Fault-Tolerance and Privacy in Distributed Optimization

Nitin Vaidya
University of Illinois at Urbana-Champaign

Acknowledgements

Shripad Gade

Lili Su

Secret to happiness is to
lower your expectations to the point
where they're already met

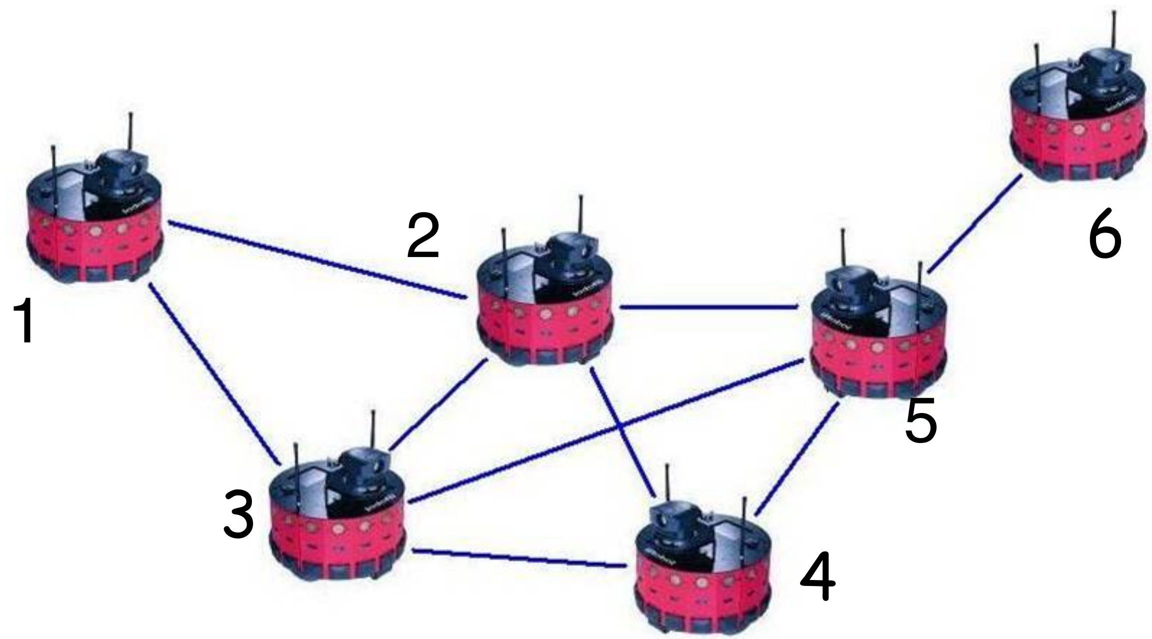
Secret to happiness is to
lower your expectations to the point
where they're already met?

- Hobbes

$$argmin \sum_{i} f_i(x)$$

Applications

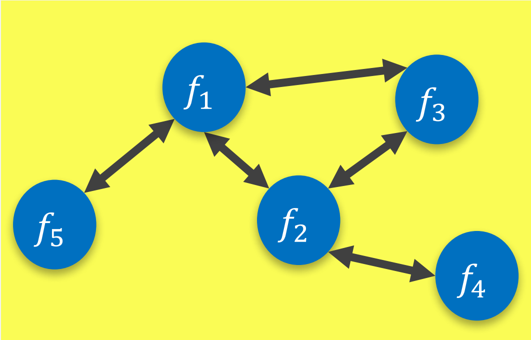
- $f_i(x)$ = cost of robot i to go to location x
- Minimize total cost of rendezvous



Applications

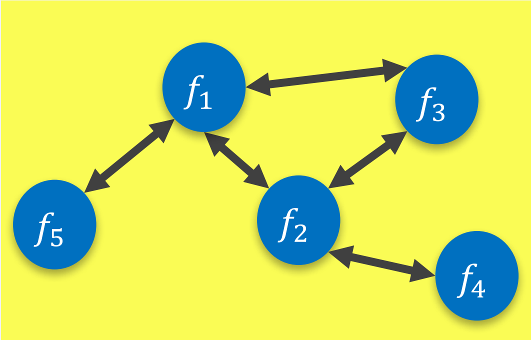
- Machine learning
- Smart grid distributed control
- …

$$\underset{i}{argmin} \sum f_i(x)$$

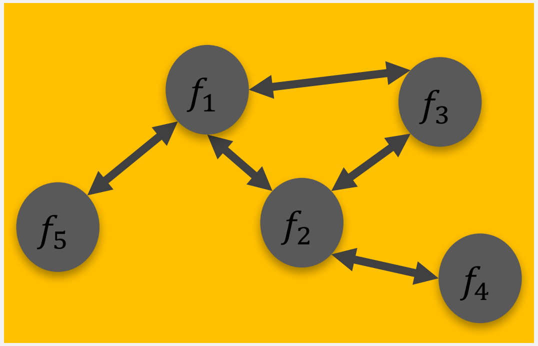


Distributed Optimization

$$argmin \sum_i f_i(x)$$

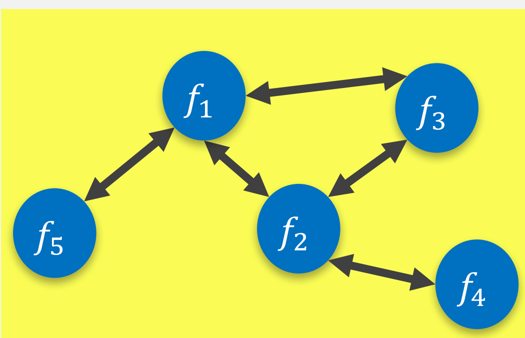


Distributed Optimization

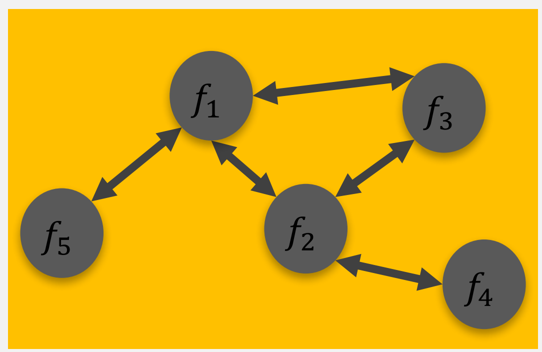


Privacy

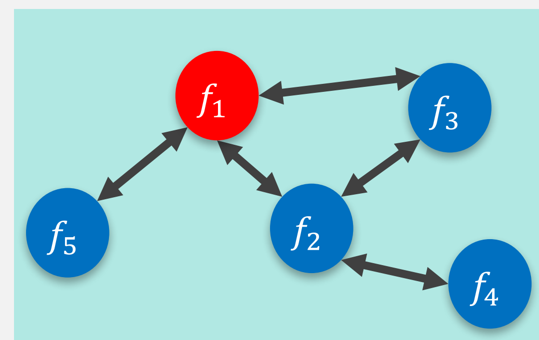
$$argmin \sum_{i} f_i(x)$$



Distributed Optimization



Privacy



Fault-tolerance

Background

Background

Reaching a Consensus

MORRIS H. DeGROOT*

1974

Consider a group of individuals who must act together as a team or committee, and suppose that each individual in the group has his own subjective probability distribution for the unknown value of some parameter. A model is presented which describes how the group might reach agreement on a common subjective probability distribution for the parameter by pooling their individual opinions. The process leading to the consensus is explicitly described and the common distribution that is reached is explicitly determined. The model can also be applied to problems of reaching a consensus when the opinion of each member of the group is represented simply as a point estimate of the parameter rather than as a probability distribution.

