Privacy Preserving Randomized Gossip Algorithms

Peter Richtárik

Workshop on Decentralized Machine Learning, Optimization and Privacy
Lille, France, October 11-12, 2017

Talked Based on

Mathematics > Optimization and Control

Privacy Preserving Randomized Gossip Algorithms

Filip Hanzely, Jakub Konečný, Nicolas Loizou, Peter Richtárik, Dmitry Grishchenko

(Submitted on 23 Jun 2017)

In this work we present three different randomized gossip algorithms for solving the average consensus problem while at the same time protecting the information about the initial private values stored at the nodes. We give iteration complexity bounds for all methods, and perform extensive numerical experiments.

Comments: 38 pages
Subjects: **Optimization and Control (math.OC)**
Cite as: **arXiv:1706.07636 [math.OC]**
(or **arXiv:1706.07636v1 [math.OC]** for this version)

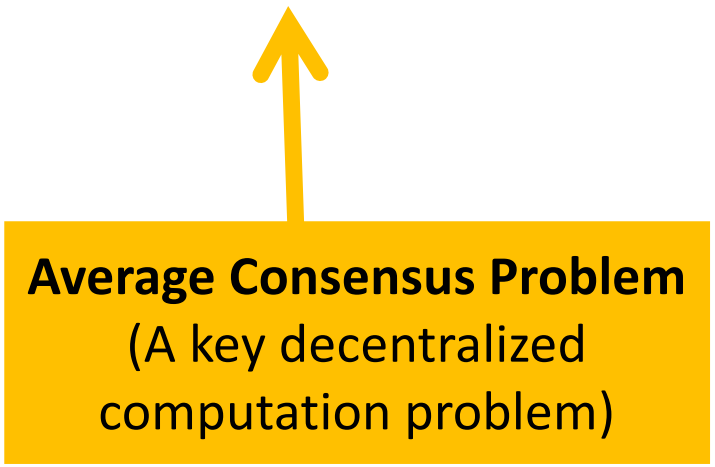
HKLRG 2017

This Talk vs Workshop Theme

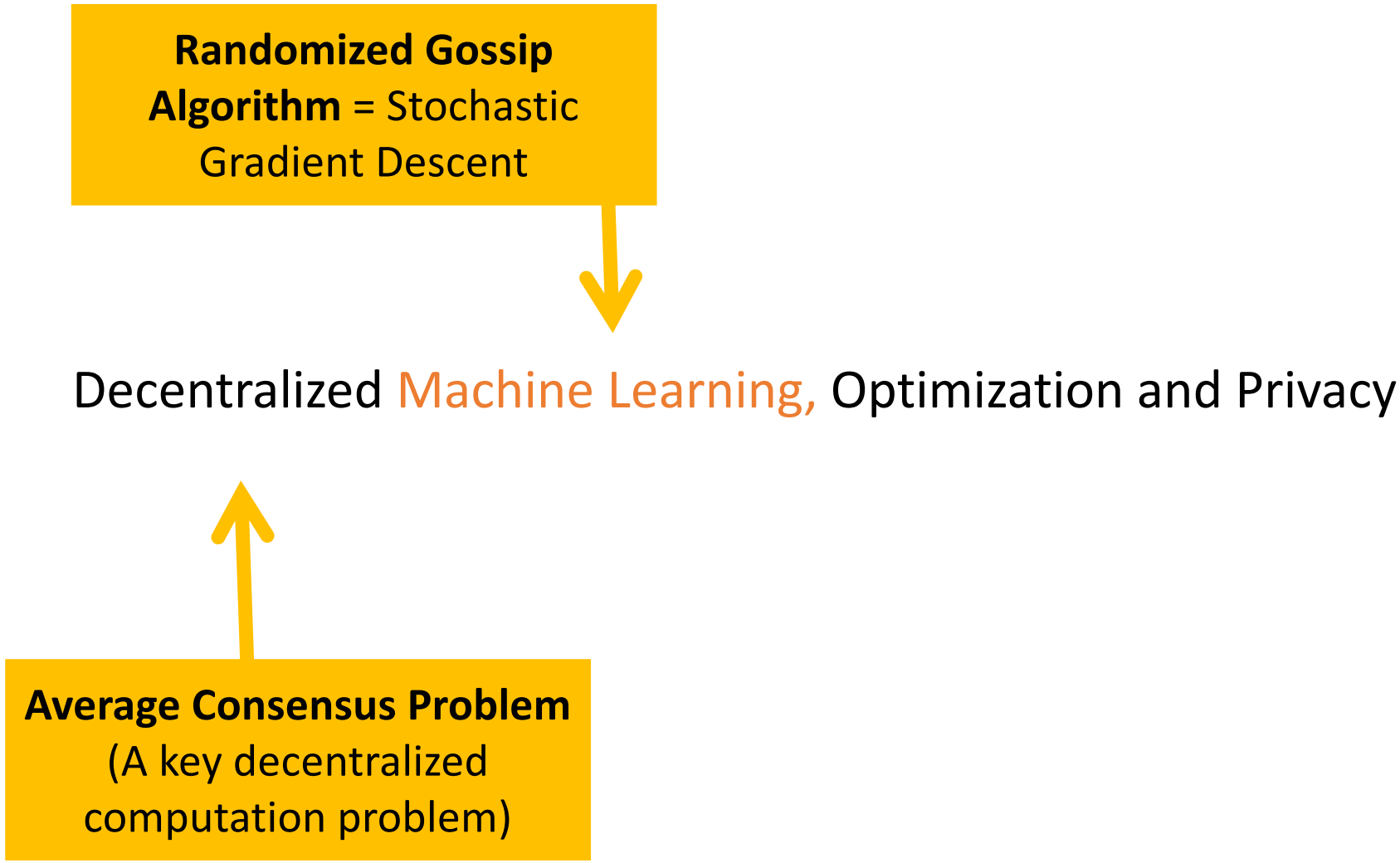
Decentralized Machine Learning, Optimization and Privacy

