Redundancy and Resilience in Distributed Optimization

Shuo Liu

Georgetown University

Optimization

- Finding *the least cost* solution

- Find $\arg\min f(x) \qquad x \in X$

Optimization

- Finding *the least cost* solution

- Find $\arg\min f(x) \qquad x \in X$

Optimization

- Finding *the least cost* solution

- Find $\arg\min f(x) \qquad x \in X$

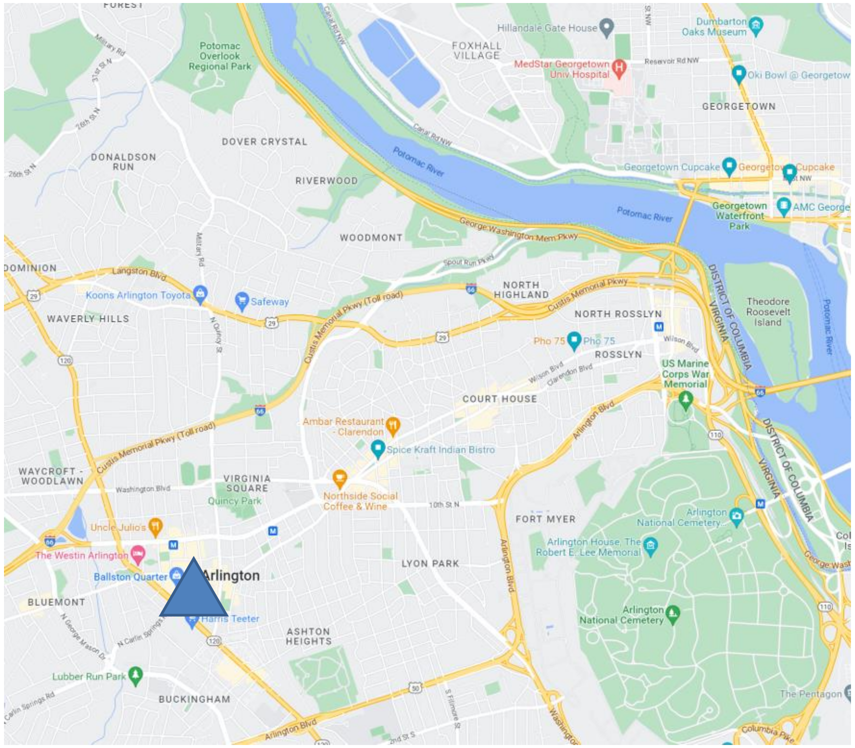
Optimization

- Finding *the least cost* solution

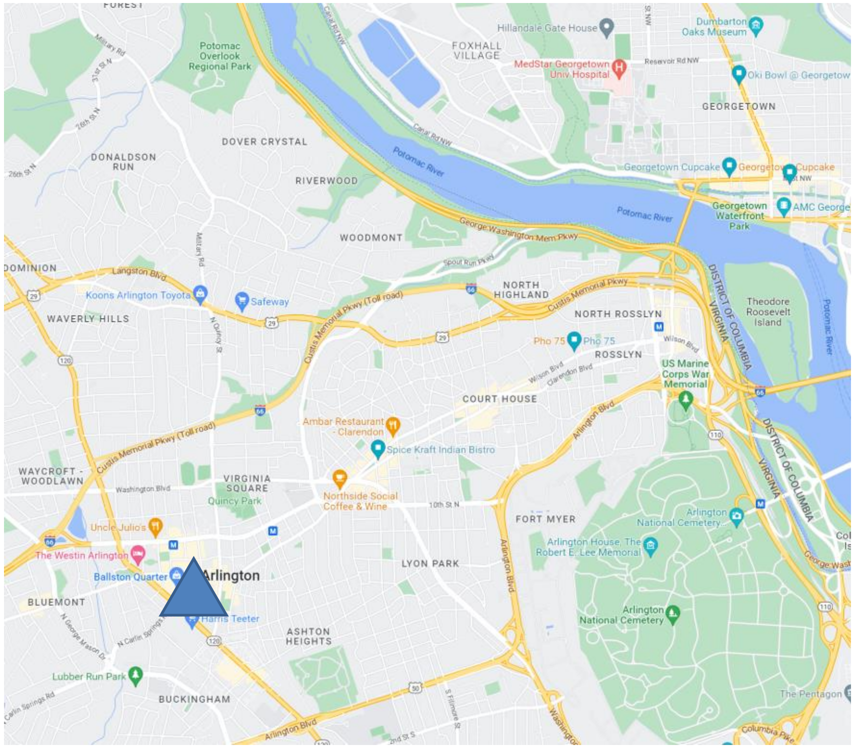
- Find $\arg\min f(x) \qquad x \in X$

- Many applications: ML, linear programming ...