1. INTRODUCTION

Consider a group of k individuals who must act together as a team or committee, and suppose that each of these k individuals can specify his own subjective probability distribution for the unknown value of some parameter θ. In this article we shall present a model which describes how the group might reach a consensus and form a common subjective probability distribution for θ simply by revealing their individual distributions to

distribution over Ω for which the probability of any measurable set A is $\sum_{i=1}^{k} p_i F_i(A)$. Some of the writers previously mentioned have suggested representing the overall opinion of the group by a probability distribution of the form $\sum_{i=1}^{k} p_i F_i$. Stone [13] has called such a linear combination an "opinion pool." The difficulty in using an opinion pool to represent the consensus of the group lies, of course, in choosing suitable weights $p_1, \cdots, p_k$. In the model that will be presented in this article, the consensus that is reached by the group will have the form of an opinion pool. However, the model is new. It explicitly describes the process which leads to the consensus and explicitly specifies the weights that are to be used in the opinion pool.

In summary, this model is believed to have three important advantages:

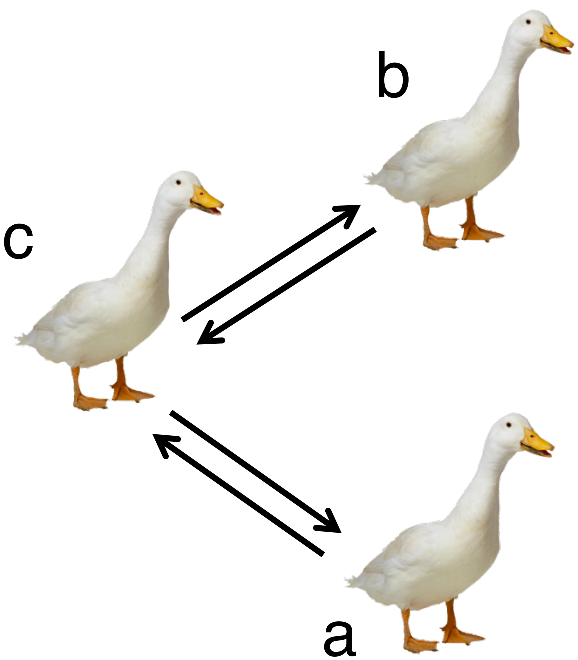
1. The process that it describes is intuitively appealing.
2. It presents simple conditions for determining whether it is possible for the group to reach a consensus.

Consensus: “Flocking problem”

Each iteration = Local averaging

$b = (b+c)/2$

$c = (a+b+c)/3$

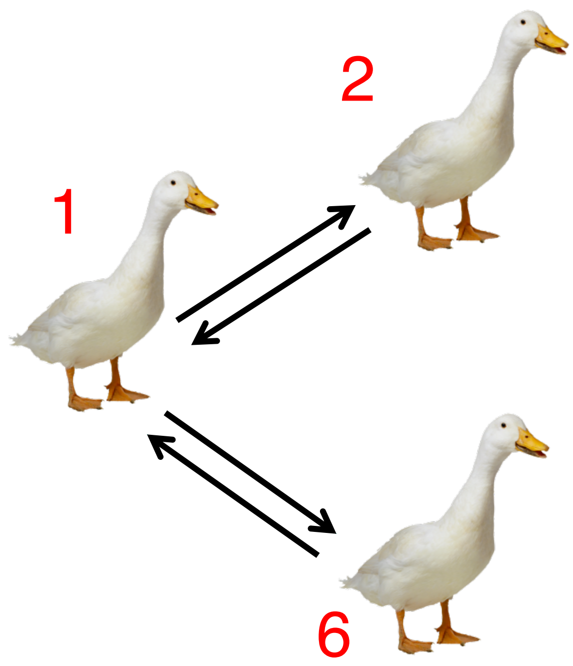


$a = (a+c)/2$

Consensus

Each iteration = Local averaging

c = (1+2+6)/3 = 3



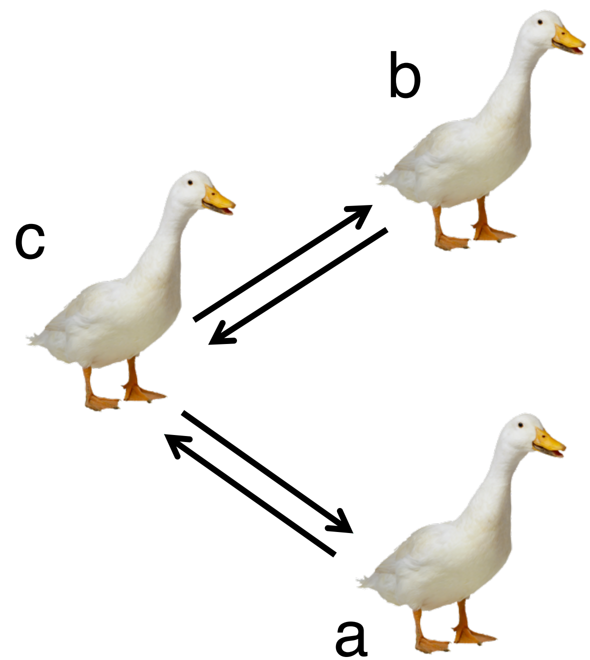
b = (1+2)/2 = 3/2

a = (1+7)/2 = 7/2

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} := \underset{M}{\begin{pmatrix} 1/2 & 0 & 1/2 \\ 0 & 1/2 & 1/2 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = M \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

b = (b+c)/2

c = (a+b+c)/3



a = (a+c)/2

$b = (b+c)/2$

$c = (a+b+c)/3$

$a = (a+c)/2$

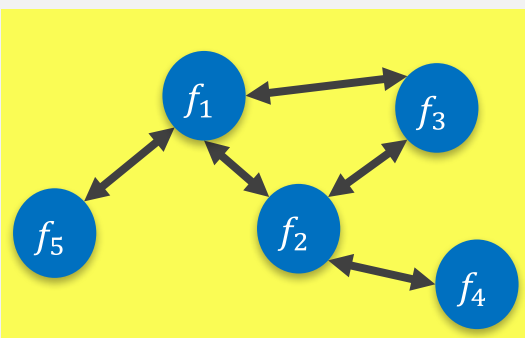
Average Consensus

- If M is chosen doubly stochastic, states converge to average of initial inputs

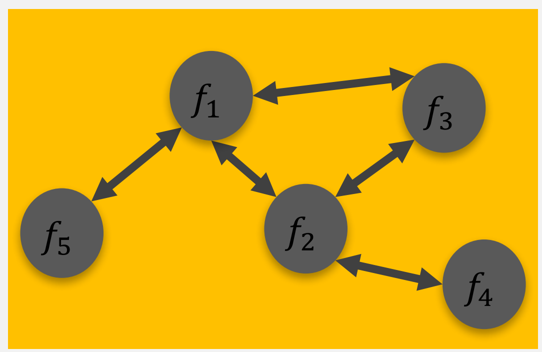
Why does this work?