This Talk vs Workshop Theme

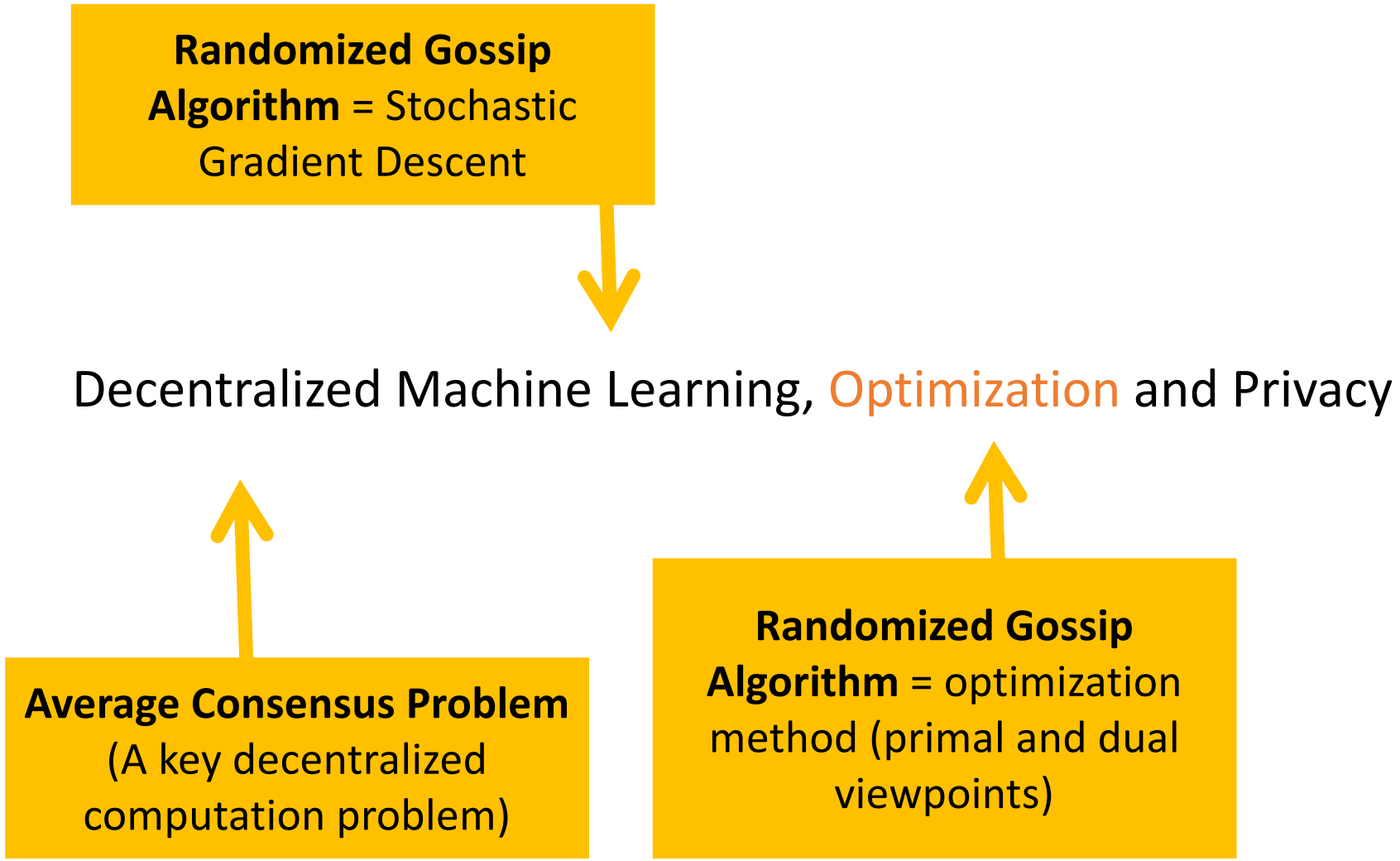
Decentralized Machine Learning, Optimization and Privacy



