g. $\mathcal{F}_{\mu,L}$.
- $\mathcal{X}_0$ is a valid set for initial points.
- $\mathcal{W}$ is a class of averaging matrices.

Discretization: optimize over $\{x_i^k, g_i^k, f_i^k\}$ such that these are consistent with the above constraints where $g_i^k = \nabla f_i(x_i^k)$ and $f_i^k = f_i(x_i^k)$.

Replace $f_i \in \mathcal{F}$ by appropriate **interpolation constraints**.
Non-linearity can be addressed using an **SDP reformulation**.

- **Exact results** when a specific matrix W is given, i.e. $W \in \{W_e\}$.

Easy PEP formulation with the *Performance-Estimation-Toolbox*.

Spectral classes of averaging matrices in PEP [1]

Consensus steps: $y_i^k = \sum_{j=1}^N W_{ij} x_j^k$ for all k, or $Y = WX$, where

- Y and X are $N \times K$ matrices of variables:

$$X = \begin{bmatrix} x_1^1 & \dots & x_1^K \\ \vdots & & \vdots \\ x_N^1 & \dots & x_N^K \end{bmatrix}, \quad Y = \begin{bmatrix} y_1^1 & \dots & y_1^K \\ \vdots & & \vdots \\ y_N^1 & \dots & y_N^K \end{bmatrix}$$

- W is a $N \times N$ symmetric stochastic matrix with eigenvalues in $[\lambda^-, \lambda^+]$, except for $\lambda_1 = 1$, i.e.

$$\lambda^- \leq \lambda_N(W) \leq \dots \leq \lambda_2(W) \leq \lambda^+ \qquad \text{where } \lambda^-, \lambda^+ \in [-1, 1].$$

Decoupling the consensus part from disagreement part:

$$X = \mathbf{1}\overline{X}^T + X_\perp, \quad Y = \mathbf{1}\overline{Y}^T + Y_\perp,$$

where $\overline{X} = \frac{1}{N}\mathbf{1}^T X$, $\overline{Y} = \frac{1}{N}\mathbf{1}^T Y$ are agents average vectors in $\mathbb{R}^K$.

Necessary constraints for consensus steps

If $Y = WX$, with W symmetric, stochastic and $\lambda(W) \in [\lambda^-, \lambda^+]$, then

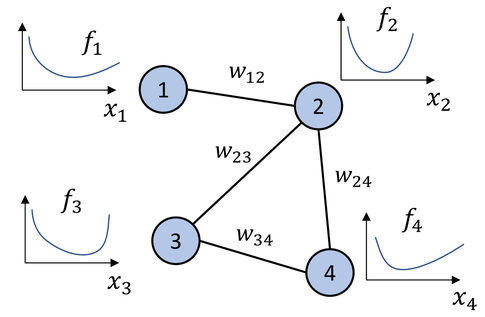
(i) $X^T Y$ and $X_\perp^T Y_\perp$ are symmetric,
(ii) The agent average is preserved: $\overline{X} = \overline{Y}$
(iii) $\lambda^- X_\perp^T X_\perp \preceq X_\perp^T Y_\perp \preceq \lambda^+ X_\perp^T X_\perp$,
(iv) $(Y_\perp - \lambda^- X_\perp)^T (Y_\perp - \lambda^+ X_\perp) \preceq 0$.

Relax *constraint* $Y = WX$ *in PEP and add these.*

Decentralized Optimization

Consider a set of agents $\{1, \dots, N\}$, holding each a local function f_i and working together to solve the following optimization problem:

$$\min_{x \in \mathbb{R}^d} \quad f(x) = \frac{1}{N}\sum_{i=1}^N f_i(x),$$
$$\Updownarrow$$
$$\min_{x_1, \dots, x_N} \quad \frac{1}{N}\sum_{i=1}^N f_i(x_i),$$
$$\text{s.t.} \quad x_i = x_j \quad \text{for all } (i,j).$$



Example of algorithm: The Distributed Gradient Descent (DGD):

$$\text{Consensus Step:} \qquad y_i^k = \sum_{j=1}^N W_{ij}\, x_j^k,$$
$$\text{Local Gradient Step:} \qquad x_i^{k+1} = y_i^k - \alpha \nabla f_i(x_i^k),$$

where $\alpha > 0$ is a constant step-size and $W \in \mathbb{R}^{N \times N}$ is an averaging matrix. These could also vary at each iteration k.