Why does this work?

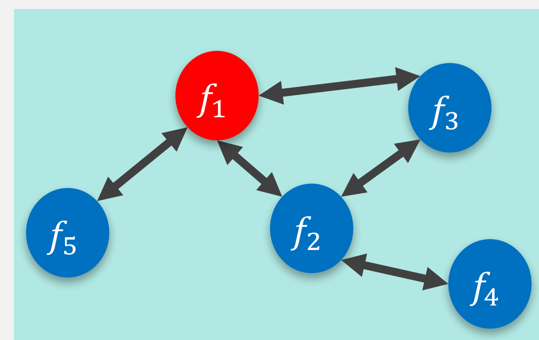
$$argmin \sum_{i} f_i(x)$$



Distributed Optimization



Privacy



Fault-tolerance

Optimization

$$argmin \sum_i f_i(x)$$

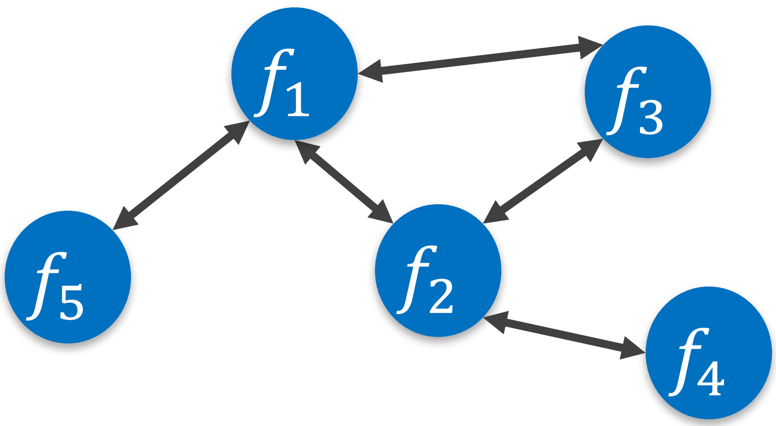
Optimization

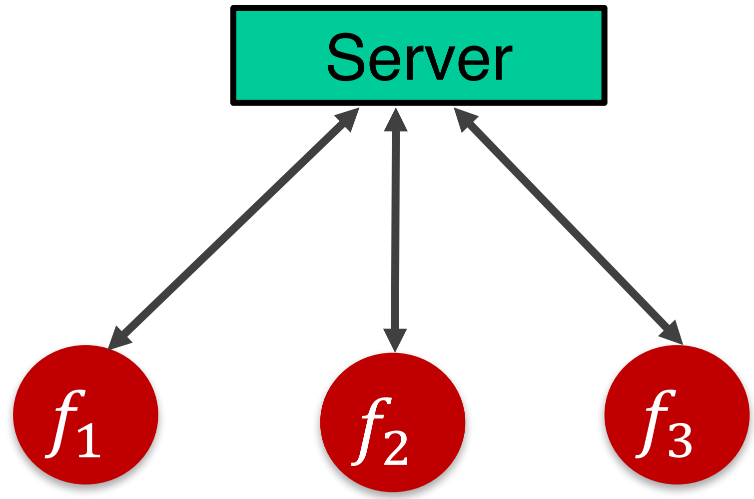
$$argmin \sum_i f_i(x)$$

Gradient Descent

$$x_{k+1} \longleftarrow x_k - \alpha_k \sum_i \nabla f_i(x_k)$$

Distributed Optimization





1984

PROBLEMS IN DECENTRALIZED DECISION MAKING AND COMPUTATION

by

John Nikolaos Tsitsiklis

B.S., Massachusetts Institute of Technology
(1980)

S.M., Massachusetts Institute of Technology
(1981)

SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE
DEGREE OF

DOCTOR OF PHILOSOPHY

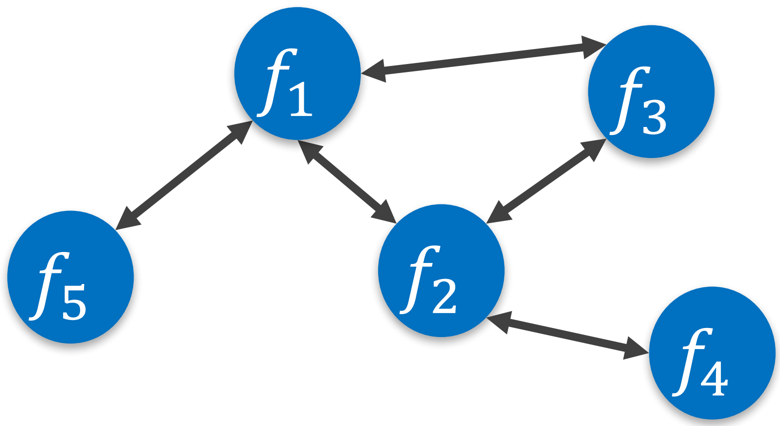
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

November 1984

Distributed <u>Peer-to-Peer</u> Optimization

- Each agent maintains local estimate x

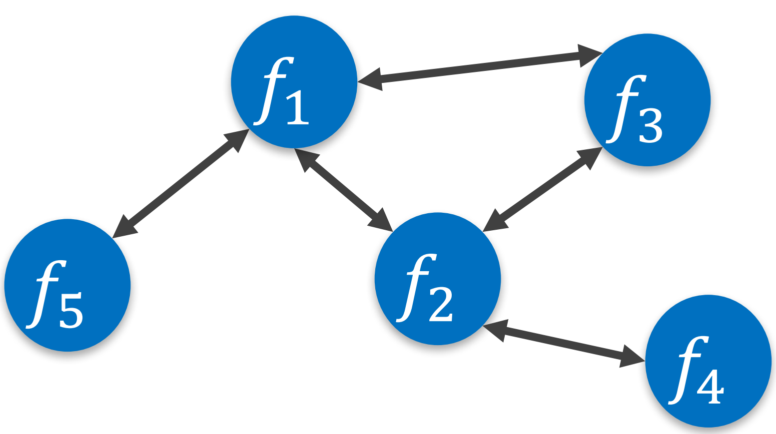


Distributed Peer-to-Peer Optimization

- Each agent maintains local estimate x

In each iteration

- Compute weighted average with neighbors' estimates



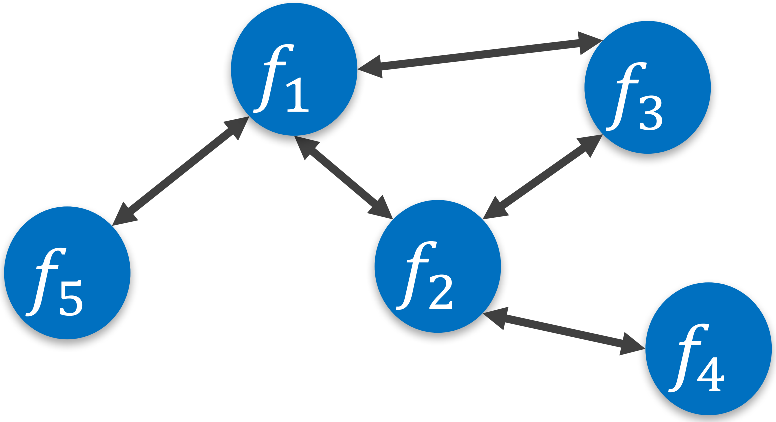
Distributed Peer-to-Peer Optimization

- Each agent maintains local estimate x

In each iteration

- Compute weighted average with neighbors' estimates
- Apply own gradient to own estimate