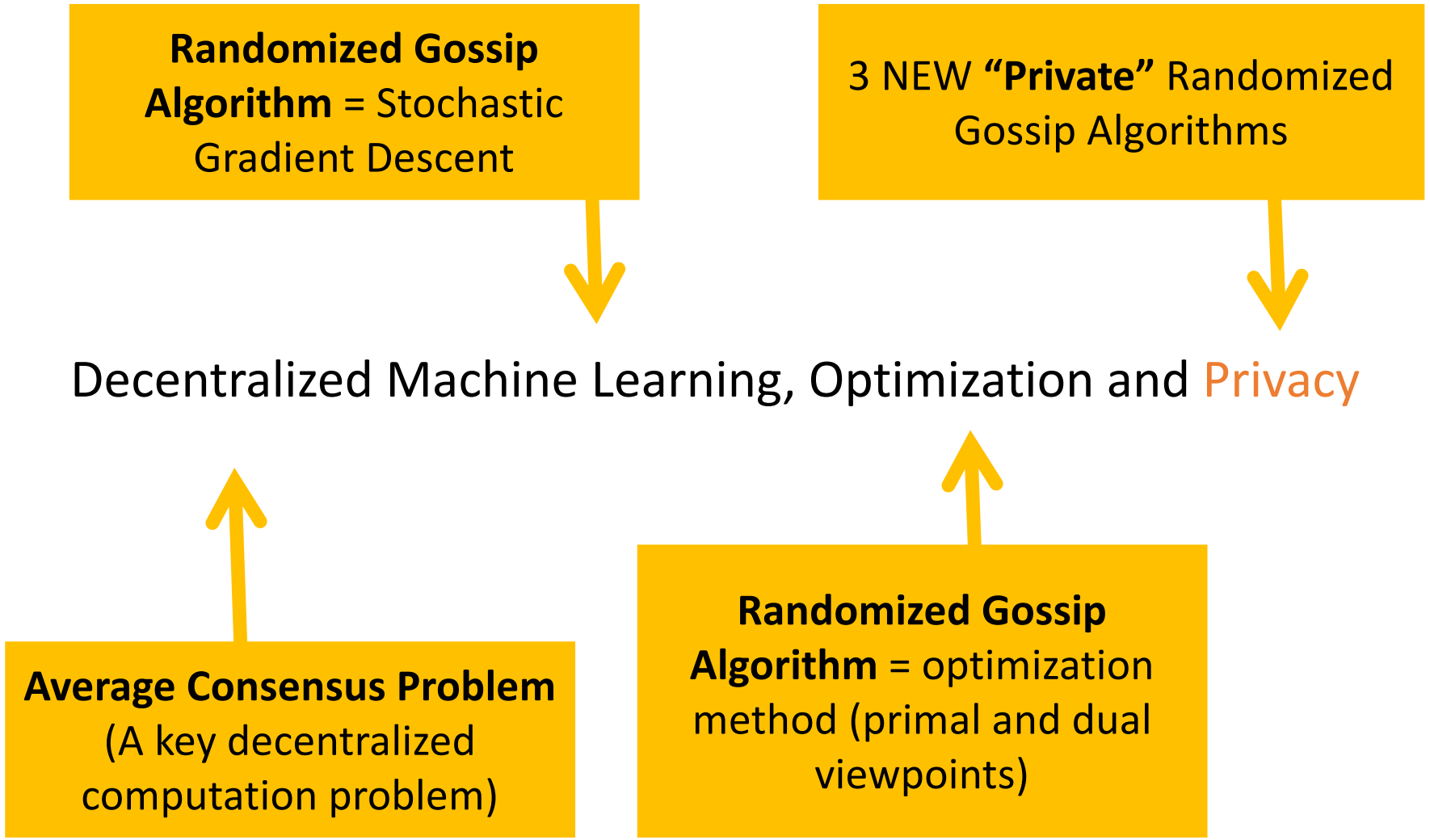
This Talk vs Workshop Theme



This Talk vs Workshop Theme



This Talk vs Workshop Theme



Outline

Part 1: Average Consensus Problem

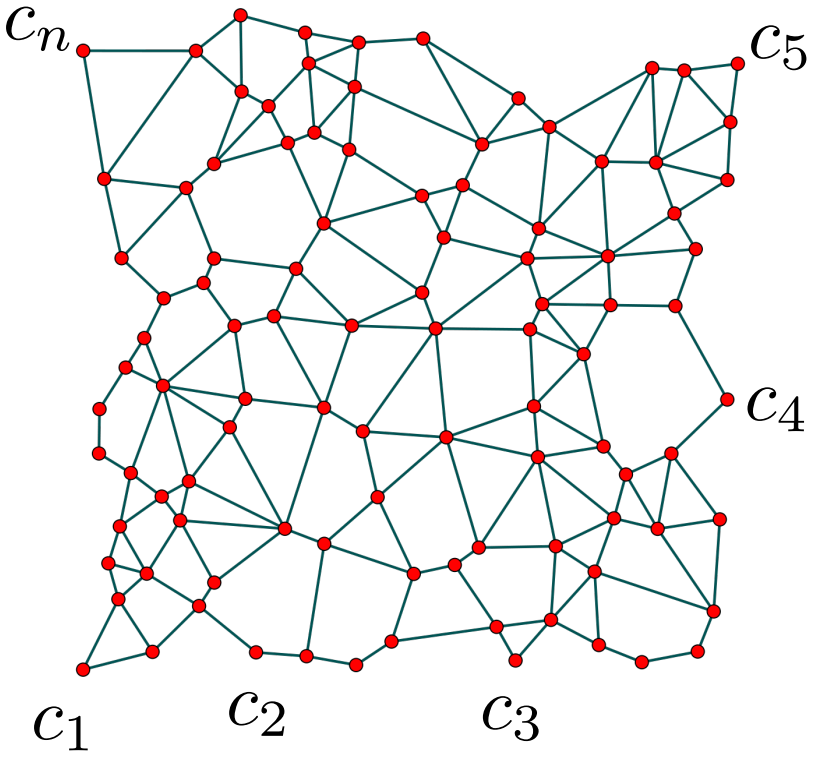
- The Problem
- Applications
- Standard Randomized Gossip Algorithm
- Connections to Optimization

Part 2: Privacy Preserving Average Consensus

- Binary Oracle
- ϵ -Gap Oracle
- Controlled Noise Insertion
- Theory
- Experiments