Optimization

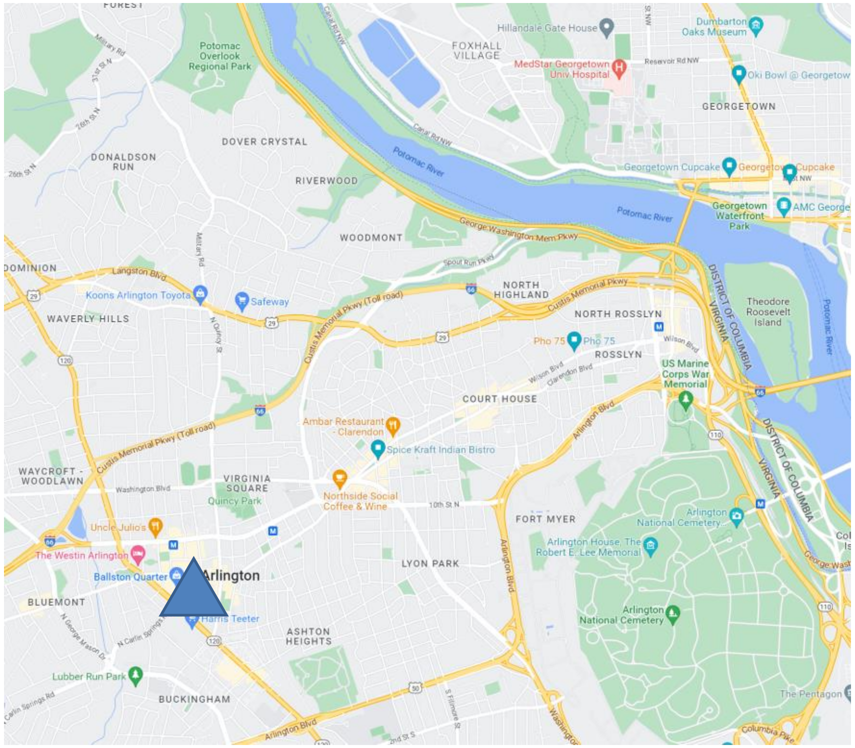


Optimization



Where cost is *minimum*

Optimization



Cost
Money, fuel, energy…
or adding them together

Where cost is *minimum*

Distributed optimization

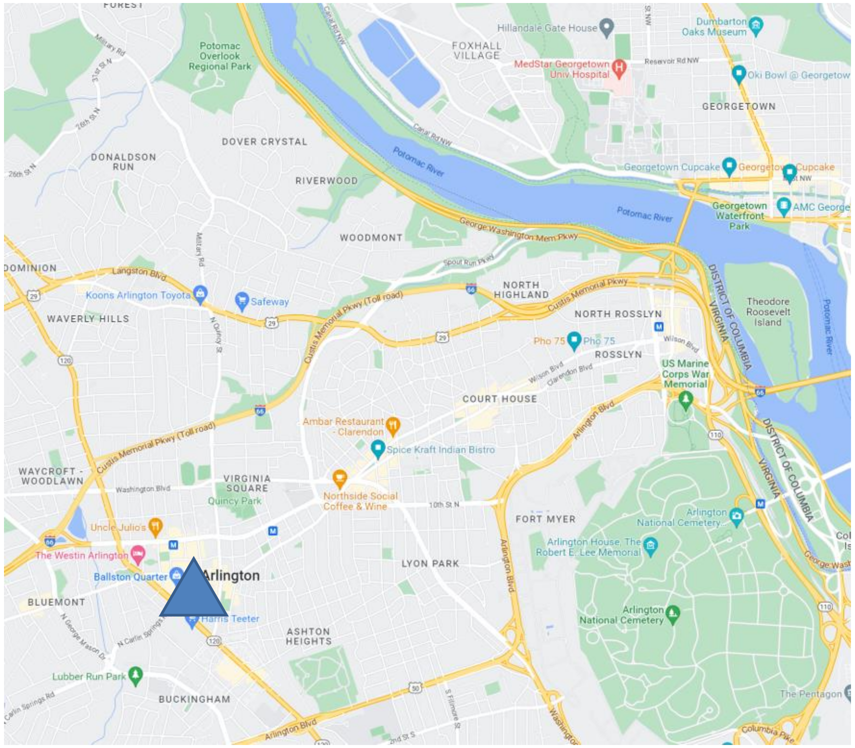
- n agents, each agent i has $f_i(x)$

- Find $\arg\min \sum_i f_i(x)$

Distributed optimization

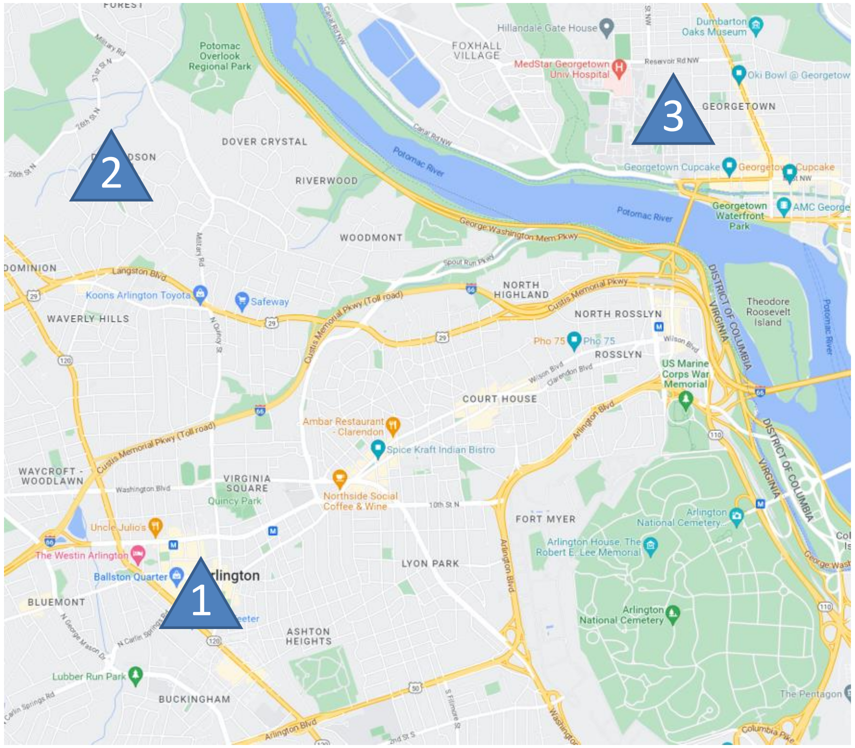
- n agent, each agent i has $f_i(x)$
- Find $\arg\min \sum_i f_i(x)$
- Without sending the whole f_i's

Distributed optimization



Where cost is *minimum*

Distributed optimization



Where *aggregate* cost is minimum

Application to machine learning

- Each agent has a local dataset

Application to machine learning

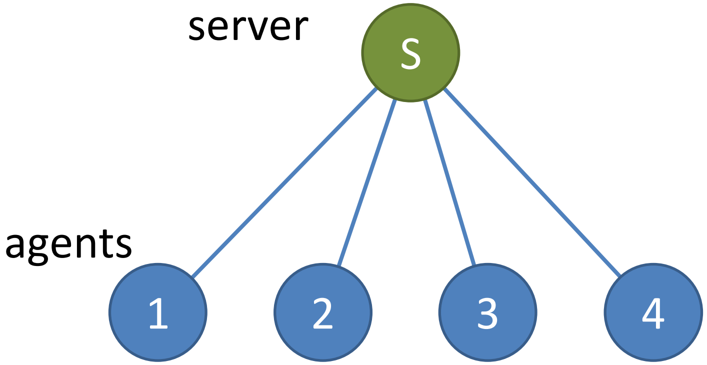
- Each agent has a local dataset
- **Agent's cost function:** Loss corresponding to its dataset

Application to machine learning

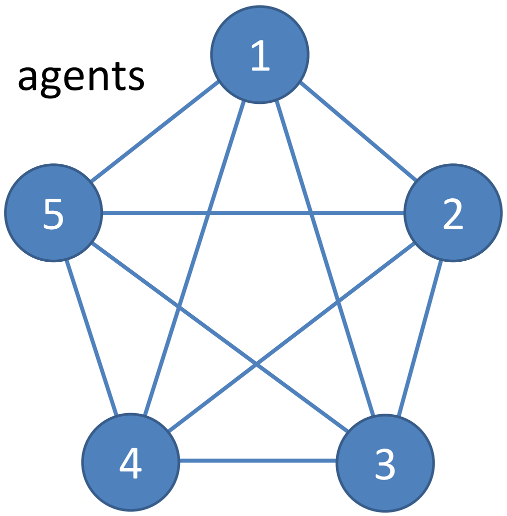
- Each agent has a local dataset
- **Agent's cost function:** Loss corresponding to its dataset
- **Goal:** Minimize the aggregate cost → train model

Distributed optimization

- Two common architectures



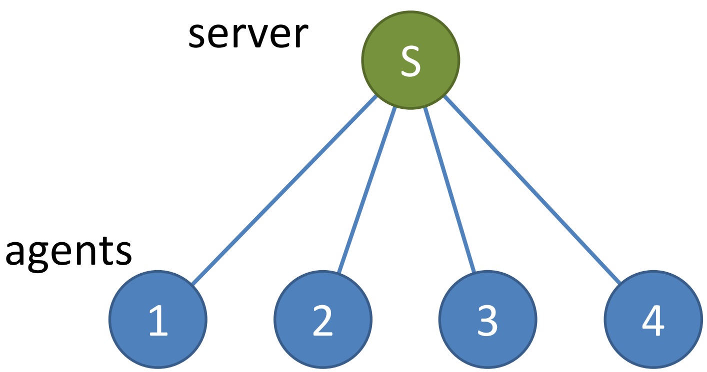
server-based



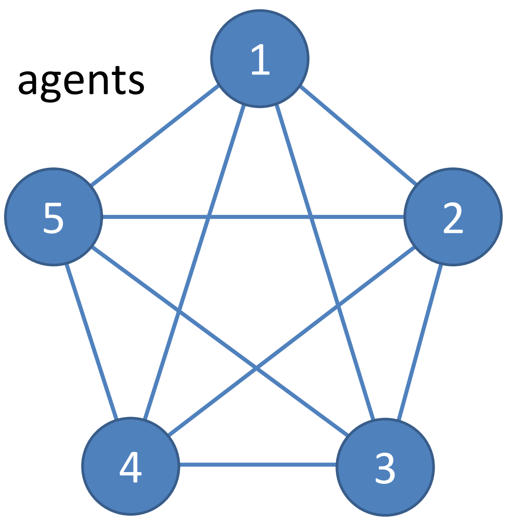
decentralized

Distributed optimization

- Two common architectures



server-based

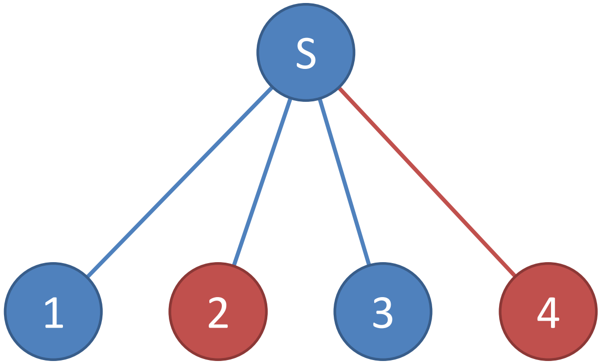


decentralized

- We focus on **server**-based architecture

Resilient distributed optimization

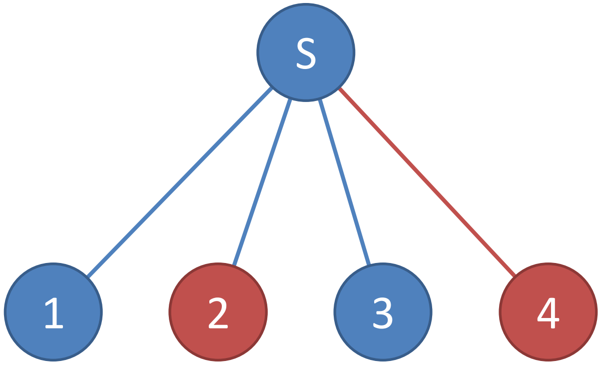
- Issues happen in practice
  - Faulty agents
  - Slow agents (stragglers)



Faulty agents

Fault models

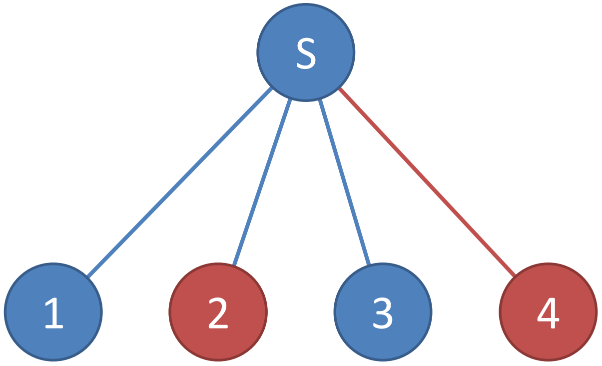
- Crash failures
- Byzantine model



Fault models

- Crash failures

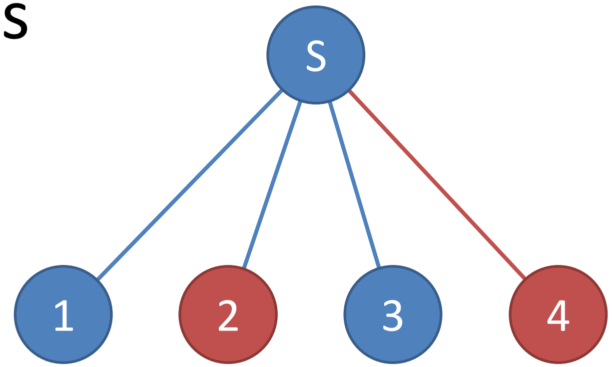
- **Byzantine model**
  - Software errors
  - Adversarial attacks