$$x_{k+1} \longleftarrow x_k - \alpha_k \nabla f_i(x_k)$$



Distributed Peer-to-Peer Optimization

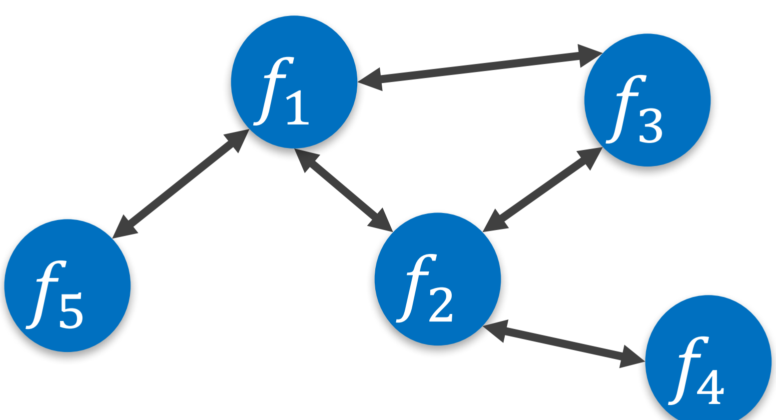
- Each agent maintains local estimate x

In each iteration

- Compute weighted average with neighbors' estimates
- Apply own gradient to own estimate

$$x_{k+1} \leftarrow x_k - \alpha_k \nabla f_i(x_k)$$

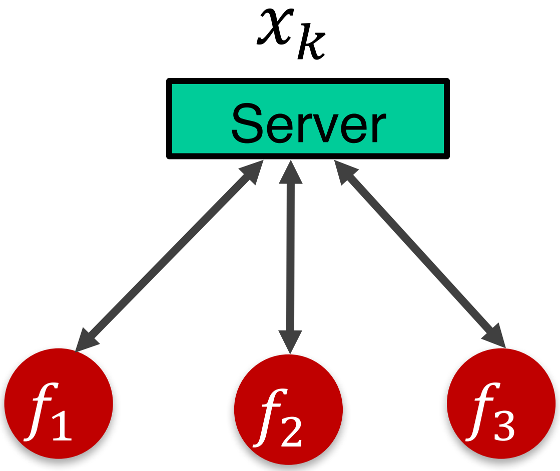
- Local estimates converge to $argmin \sum_i f_i(x)$



Why does this work?

Distributed Client-Server Optimization

- Server S maintains estimate x_k
- Client i knows $f_i(x)$

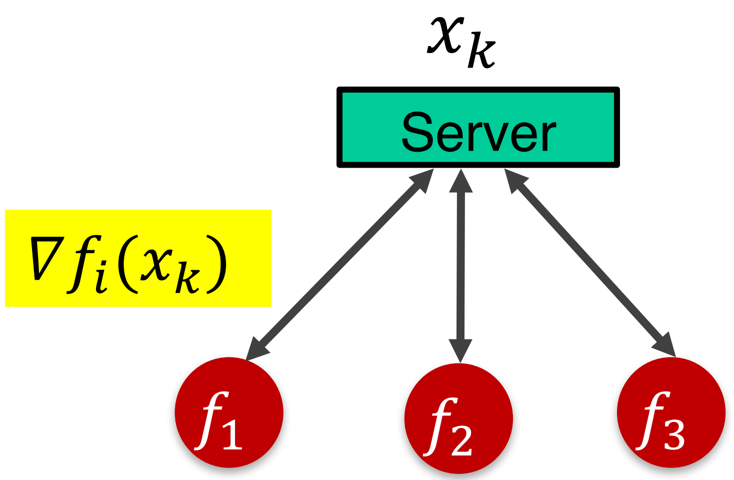


Distributed Client-Server Optimization

- Server S maintains estimate x_k
- Client i knows $f_i(x)$

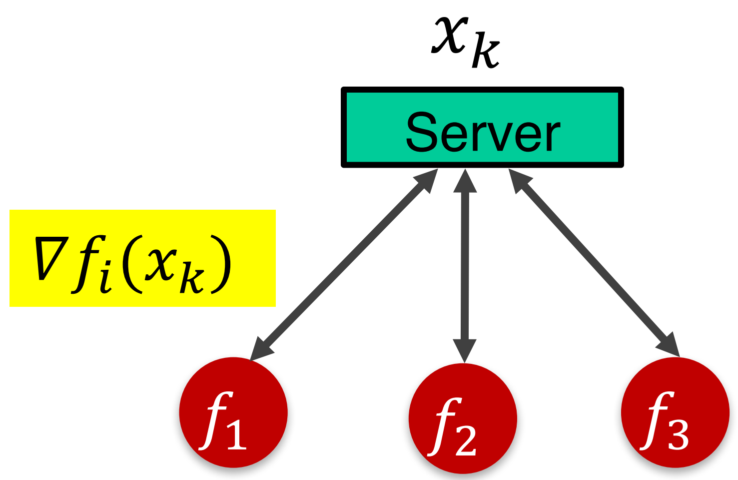
In each iteration

- Client i
  - Download x_k from server
  - Upload gradient $\nabla f_i(x_k)$



Distributed Client-Server Optimization

- Server S maintains estimate x_k
- Client i knows $f_i(x)$



In each iteration

- Client i
  - Download x_k from server
  - Upload gradient $\nabla f_i(x_k)$

- Server

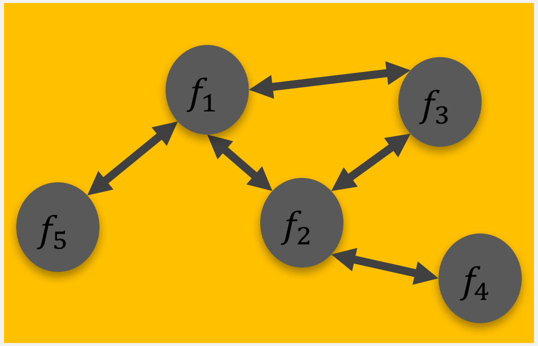
$$x_{k+1} \leftarrow x_k - \alpha_k \sum_i \nabla f_i(x_k)$$