Part 1
Average Consensus
&
Randomized Gossip Algorithm

The Average Consensus Problem



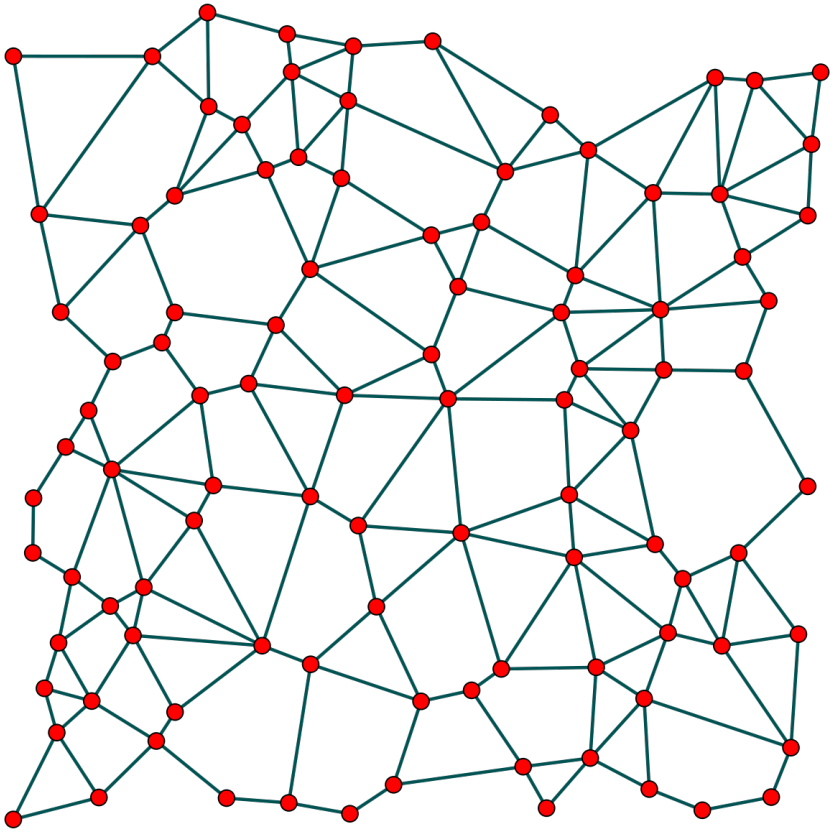
Undirected & connected graph

- *n* nodes
- *m* edges (communication links)
- node *i* stores private value c_i

Average Consensus Problem:

Compute the average of the values $c_1, c_2, \ldots, c_n$
Communication links = edges

Applications



Social network mining

Nodes: people
Edges: friendship
Task: **Compute average salary**

Sensor network computations

Nodes: sensors
Edges: close-by sensors
Task: **Compute average temperature**

Federated averaging/learning

Nodes: mobile devices
Edges: communication links
Task: **Compute average ML model**

Federated Learning

(Keith Bonawitz will talk about this)

Google Research Blog

The latest news from Research at Google

Federated Learning: Collaborative Machine Learning without Centralized Training Data

Thursday, April 06, 2017

Posted by Brendan McMahan and Daniel Ramage, Research Scientists

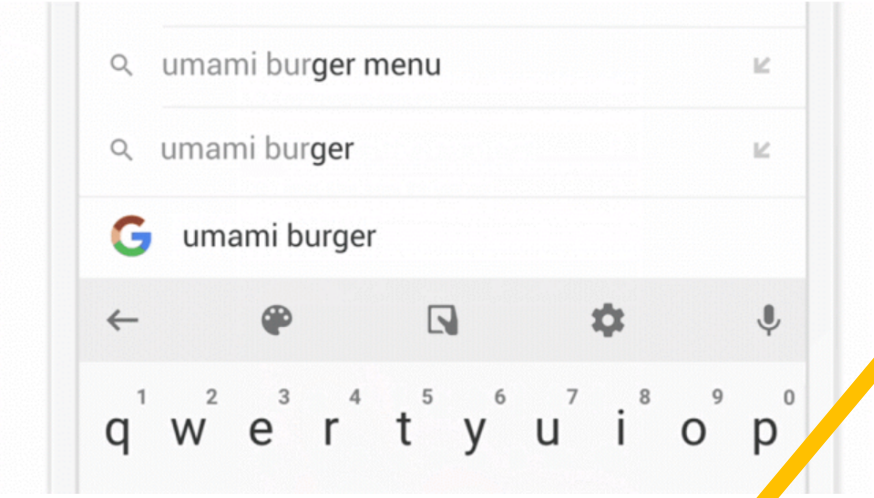
Standard machine learning approaches require centralizing the training data on one machine or in a datacenter. And Google has built one of the most secure and robust cloud infrastructures for processing this data to make our services better. Now for models trained from user interaction with mobile devices, we're introducing an additional approach: *Federated Learning*.