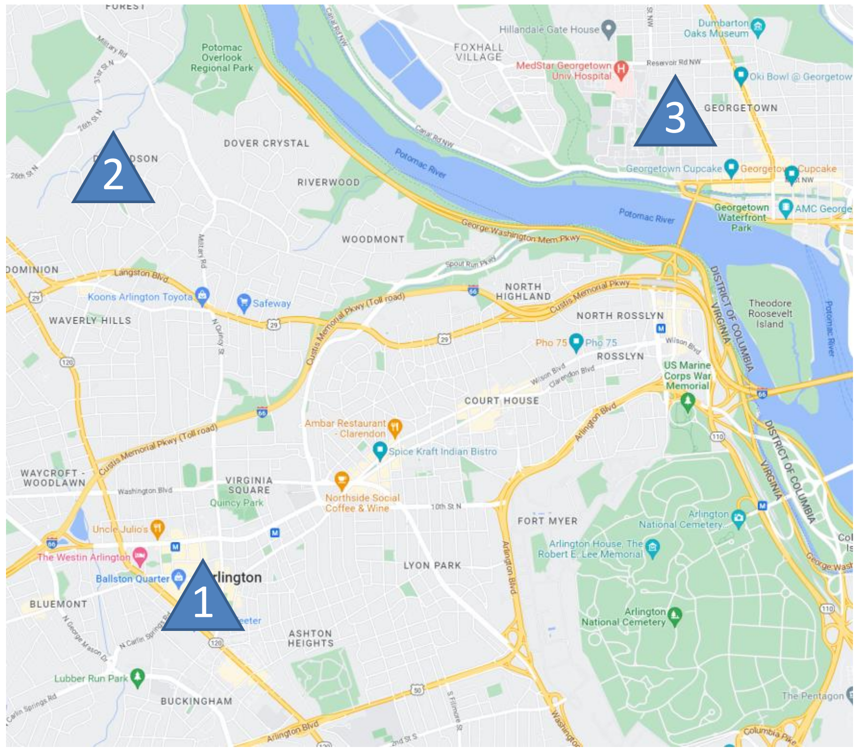
Byzantine model

- Make no assumption on agent behavior
- Cap the number of faulty agents

An algorithm that tolerates f faulty agents

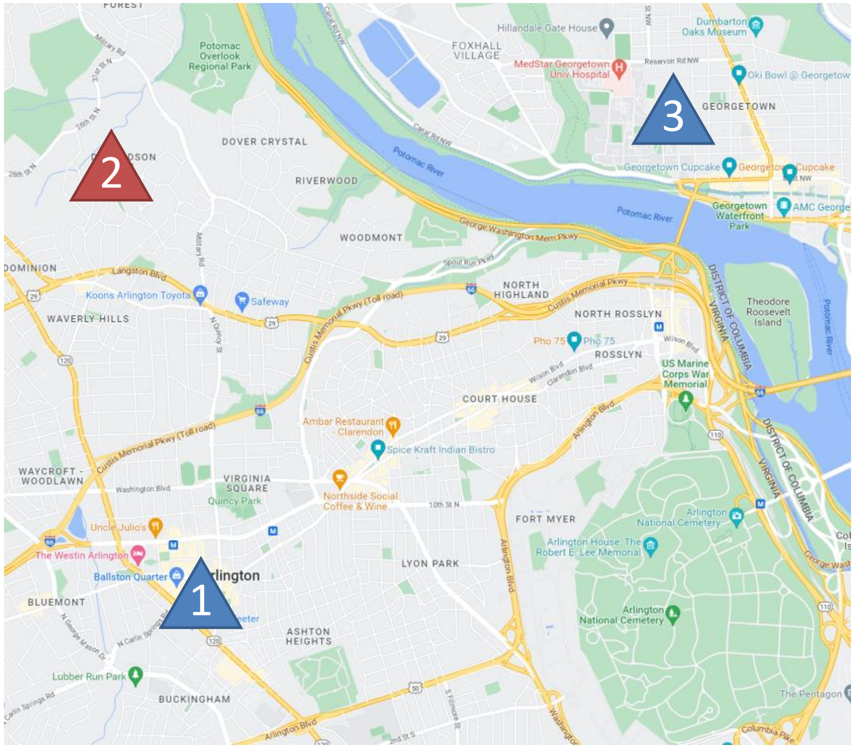


Impact of faulty agents



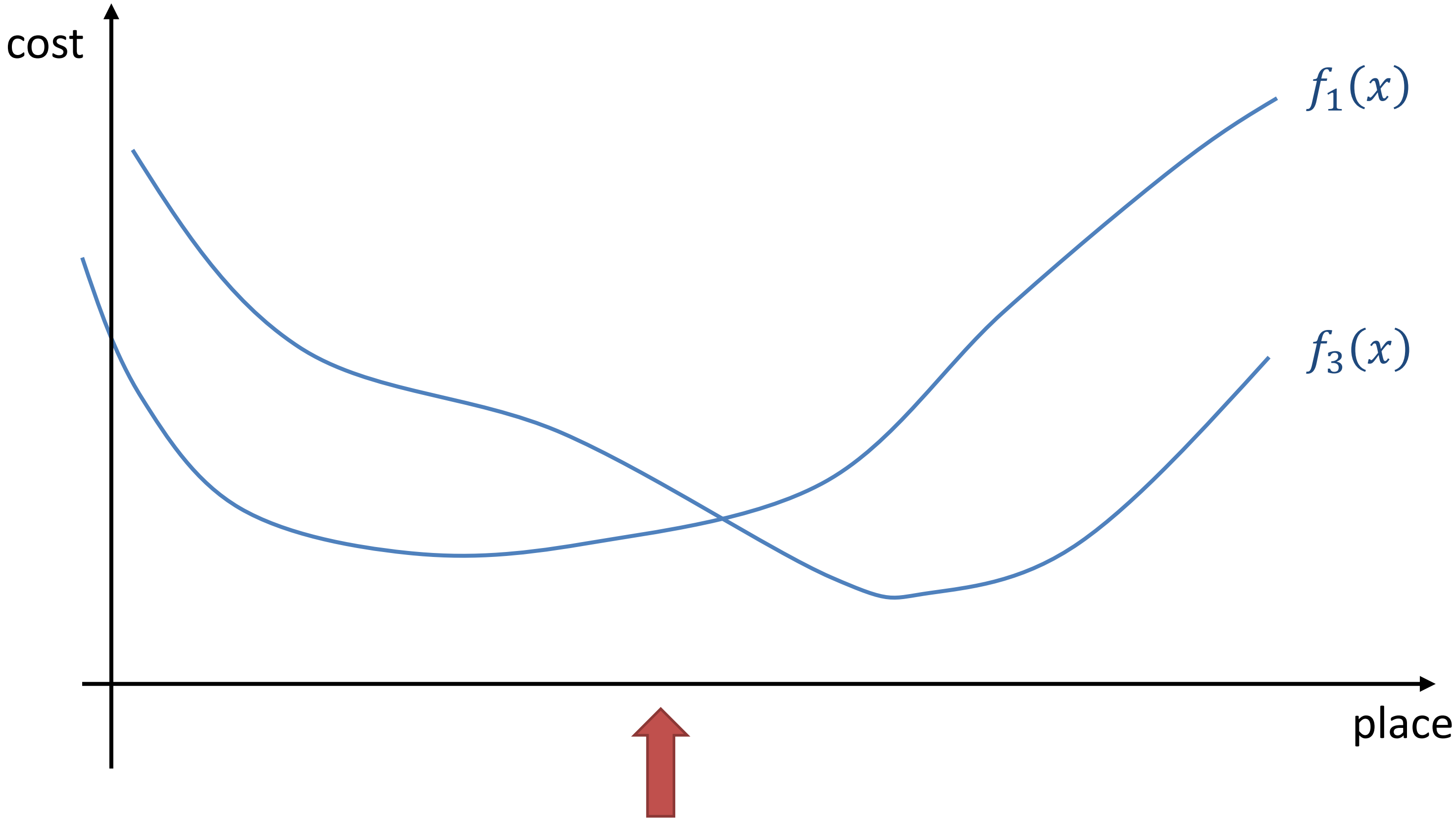
Where *aggregate* cost is minimum

Impact of faulty agents



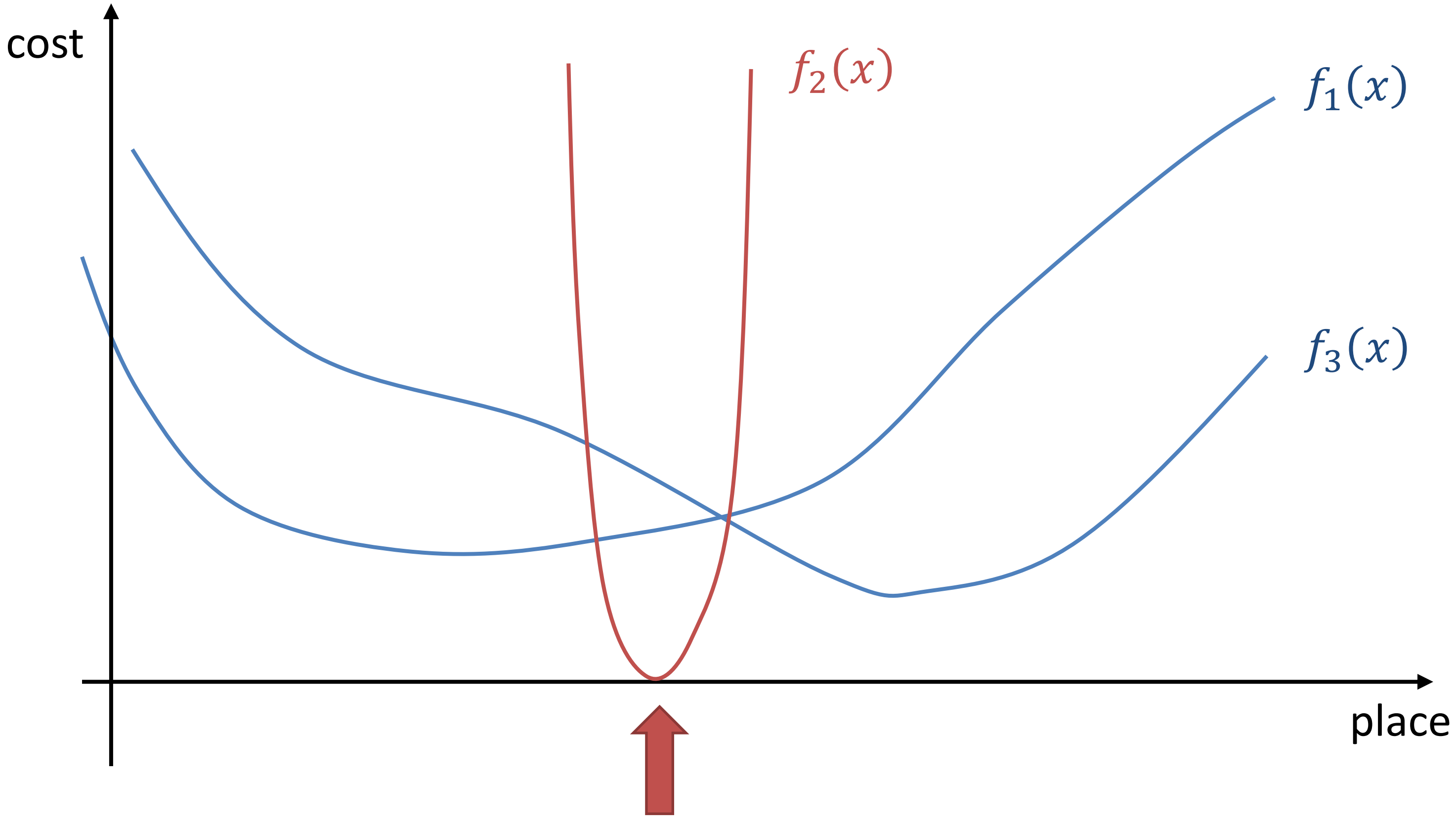
“I only want us to meet there”

Impact of faulty agents



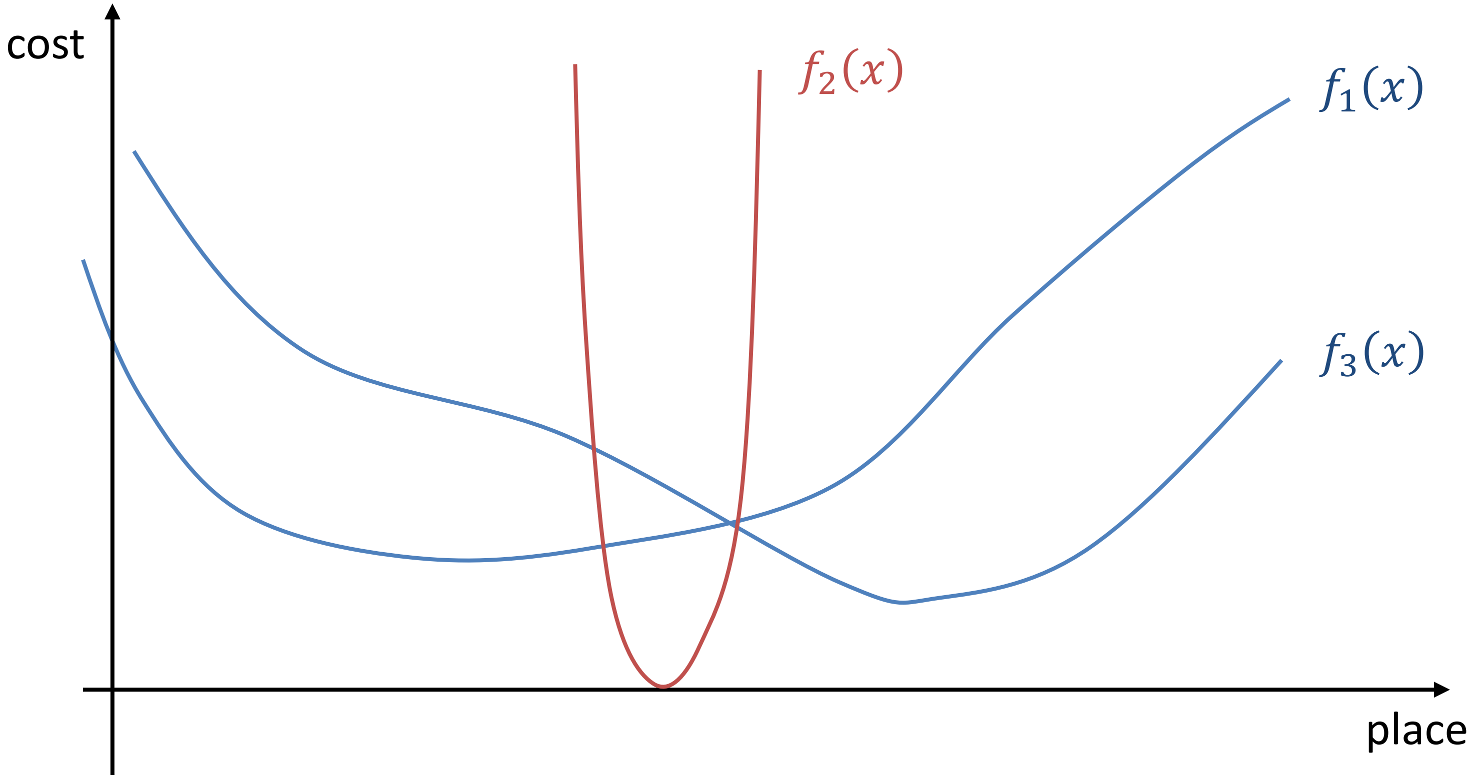
"I only want us to meet there"

Impact of faulty agents



“I only want us to meet there”

Impact of faulty agents



One faulty agent can disrupt the system

Impact of faulty agents

- Solving $\arg\min \sum_i f_i(x)$ is not useful
  - Faulty agents can send adversarial information

Impact of faulty agents

- Solving $\arg\min \sum_i f_i(x)$ is not useful
  - Faulty agents can send adversarial information

- What should be the goal of fault-tolerant optimization?

Fault-tolerance goal

- Minimize cost over only honest agents

- Find $\arg\min \sum_{\text{honest } i} f_i(x)$

$$\arg\min \sum_{\text{honest } i} f_i(x)$$

- Not achievable in general
- When can we achieve it exactly?

  ➔ Exact fault-tolerance
- When can we achieve it approximately?

  ➔ Approximate fault-tolerance

Exact fault-tolerance

Exact fault-tolerance

- Output $\arg\min \sum_{\mathrm{honest}\ i} f_i(x)$