Variations

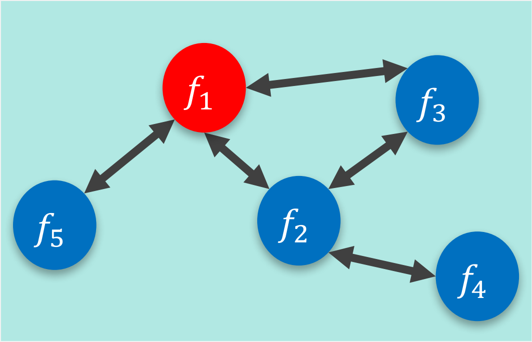
- Asynchronous
- Stochastic

Our Work

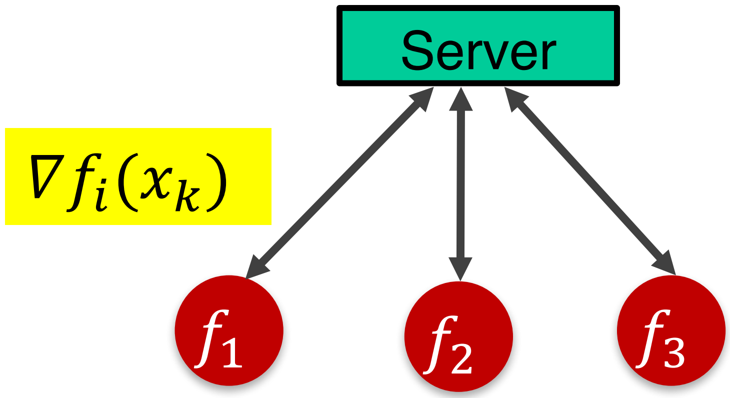
Challenges



Privacy



Fault-tolerance



Server observes gradients ➔ privacy compromised

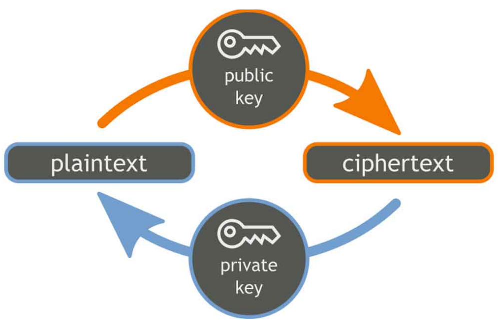
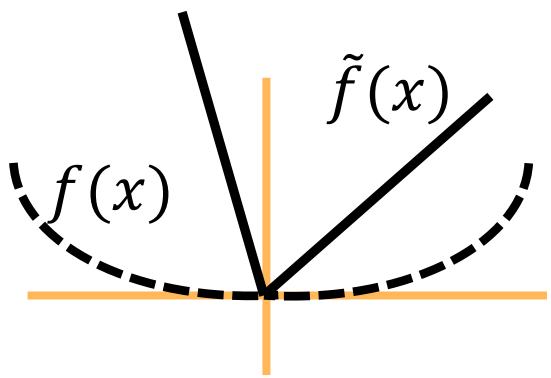
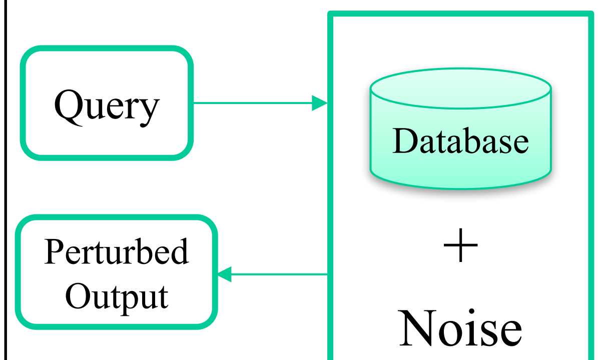
- Similar concerns in a peer-to-peer setting as well

- Similar concerns in a peer-to-peer setting as well

Achieve privacy and yet collaboratively compute

$$argmin \sum_i f_i(x)$$

Related Work

Cryptographic Methods	Transformation Methods	Differential Privacy
		

Solution Approach

- Motivated by secret sharing

Solution Approach

- Motivated by secret sharing

- Several variations …

Add noise that "deterministically cancels out"

