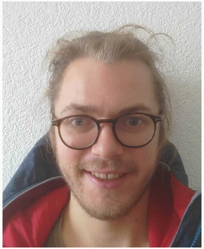
# Byzantine Robust Optimization: A dual approach

Renaud Gaucher

12/03/2024

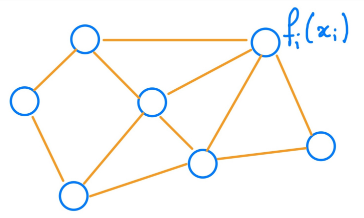


**Hadrien Hendrikx**
*TOTH*
*INRIA Grenoble*

**Aymeric Dieuleveut**
*CMAP*
*École Polytechnique*

# Decentralized Optimization

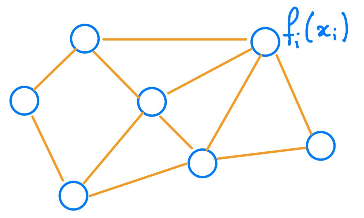
- $n$ computing units in a network with
  - Local cost functions;
  - Local memory.



# Decentralized Optimization

- $n$ computing units in a network with
  - Local cost functions;
  - Local memory.
- Cooperate to find a global solution:

$$x^* \in \mathrm{argmin}_{x \in \mathbb{R}^d} \left\{ f(x) := \sum_{i=1}^{n} f_i(x) \right\}.$$

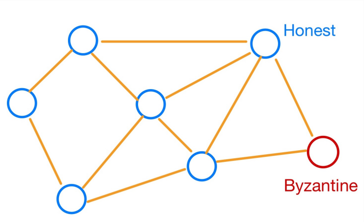


# Decentralized Optimization under Byzantine corruption

- $n$ computing units in a network with
  - Local cost functions;
  - Local memory.
- Units cooperate to find a global solution:

$$x^* \in \mathrm{argmin}_{x\in\mathbb{R}^d} \left\{ f_h(x) := \sum_{i \text{ honest}} f_i(x) \right\}.$$

- Some unknown units are *Byzantine*, i.e malicious and omniscient.



# Duality in Decentralized optimization

Idea: Decentralized optimization = optimization under constraint !

# Duality in Decentralized optimization

Idea: Decentralized optimization = optimization under constraint !

1. Each unit has a local parameter $X_i$;

# Duality in Decentralized optimization

Idea: Decentralized optimization = optimization under constraint !

1. Each unit has a local parameter $X_i$;
2. Objective function writes $F(X_1, ..., X_n) = \sum_i f_i(X_i)$;

# Duality in Decentralized optimization

Idea: Decentralized optimization = optimization under constraint !

1. Each unit has a local parameter $X_i$;
2. Objective function writes $F(X_1, ..., X_n) = \sum_i f_i(X_i)$;
3. Each edge is an equality constraint between two neighbors: $C[X_1, ..., X_n] = [X_i - X_j]_{(i \sim j)}$.

# Duality in Decentralized optimization

Idea: Decentralized optimization = optimization under constraint !

1. Each unit has a local parameter $X_i$;
2. Objective function writes $F(X_1, ..., X_n) = \sum_i f_i(X_i)$;
3. Each edge is an equality constraint between two neighbors: $C[X_1, ..., X_n] = [X_i - X_j]_{(i\sim j)}$.

Primal problem is equivalent to a dual problem on the edges:

$$\min_{CX=0} F(X)$$

# Duality in Decentralized optimization

Idea: Decentralized optimization = optimization under constraint !

1. Each unit has a local parameter $X_i$;
2. Objective function writes $F(X_1, ..., X_n) = \sum_i f_i(X_i)$;
3. Each edge is an equality constraint between two neighbors: $C[X_1, ..., X_n] = [X_i - X_j]_{(i \sim j)}$.

Primal problem is equivalent to a dual problem on the edges:

$$\min_{CX=0} F(X) \iff \min_{\Lambda \in \mathbb{R}^{edges \times d}} F^*(C^T \Lambda).$$

# Duality in Decentralized optimization

Idea: Decentralized optimization = optimization under constraint !

1. Each unit has a local parameter $X_i$;
2. Objective function writes $F(X_1, ..., X_n) = \sum_i f_i(X_i)$;
3. Each edge is an equality constraint between two neighbors:
   $C[X_1, ..., X_n] = [X_i - X_j]_{(i\sim j)}$.

Primal problem is equivalent to a dual problem on the edges:

$$\min_{CX=0} F(X) \iff \min_{\Lambda \in \mathbb{R}^{edges \times d}} F^*(C^T\Lambda).$$

Solved using Gradient Descent.

# Gossip (dual) algorithms

$\Longrightarrow$ **Duality gives practical (and efficient) decentralized algorithms.**

*For each units $i$:*

1. Compute a gradient,

# Gossip (dual) algorithms

$\Longrightarrow$ **Duality gives practical (and efficient) decentralized algorithms.**

*For each units $i$:*

1. Compute a gradient,
2. Share it with neighbors,

$\Longrightarrow$ **Duality gives practical (and efficient) decentralized algorithms.**

*For each units $i$:*

1. Compute a gradient,
2. Share it with neighbors,
3. **Average** messages received,

# Gossip (dual) algorithms

$\Longrightarrow$ **Duality gives practical (and efficient) decentralized algorithms.**

*For each units $i$:*

1. Compute a gradient,
2. Share it with neighbors,
3. **Average** messages received,
4. Actualize your parameter using it.

**Why using Lagrangian duality in a Byzantine setting?**

**Why using Lagrangian duality in a Byzantine setting?**

$$\min_{\Lambda \in \mathbb{R}^{edges \times d}} F^*(C^T \Lambda).$$

1. Each row of $\Lambda$ translate the interactions between two neighbors through an edge.

**Why using Lagrangian duality in a Byzantine setting?**

$$\min_{\Lambda \in \mathbb{R}^{edges \times d}} F^*(C^T \Lambda).$$

1. Each row of $\Lambda$ translate the interactions between two neighbors through an edge.
2. Controlling the dual variable $\Lambda$ means controlling how much units influence each others.

**Why using Lagrangian duality in a Byzantine setting?**

$$\min_{\Lambda \in \mathbb{R}^{edges \times d}} F^*(C^T \Lambda).$$

1. Each row of $\Lambda$ translate the interactions between two neighbors through an edge.
2. Controlling the dual variable $\Lambda$ means controlling how much units influence each others.

$\Rightarrow$ lever for limiting influence of Byzantines units

Questions ?

# Appendix

# Gossip dual algorithm

*For each units $i$:*
*Initialize $y_i^0 = 0$*

1. compute $x_i^t := \nabla f_i^*(y_i^t)$,
2. share $x_i^t$ with his neighbors,
3. actualization of $y_i^t$:
$$y_i^{t+1} := y_i^t - \eta \sum_{j \sim i} \left( x_i^t - x_j^t \right) .$$

$\implies$ **Duality gives practical decentralized algorithms !**