Exact fault-tolerance

- Output $\arg\min \sum_{\text{honest } i} f_i(x)$

- Impossible without **redundancy** in cost functions

Redundancy

- Correlations among cost functions

- Common in practice
  - Observing the same object
  - Similarity between datasets
  - Correlation among data points

Exact fault-tolerance

- Output $\arg\min \sum_{\text{honest } i} f_i(x)$

- What type of redundancy required for exact fault-tolerance?

[Gupta & Vaidya, 2020]

$2f$-redundancy

[Gupta & Vaidya, 2020]

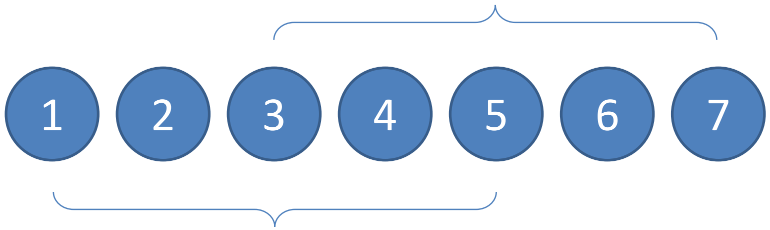
$\boxed{\text{Aggregate of every } n - 2f \text{ functions has the same minimum set}}$

$2f$-redundancy

[Gupta & Vaidya, 2020]

Aggregate of every $n - 2f$ functions has the same minimum set

$$x_2 = \arg\min \textstyle\sum_{i=3}^{7} f_i(x)$$



$$x_1 = \arg\min \textstyle\sum_{i=1}^{5} f_i(x)$$

$$n = 7$$
$$f = 1$$

$$x_1 = x_2 = \arg\min \textstyle\sum (5 \text{ functions})$$

$2f$-redundancy [Gupta & Vaidya, 2020]

Aggregate of every $n - 2f$ functions has the same minimum set

$2f$-redundancy $\Longleftrightarrow$ Exact fault-tolerance

$2f$-redundancy

- Strong condition, but not impossible
  - Example: replicated datasets

$2f$-redundancy

- Strong condition, but not impossible
  - Example: replicated datasets

- Difficult to satisfy in general

- Can we define a weaker goal?