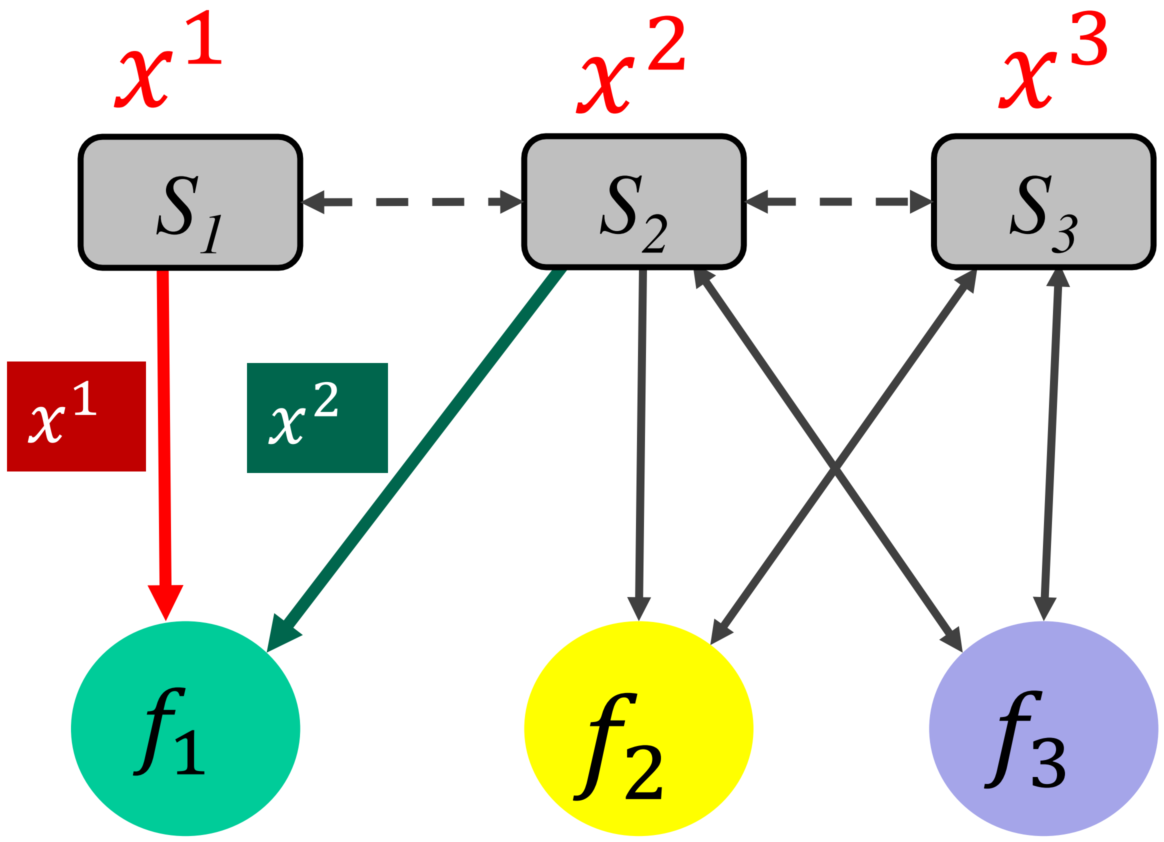
Solution Approach

- Motivated by secret sharing
- Several variations …

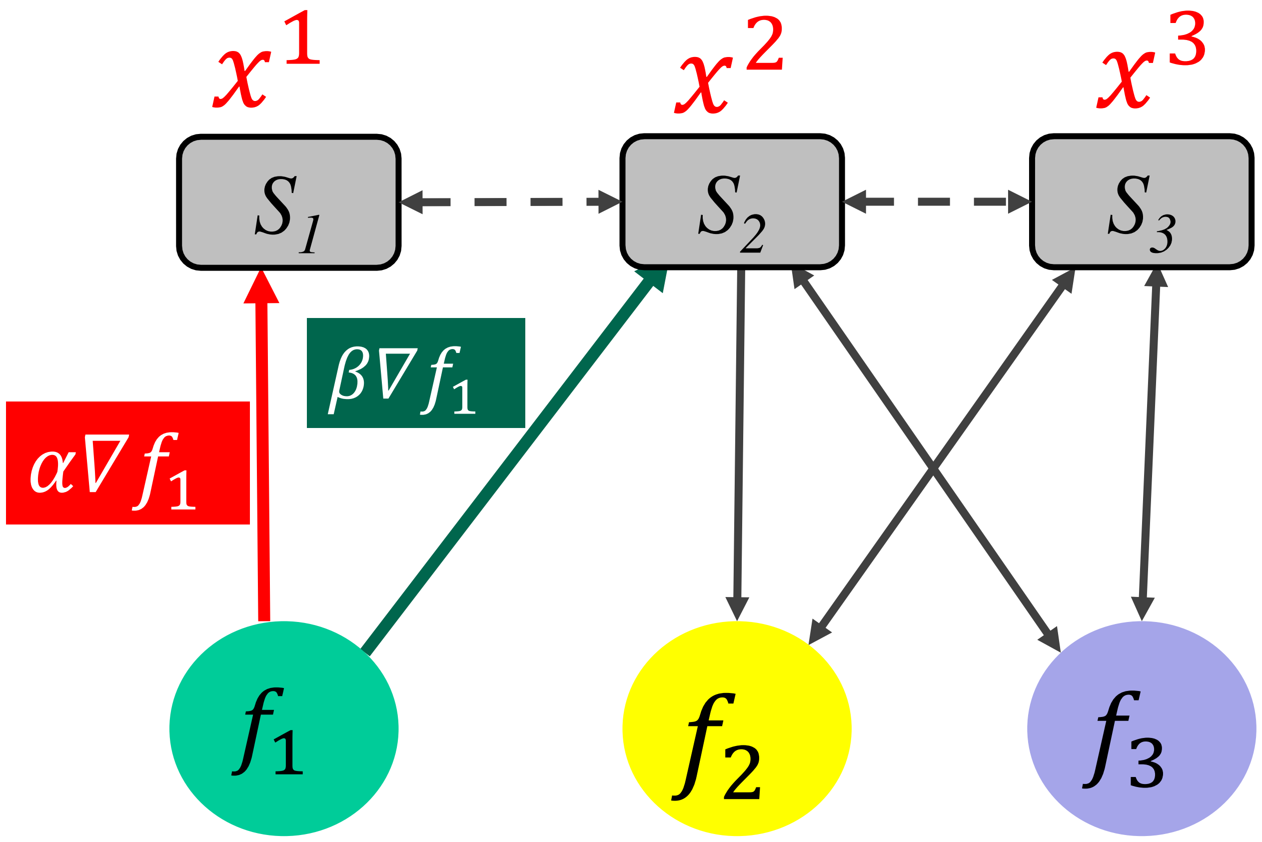
Add noise that "deterministically cancels out"