Federated Learning enables mobile phones to collaboratively learn a shared prediction model while keeping all the training data on device, decoupling the ability to do machine learning from the need to store the data in the cloud. This goes beyond the use of local models that make predictions on mobile devices (like the Mobile Vision API and On-Device Smart Reply) by bringing model *training* to the device as well.

It works like this: your device downloads the current model, improves it by learning from data on your phone, and then summarizes the changes as a small focused update. Only this update to the model is sent to the cloud, using encrypted communication, where it is immediately averaged with other user updates to improve the shared model. All the training data remains on your device, and no individual updates are stored in the cloud.

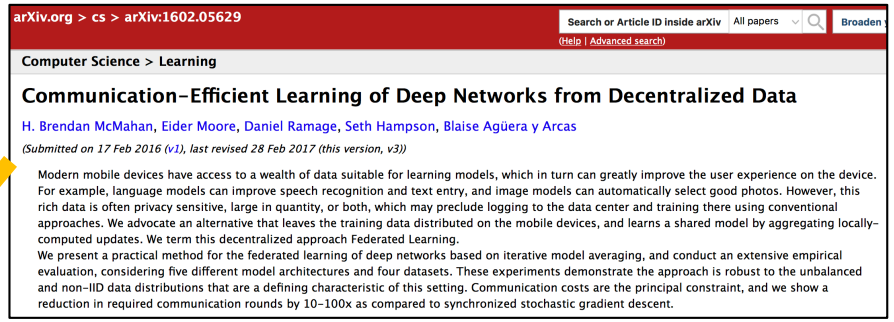
Federated Learning allows for smarter models, lower latency, and less power consumption, all while ensuring privacy. And this approach has another immediate benefit: in addition to providing an update to the shared model, the improved model on your phone can also be used immediately, powering experiences personalized by the way you use your phone.

We're currently testing Federated Learning in Gboard on Android, the Google Keyboard. When Gboard shows a suggested query, your phone locally stores information about the current context and whether you clicked the suggestion. Federated Learning processes that history on-device to suggest improvements to the next iteration of Gboard's query suggestion model.



To make Federated Learning possible, we had to overcome many algorithmic and technical challenges. In a typical machine learning system, an optimization algorithm like Stochastic Gradient Descent (SGD) runs on a large dataset partitioned homogeneously across servers in the cloud. Such highly iterative algorithms require low-latency, high-throughput connection to the training data. But in the Federated Learning setting, the data is distributed across millions of devices in a highly uneven fashion. In addition, these devices have significantly higher-latency, lower-throughput connections and are only intermittently available for training.

These bandwidth and latency limitations motivate our Federated Averaging algorithm, which can train deep networks using 10-100x less communication compared to a naive federated version of SGD. The key idea is to use the powerful processors in modern mobile devices to compute higher quality updates than simple gradient steps. Since it takes fewer iterations of high-quality updates to produce a good model, training can use much less communication. As upload speeds are typically much slower than download speeds, we also developed a novel way to reduce upload communication costs up to another 100x by compressing updates using random rotations and quantization. While these approaches are focused on training deep networks, we've also designed algorithms for high-dimensional sparse convex models which excel on problems like click-through-rate prediction.


arXiv.org > cs > arXiv:1602.05629

Search or Article ID inside arXiv | All papers | Broaden

(Help | Advanced search)