Approximate fault-tolerance

Approximate fault-tolerance

- Need a measure for approximation

- We define (f, ϵ)-resilience

(f, ϵ)-resilience

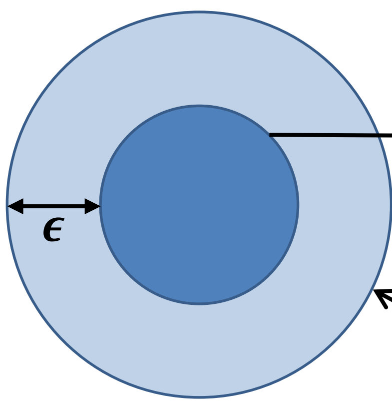
Algorithm output within ϵ of minimum for aggregate of every $n - f$ honest functions, in presence of f Byzantine agents

$$\text{distance}\left(\text{output},\ \ \arg\min \sum_{\substack{\text{honest}\\ n-f}} f_i(x)\right) \leq \epsilon$$

(f, ϵ)-resilience

$$\text{distance}\left(\text{output}, \ \arg\min \sum_{\substack{\text{honest} \\ n-f}} f_i(x)\right) \leq \epsilon$$

ϵ – error margin of the algorithm



$$x_{\mathcal{H}} = \arg\min \sum_{i \in \mathcal{H}} f_i(x)$$

Output within

$\mathcal{H}$ is set of $n - f$ honest agents

Approximate fault-tolerance

- (f, ϵ)-resilience describes an algorithm
- **Goal:** Achieve (f, ϵ)-resilience
- This requires adequate redundancy in cost functions

$(2f, \epsilon)$-redundancy

Aggregate of any $n - f$ cost functions and its subset of size $\geq n - 2f$ have minimizers within ϵ of each other

$$\text{distance}\left(\arg\min \sum_{\substack{\text{honest} \\ n-f}} f_i(x), \arg\min \sum_{\substack{\geq n-2f \\ subset}} f_i(x)\right) \leq \epsilon$$

$(2f, \epsilon)$-redundancy

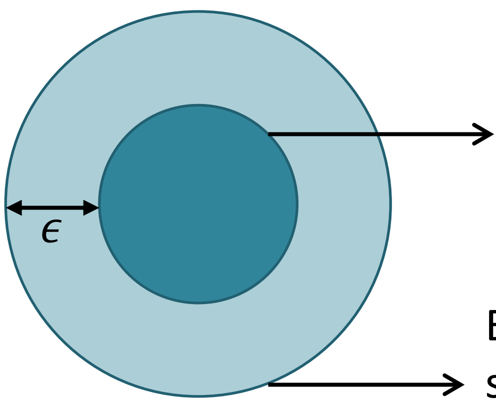
$$\boxed{\text{distance}\left(\arg\min \sum_{\substack{\text{honest}\\ n-f}} f_i(x)\,, \arg\min \sum_{\substack{\geq n-2f\\ subset}} f_i(x)\right) \leq \epsilon}$$

ϵ – how redundant the costs functions are

$(2f, \epsilon)$-redundancy

$$\boxed{\text{distance}\left(\arg\min \sum_{\substack{\text{honest} \\ n-f}} f_i(x), \arg\min \sum_{\substack{\geq n-2f \\ subset}} f_i(x)\right) \leq \epsilon}$$

ϵ – how redundant the costs functions are



$$x_{\mathcal{H}} = \arg\min \sum_{i \in \mathcal{H}} f_i(x)$$

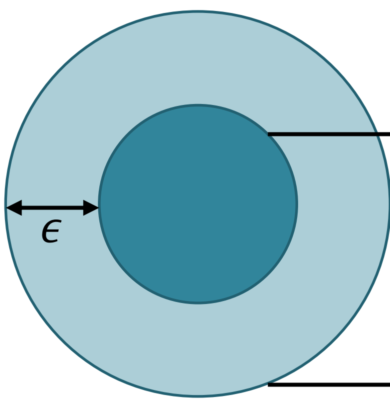
Every $\geq n - 2f$
subset minimizes
within this range

$(2f, \epsilon)$-redundancy

$(2f, \epsilon)$-redundancy

$$\text{distance}\left(\arg\min \sum_{\substack{\text{honest} \\ n-f}} f_i(x), \arg\min \sum_{\substack{\geq n-2f \\ subset}} f_i(x)\right) \leq \epsilon$$

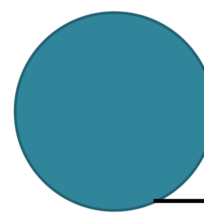
ϵ – how redundant the costs functions are



$x_{\mathcal{H}} = \arg\min \sum_{i \in \mathcal{H}} f_i(x)$

Every $\geq n - 2f$ subset minimizes within this range

$(2f, \epsilon)$-redundancy



Both $x_{\mathcal{H}}$ and minimizers of $n - 2f$ subsets

$2f$-redundancy
$(\epsilon = 0)$

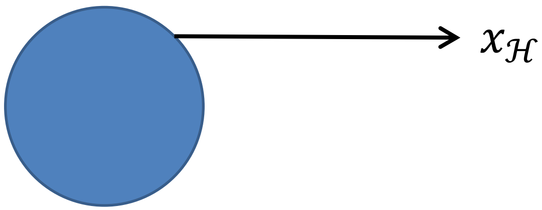
$(2f, \epsilon)$-redundancy

- $(2f, \epsilon)$-redundancy describes cost functions
- Can we achieve resilience with $(2f, \epsilon)$-redundancy?

Theoretical results

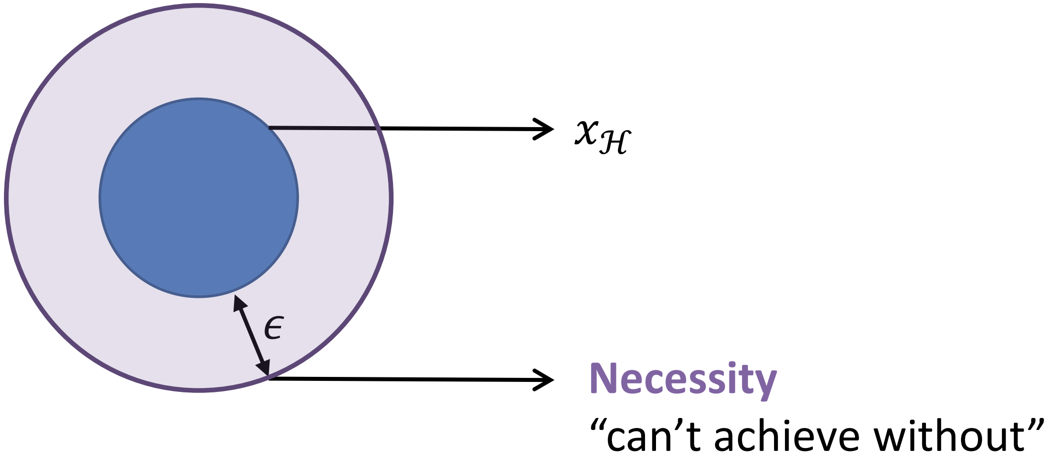
Necessity

(f, ϵ)-resilience can be achieved only if cost functions satisfy $(2f, \epsilon)$-redundancy



Necessity

(f, ϵ)-resilience can be achieved only if cost functions satisfy $(2f, \epsilon)$-redundancy

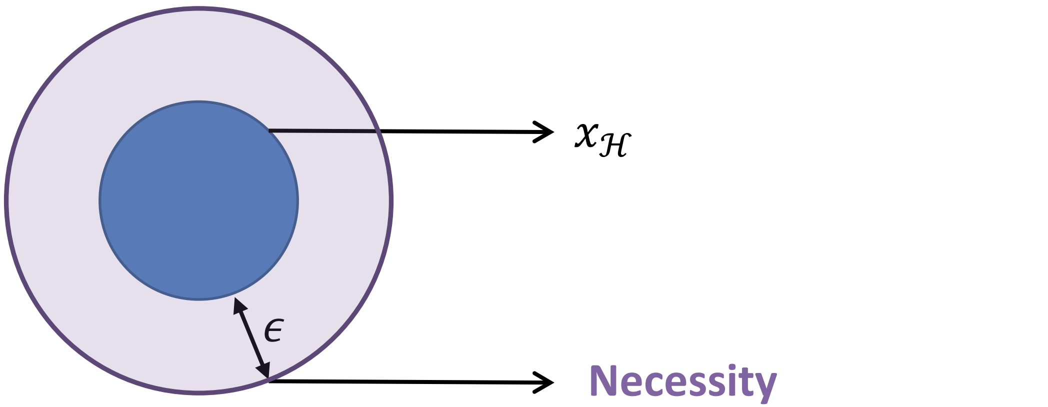


Necessity

(f, ϵ)-resilience can be achieved only if cost functions satisfy $(2f, \epsilon)$-redundancy

Sufficiency

If the cost functions satisfy $(2f, \epsilon)$-redundancy, $(f, 2\epsilon)$-resilience can be achieved