Shripad Gade

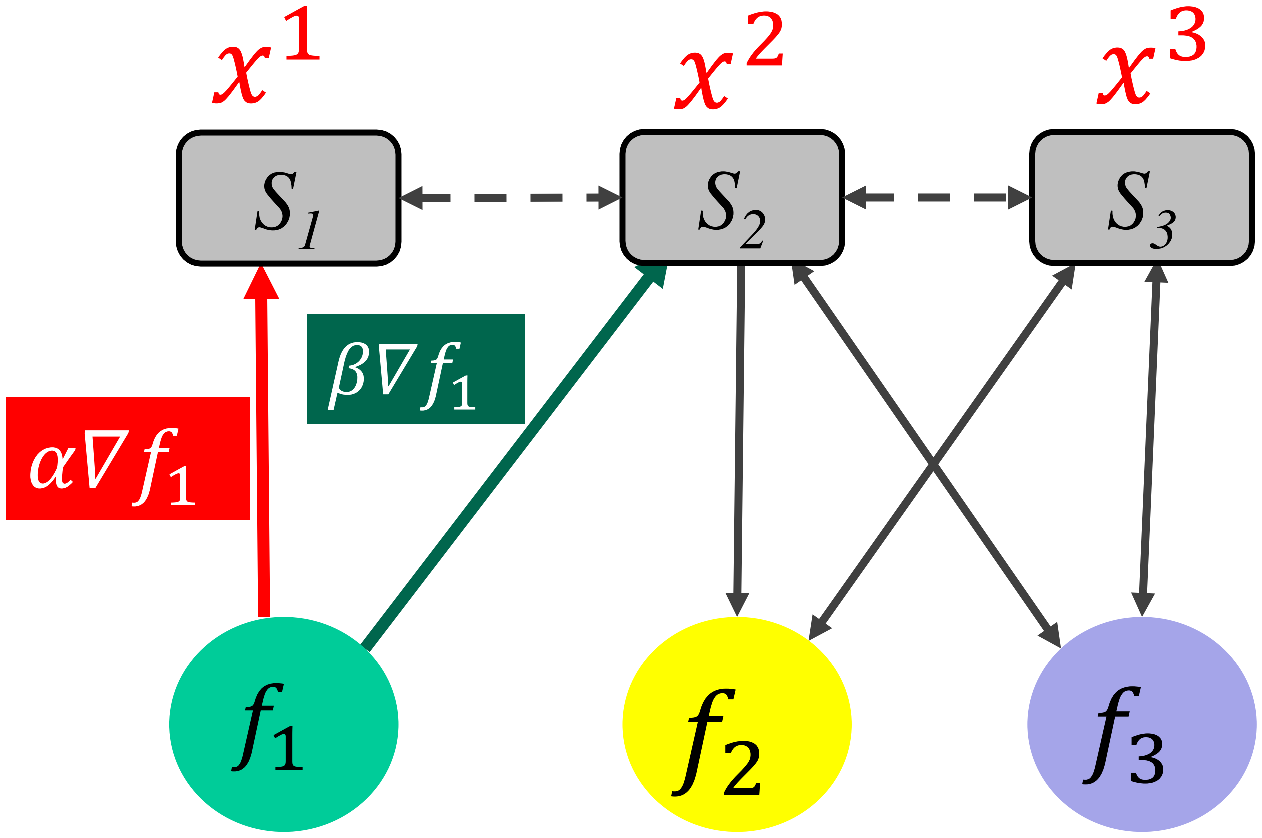
Client-Server Illustration



Client-Server Illustration



Client-Server Illustration

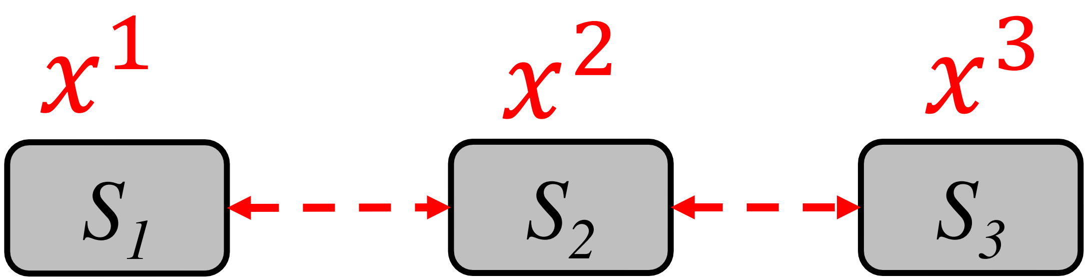


Server 1 applies received gradients to compute new x^1

Server Consensus

- Servers periodically perform a "consensus" step

➔ Exchange estimates, and compute weighted average



Symmetric Weights

- Weights $(\alpha, \beta, \ldots)$ used by each client sum to 1 over time window of size

  → Ensures that each client is "weighed" equally

- Weights known only to each client

  → Improves privacy

Convergence

- Estimates converge to

$$argmin \sum_i f_i(x)$$

Privacy

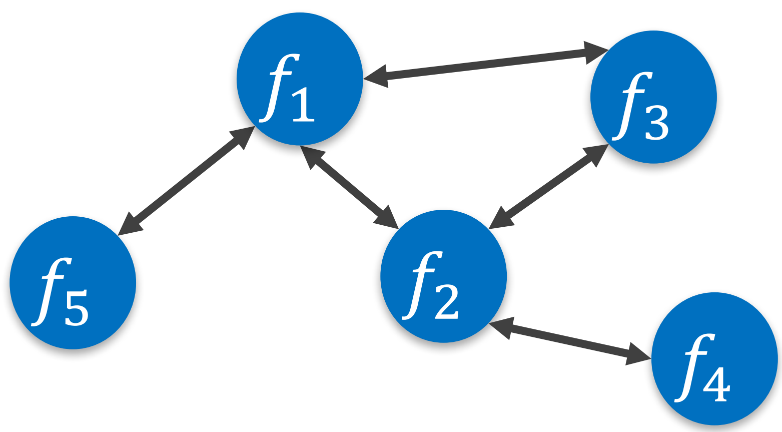
- With suitable choice of weights

  no strict subset of servers can learn any client's

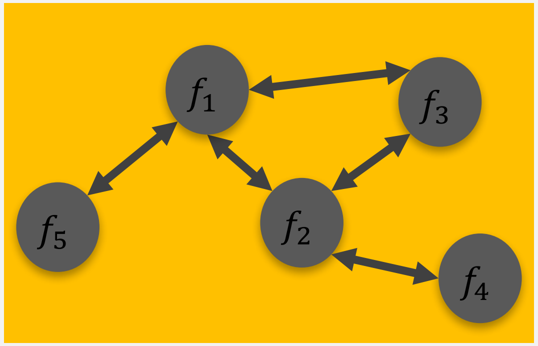
  cost function

Peer-to-Peer

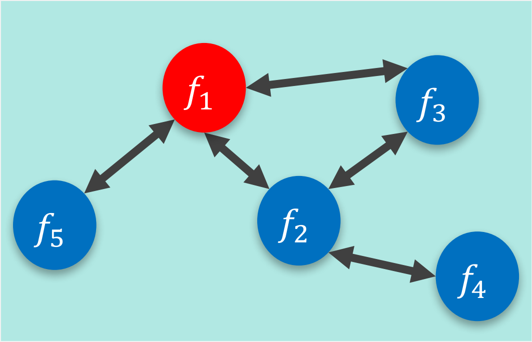
- Add noise in information sent to neighbors such that noise cancels out over all neighbors



Challenges



Privacy



Fault-tolerance

Fault-Tolerance

Lili Su

Byzantine Fault Model

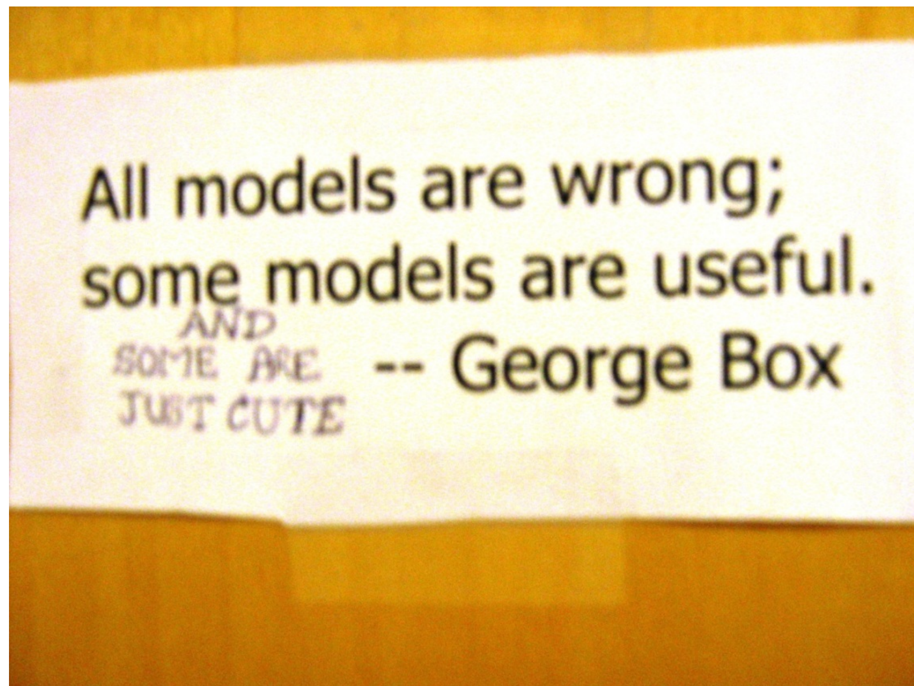
- No constraint
  on misbehavior
  of a faulty node