Application to the analysis of algorithms [1]

- **DGD**, K steps with $\alpha = \frac{1}{\sqrt{K}}$, W such that $\lambda_j(W) \in [-\lambda, \lambda]$ for $j = 2, \dots, N$ and f_i convex with bounded subgradients.

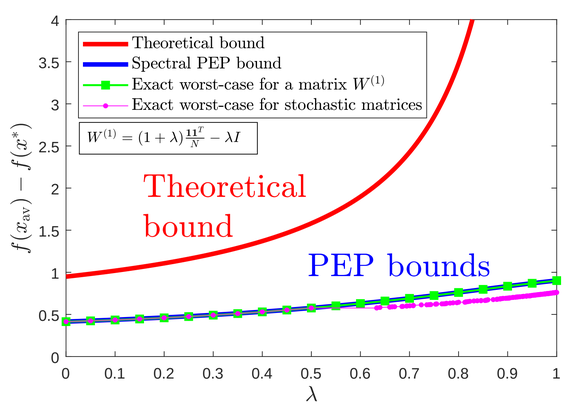


Figure 1: Tightness analysis of the PEP bound for 10 iterations DGD. ($N = 3$)

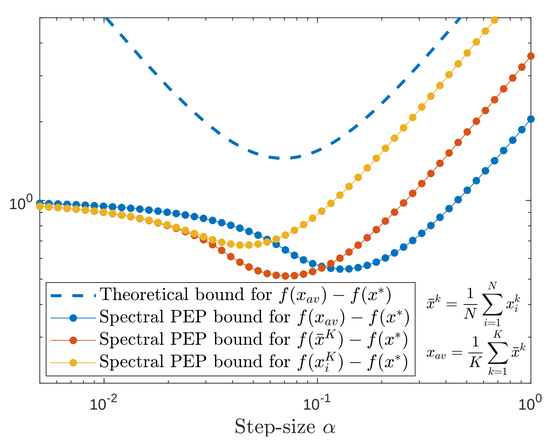


Figure 2: Tuning of α for 10 iterations of DGD and $\lambda = 0.8$, with different performance criterions. ($N = 3$)

- **DIGing** with $\lambda_j(W) \in [-\lambda, \lambda]$ and f_i L-smooth and μ-strongly convex.

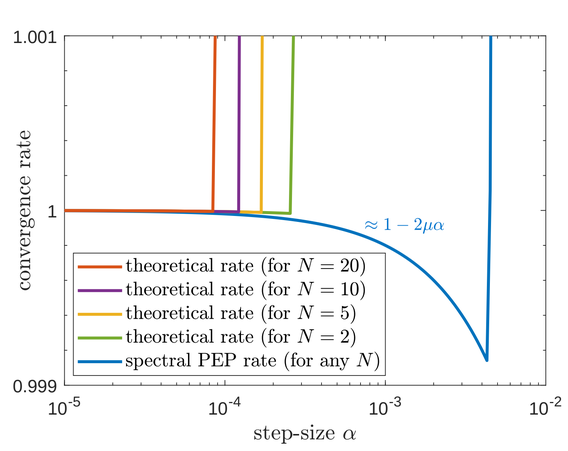


Figure 3: Tuning of α for 10 iterations of DIGing and $\lambda = 0.9$, in comparison with the theoretical bound.

Findings and observations

The PEP bound for DGD is

- presenting a large gap with respect to the theory.
- Independent of N.
- tight for generalized* doubly-stochastic matrices.

*allowing negative elements.

- available for many performance criterions.
- useful for tuning the step-size.

We can answer a large diversity of (new) questions !

Findings and observations

- Tuning α in DIGing, based on the PEP bound, improves its convergence rate guarantee by orders of magnitude.
- Improvement scales with N.

References

[1] S. Colla and J. M. Hendrickx. "Automatic Performance Estimation for Decentralized Optimization". In: *preprint* (2022).