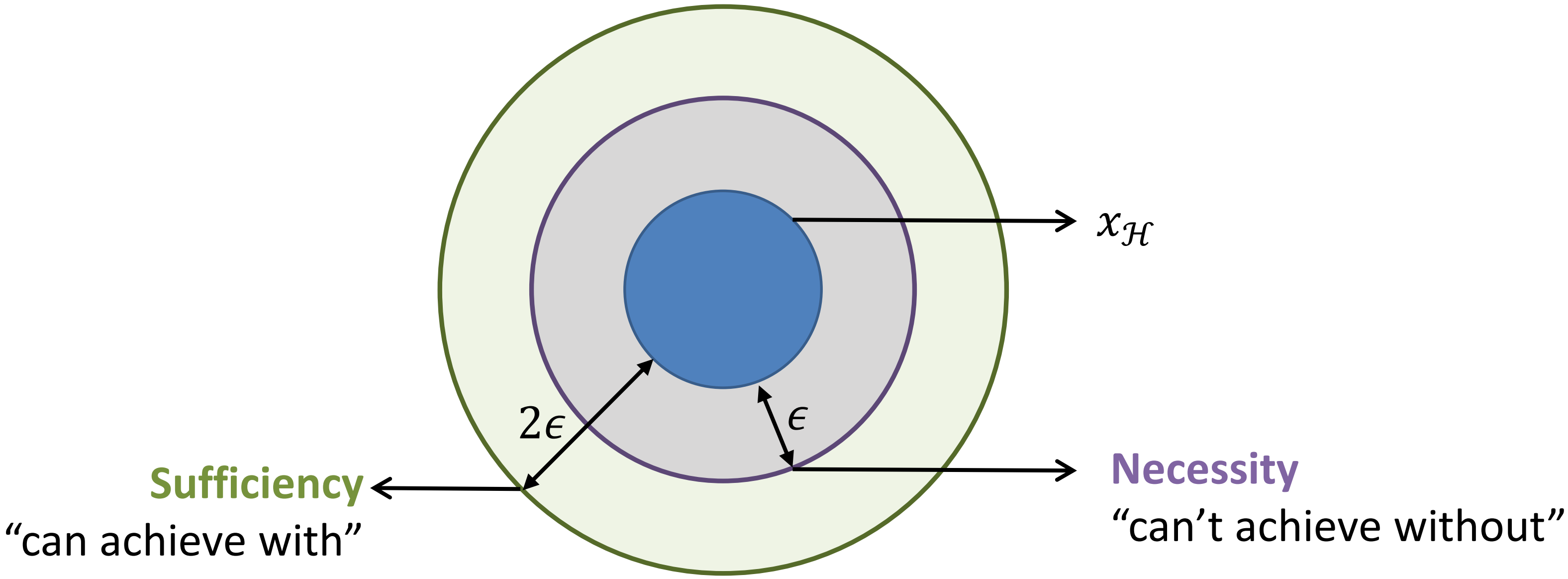


Necessity

(f, ϵ)-resilience can be achieved only if cost functions satisfy $(2f, \epsilon)$-redundancy

Sufficiency

If the cost functions satisfy $(2f, \epsilon)$-redundancy, $(f, 2\epsilon)$-resilience can be achieved



Approximate fault-tolerance

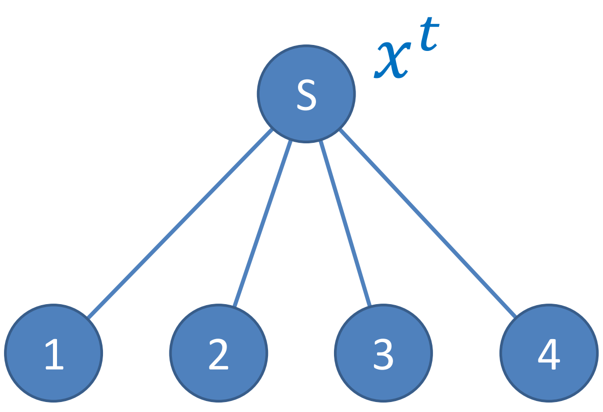
- **Result:** $\left(f, O(\epsilon)\right)$-resilience is achievable with $(2f, \epsilon)$-redundancy, and not achievable without *(In theory)*

- Is there any practical solution?

Fault-tolerant algorithm

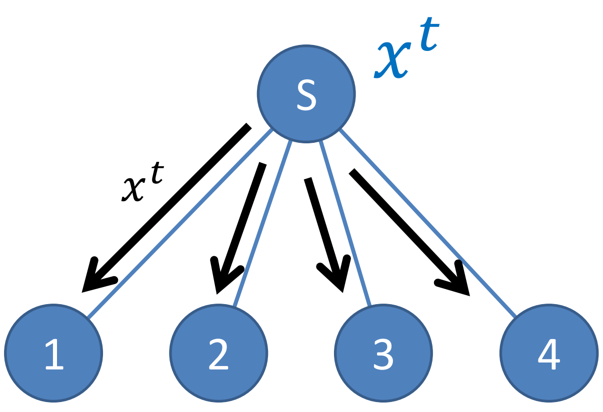
Distributed gradient descent

- Server maintains estimate x^t
- In each iteration t



Distributed gradient descent

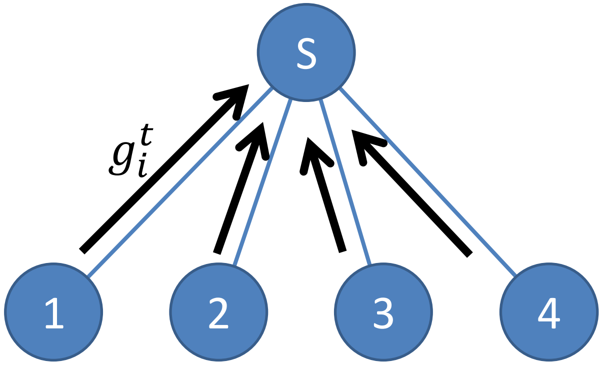
- Server maintains estimate x^t
- In each iteration t
  - Server broadcasts current estimate x^t



Distributed gradient descent

- Server maintains estimate x^t
- In each iteration t
  - Server broadcasts current estimate x^t

    Agents send back gradients g_i^t

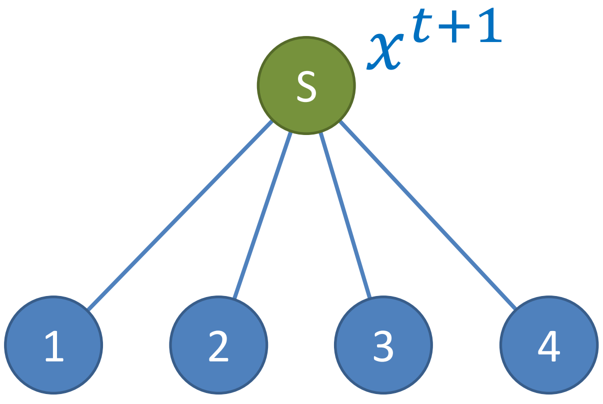


Distributed gradient descent

- Server maintains estimate x^t
- In each iteration t
  - Server broadcasts current estimate x^t

    Agents send back gradients g_i^t
  - Server collects the gradients and makes an update by

$$x^{t+1} = x^t - \eta \cdot \sum_i g_i^t$$

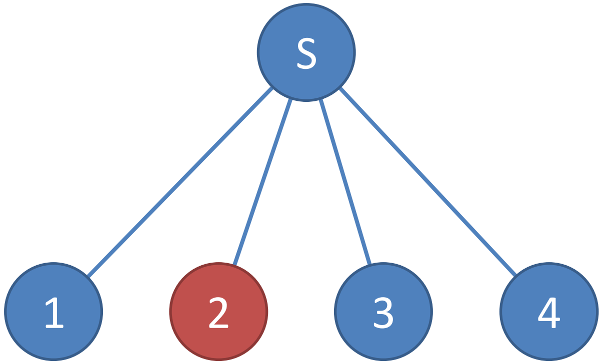


Distributed gradient descent

- Server maintains estimate x^t
- In each iteration t
  - Server broadcasts current estimate x^t

    Agents send back gradients g_i^t
  - Server collects the gradients and makes an update by

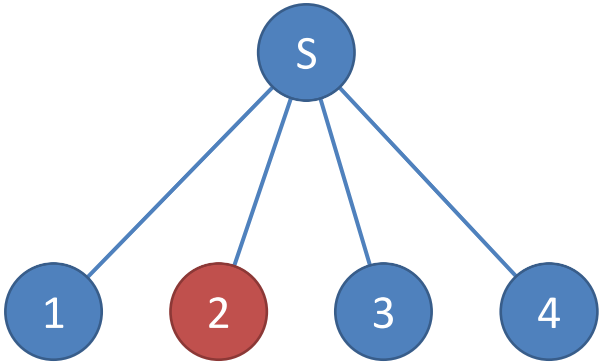
$$x^{t+1} = x^t - \eta \cdot \sum_i g_i^t$$



- Fails with faults

Fault-tolerant algorithm

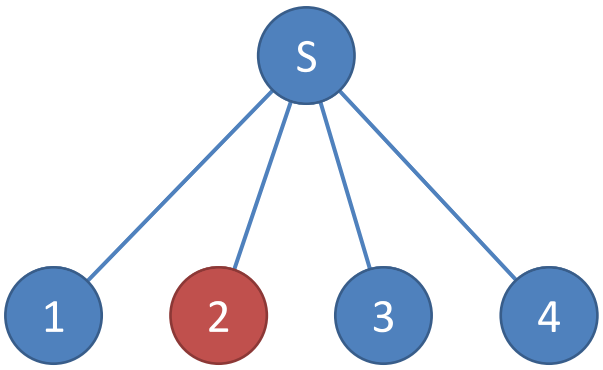
- Distributed gradients descent with filters
- In each iteration t



Fault-tolerant algorithm

- Distributed gradients descent with filters
- In each iteration t
  - Server broadcasts current estimate x^t