Byzantine Fault Model

- No constraint

  on misbehavior

  of a faulty node



1980

Reaching Agreement in the Presence of Faults

M. PEASE, R. SHOSTAK, AND L. LAMPORT

SRI International, Menlo Park, California

ABSTRACT. The problem addressed here concerns a set of isolated processors, some unknown subset of which may be faulty, that communicate only by means of two-party messages. Each nonfaulty processor has a private value of information that must be communicated to each other nonfaulty processor. Nonfaulty processors always communicate honestly, whereas faulty processors may lie The problem is to devise an algorithm in which processors communicate their own values and relay values received from others that allows each nonfaulty processor to infer a value for each other processor The value inferred for a nonfaulty processor must be that processor's private value, and the value inferred for a faulty one must be consistent with the corresponding value inferred by each other nonfaulty processor

It is shown that the problem is solvable for, and only for, $n \geq 3m + 1$, where m is the number of faulty processors and n is the total number. It is also shown that if faulty processors can refuse to pass on information but cannot falsely relay information, the problem is solvable for arbitrary $n \geq m \geq 0$. This weaker assumption can be approximated in practice using cryptographic methods

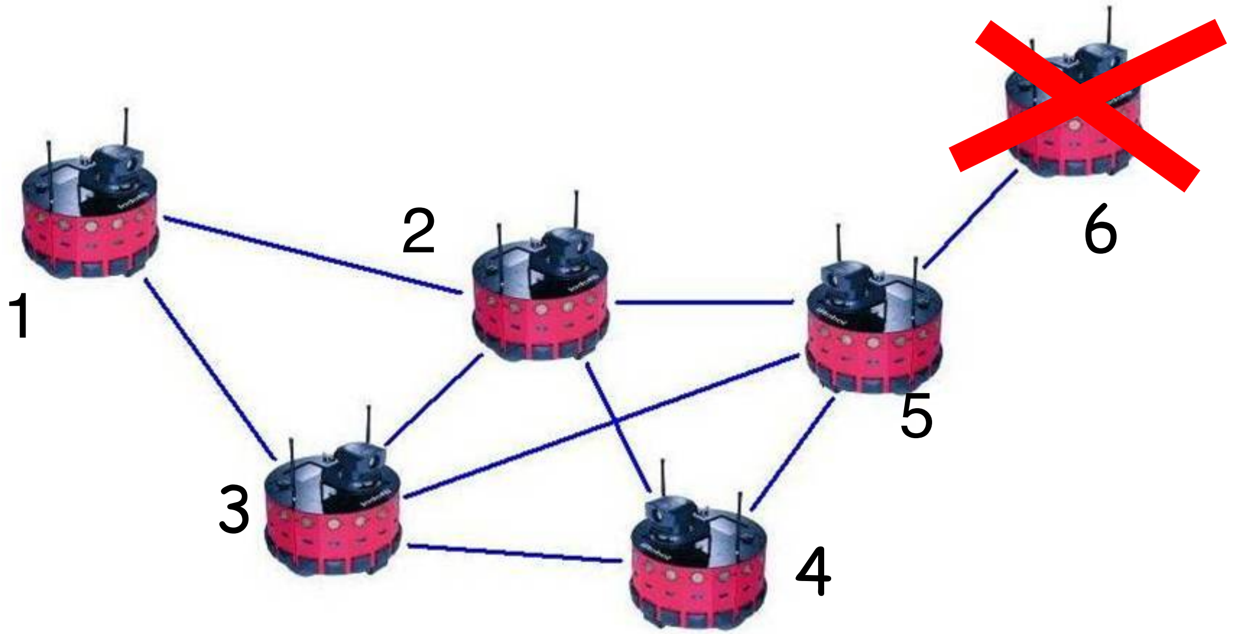
KEY WORDS AND PHRASES. agreement, authentication, consistency, distributed executive, fault avoidance, fault tolerance, synchronization, voting

CR CATEGORIES: 3.81, 4.39, 5.29, 5.39, 6.22

Fault-Tolerant Optimization

- $f_i(x)$ = cost of robot i to go to location x

- Faulty agent may choose arbitrary cost function



Fault-Tolerant Optimization

- The original problem is not meaningful

$$argmin \sum_i f_i(x)$$

Fault-Tolerant Optimization

- The original problem is not meaningful

$$argmin \sum_{i} f_i(x)$$

- Optimize cost over only non-faulty agents

$$argmin \sum_{i\ good} f_i(x)$$

Fault-Tolerant Optimization

- The original problem is not meaningful

$$argmin \sum_{i} f_i(x)$$

- Optimize cost over only non-faulty agents

Impossible!

$$argmin \sum_{i\ good} f_i(x)$$

Fault-Tolerant Optimization

- Optimize weighted cost over only non-faulty agents

$$\underset{i\ good}{argmin} \sum f_i(x)\, \boldsymbol{\alpha}_i$$

Fault-Tolerant Optimization

- Optimize weighted cost over only non-faulty agents

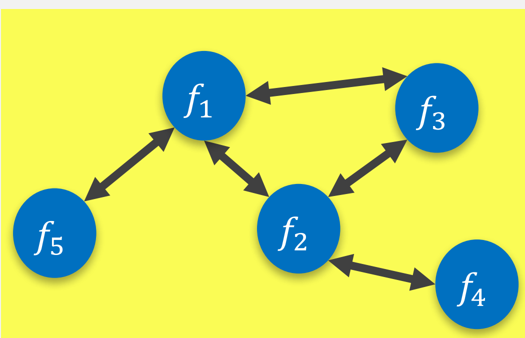
$$\underset{i\ good}{argmin \sum} f_i(x)\, \boldsymbol{\alpha}_i$$

Lower bounds on number and value of

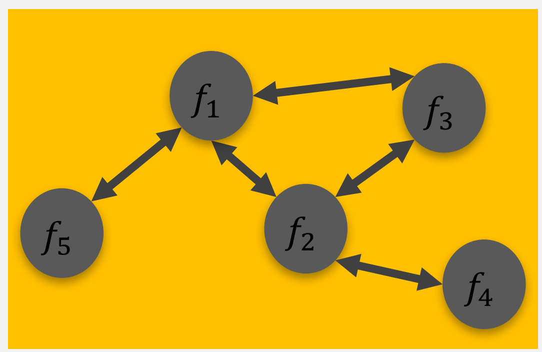
of non-zero weights $\boldsymbol{\alpha}_{\mathrm{i}}$

Summary

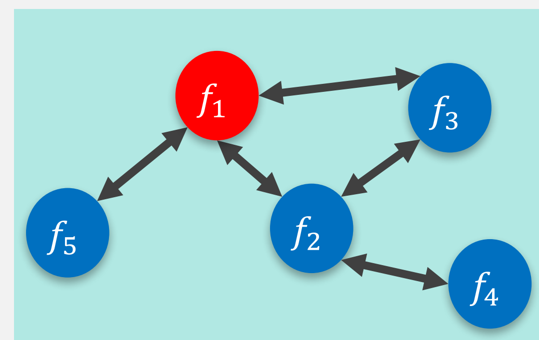
$$\underset{i}{argmin} \sum f_i(x)$$



Distributed Optimization



Privacy



Fault-tolerance

Thanks!

disc.ece.illinois.edu