    Agents send back gradients or faulty vectors g_i^t

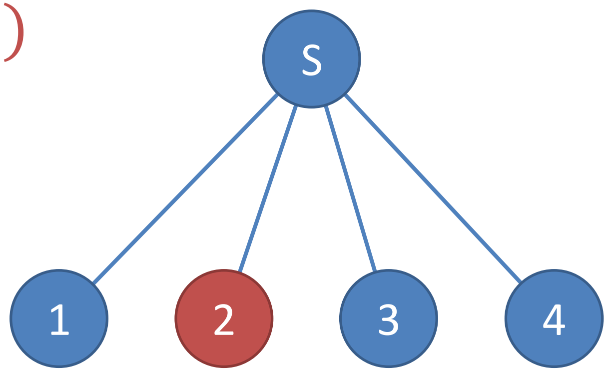


Fault-tolerant algorithm

- Distributed gradients descent with filters
- In each iteration t
  - Server broadcasts current estimate x^t

    Agents send back gradients or faulty vectors g_i^t
  - Server collects and filters the vectors, and makes an update

$$x^{t+1} = x^t - \eta \cdot \mathrm{GradFil}(g_i^t)$$



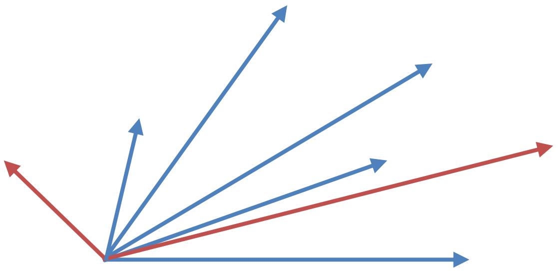
Gradient filters

- Also called gradient aggregation rules
- Mitigate faulty vectors
- Many designs
  - Krum [Blanchard et al., 2017]
  - Coordinate-wise methods [Yin et al., 2018]
  - Comparative gradient elimination [Gupta & Vaidya, 2019]
  - Geometric median, median of means, … [Chen et al., 2017]

Gradient filters

- Comparative gradient elimination

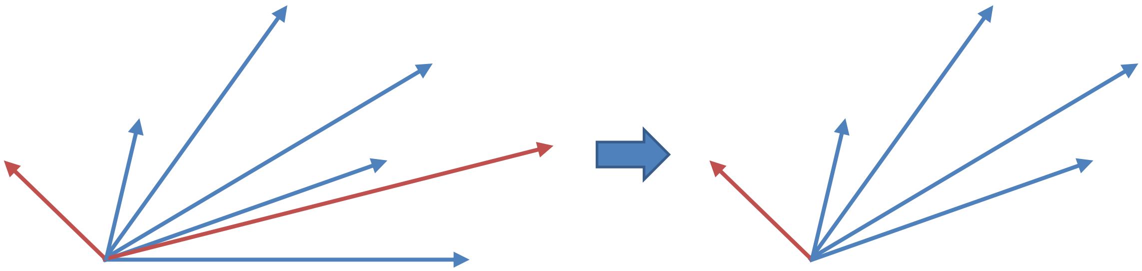
$$\boldsymbol{n = 7}$$
$$\boldsymbol{f = 2}$$



Gradient filters

$\boldsymbol{n = 7}$
$\boldsymbol{f = 2}$

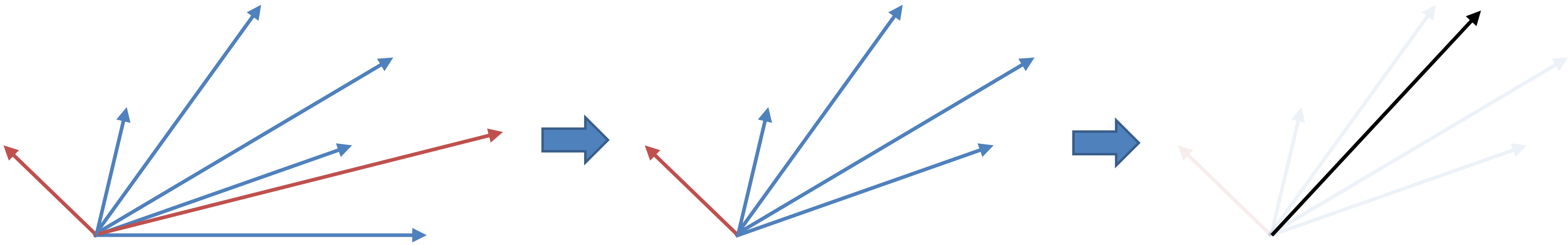
- Comparative gradient elimination



Gradient filters

$$\boldsymbol{n = 7}$$
$$\boldsymbol{f = 2}$$

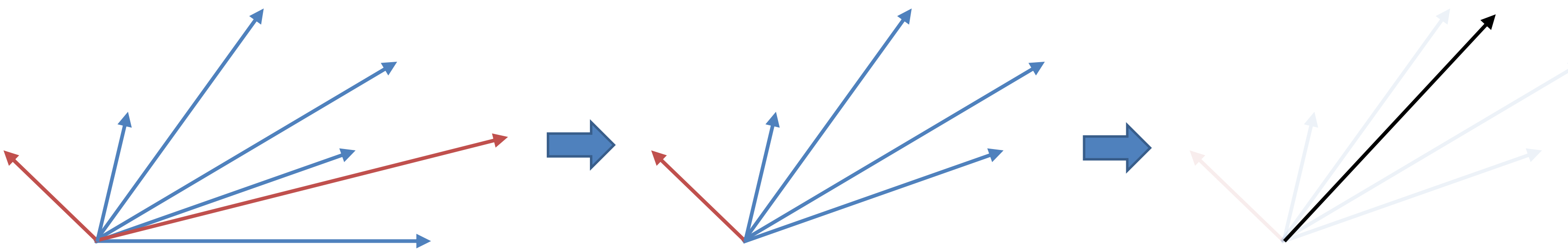
- Comparative gradient elimination



Gradient filters

$\boldsymbol{n = 7}$
$\boldsymbol{f = 2}$

- Comparative gradient elimination



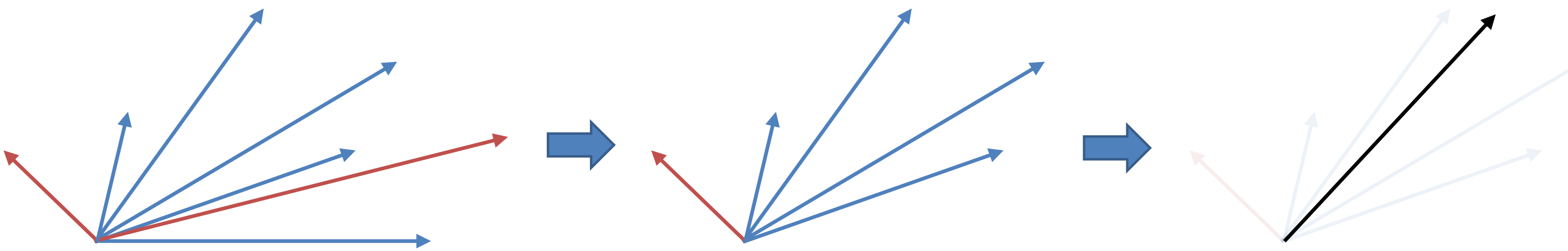
- Coordinate-wise trimmed mean

			⋮			
1	4	5	2	3	5	6
			⋮			

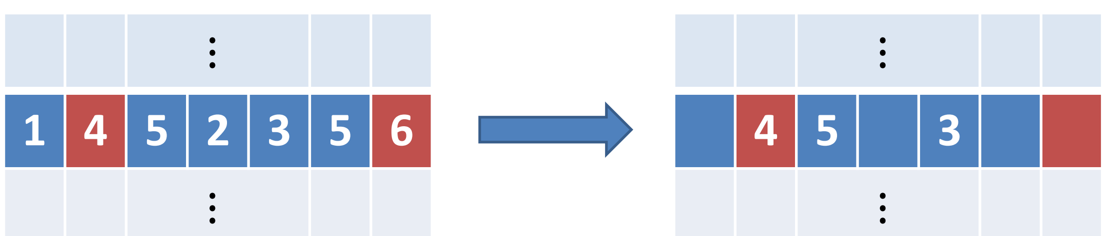
Gradient filters

- Comparative gradient elimination

$$n = 7$$
$$f = 2$$



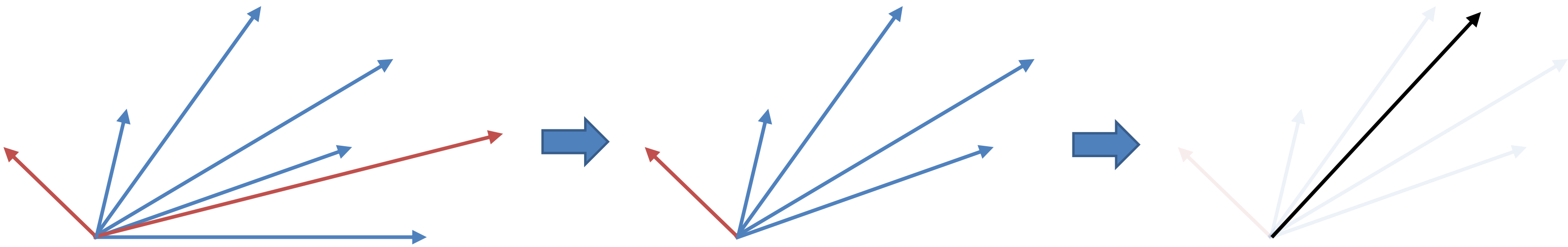
- Coordinate-wise trimmed mean



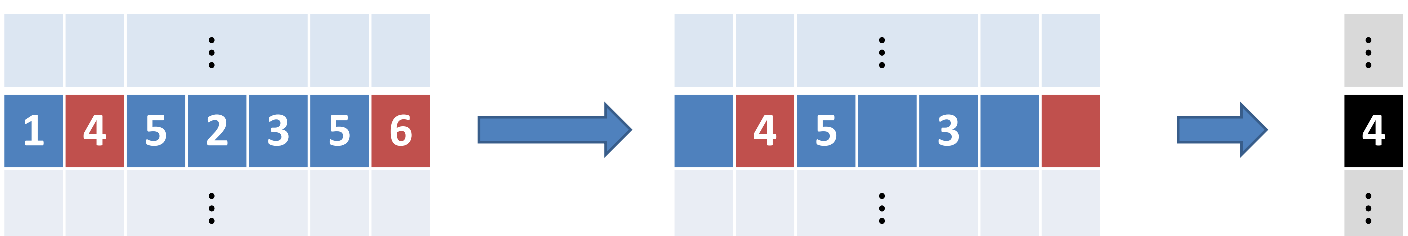
Gradient filters

$n = 7$
$f = 2$

- Comparative gradient elimination



- Coordinate-wise trimmed mean



Performance

With $(2f, \epsilon)$-redundancy, the algorithm is $\left(f, \mathcal{O}(\epsilon)\right)$-resilient

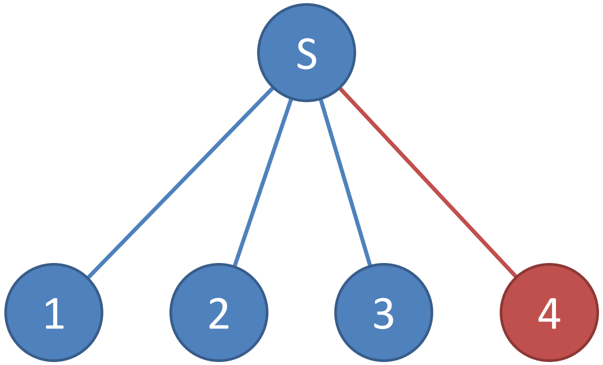
- Error margin decided by gradient filter
- Gradient elimination requires $f \leq n/3$
- Trimmed mean's margin depends on parameter vector size

• Trade-off between complexity and error margin

Stragglers + Byzantine agents

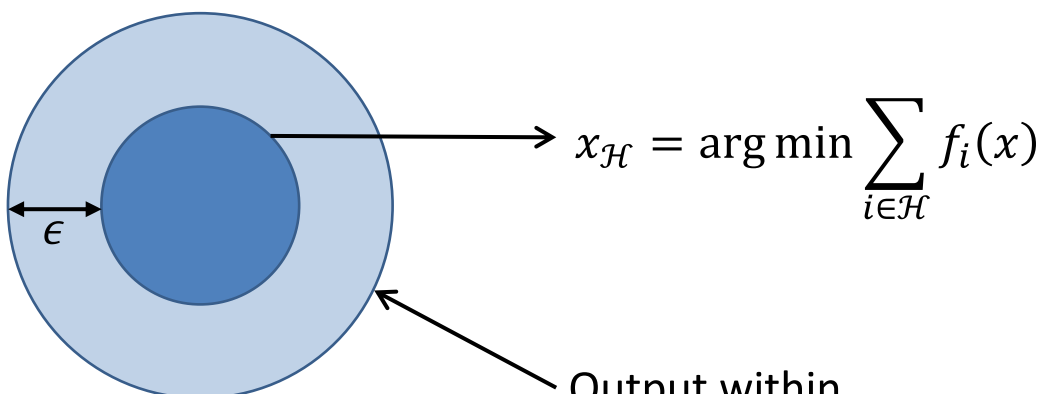
Stragglers

- Stragglers are slow agents
- Stragglers delay synchronous algorithm
- Solution
  - Don’t wait for slow agents
  - Exploit redundancy



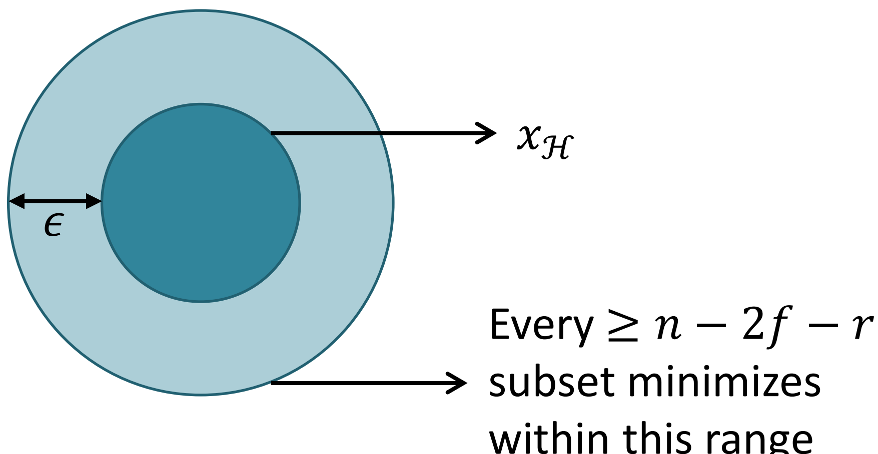
Algorithm: $(f, r; \epsilon)$-resilient

Algorithm output within ϵ of minimum for aggregate of every $n - f$ honest functions in presence of f Byzantine agents and r stragglers



Costs: $(f, r; \epsilon)$-redundancy

Aggregate of any $n - f$ cost functions and its subset of size $\geq n - 2f - r$ have minimizers within ϵ of each other



$(f, r; \epsilon)$-redundancy

Resilient distributed optimization

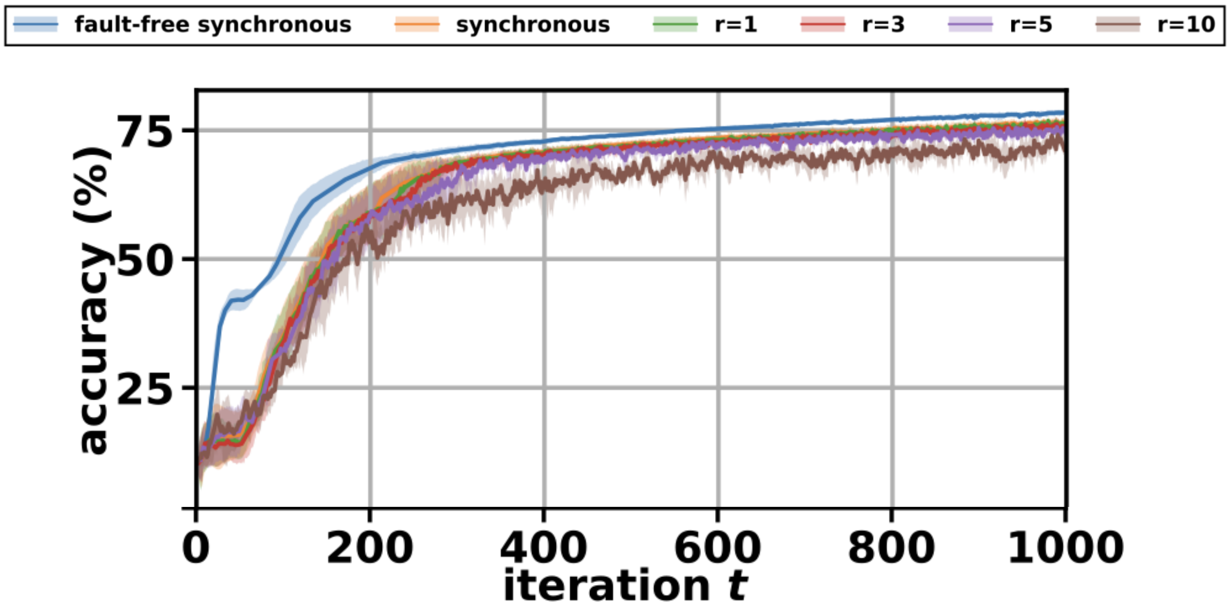
- Can show similar necessity for $(f, r; \epsilon)$-redundancy

- Can show $\left(f, r; \mathcal{O}(\epsilon)\right)$-resilience for DGD + filter

Experiments

f: faulty agents
r: stragglers

- MNIST on LeNet – Label-flipping faults
- $n = 20, f = 3$, various r's



Summary

- Defined (f, ϵ)-resilience and $(2f, \epsilon)$-redundancy
- Obtained necessary and sufficient conditions
- Algorithm with gradient filters
- Extended to stragglers

Thank you

Thank you

Questions

Presented papers

- Nirupam Gupta and Nitin H. Vaidya. Fault-Tolerance in Distributed Optimization: The Case of Redundancy. (PODC 2020)
- Shuo Liu, Nirupam Gupta, and Nitin H. Vaidya. Approximate Byzantine Fault-Tolerance in Distributed Optimization. (PODC 2021)
- Shuo Liu, Nirupam Gupta, and Nitin H Vaidya. Impact of Redundancy on Resilience in Distributed Optimization and Learning. (ICDCN 2023)