Computer Science > Learning

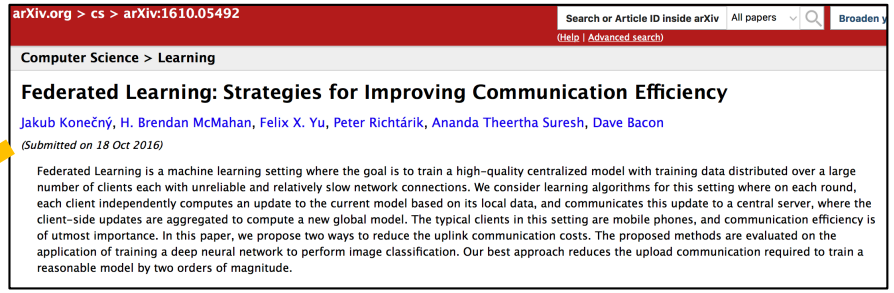
Communication-Efficient Learning of Deep Networks from Decentralized Data

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Agüera y Arcas

(Submitted on 17 Feb 2016 (v1), last revised 28 Feb 2017 (this version, v3))

Modern mobile devices have access to a wealth of data suitable for learning models, which in turn can greatly improve the user experience on the device. For example, language models can improve speech recognition and text entry, and image models can automatically select good photos. However, this rich data is often privacy sensitive, large in quantity, or both, which may preclude logging to the data center and training there using conventional approaches. We advocate an alternative that leaves the training data distributed on the mobile devices, and learns a shared model by aggregating locally-computed updates. We term this decentralized approach Federated Learning.

We present a practical method for the federated learning of deep networks based on iterative model averaging, and conduct an extensive empirical evaluation, considering five different model architectures and four datasets. These experiments demonstrate the approach is robust to the unbalanced and non-IID data distributions that are a defining characteristic of this setting. Communication costs are the principal constraint, and we show a reduction in required communication rounds by 10-100x as compared to synchronized stochastic gradient descent.


arXiv.org > cs > arXiv:1610.05492

Search or Article ID inside arXiv | All papers | Broaden

(Help | Advanced search)

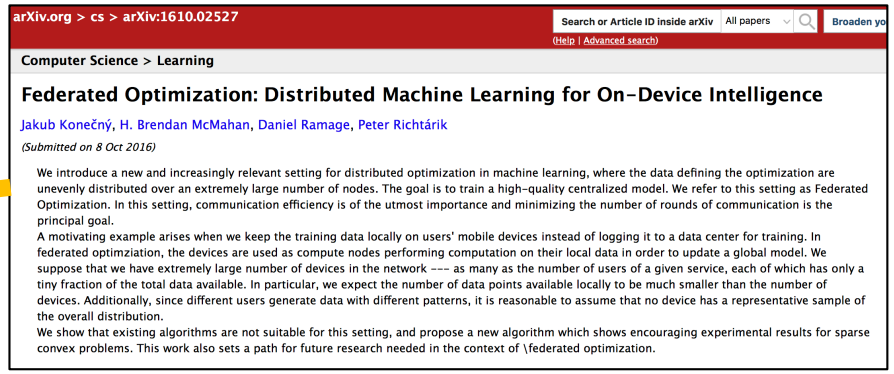
Computer Science > Learning

Federated Learning: Strategies for Improving Communication Efficiency

Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, Dave Bacon

(Submitted on 18 Oct 2016)

Federated Learning is a machine learning setting where the goal is to train a high-quality centralized model with training data distributed over a large number of clients each with unreliable and relatively slow network connections. We consider learning algorithms for this setting where on each round, each client independently computes an update to the current model based on its local data, and communicates this update to a central server, where the client-side updates are aggregated to compute a new global model. The typical clients in this setting are mobile phones, and communication efficiency is of utmost importance. In this paper, we propose two ways to reduce the uplink communication costs. The proposed methods are evaluated on the application of training a deep neural network to perform image classification. Our best approach reduces the upload communication required to train a reasonable model by two orders of magnitude.


arXiv.org > cs > arXiv:1610.02527

Search or Article ID inside arXiv | All papers | Broaden yo

(Help | Advanced search)

Computer Science > Learning

Federated Optimization: Distributed Machine Learning for On-Device Intelligence

Jakub Konečný, H. Brendan McMahan, Daniel Ramage, Peter Richtárik

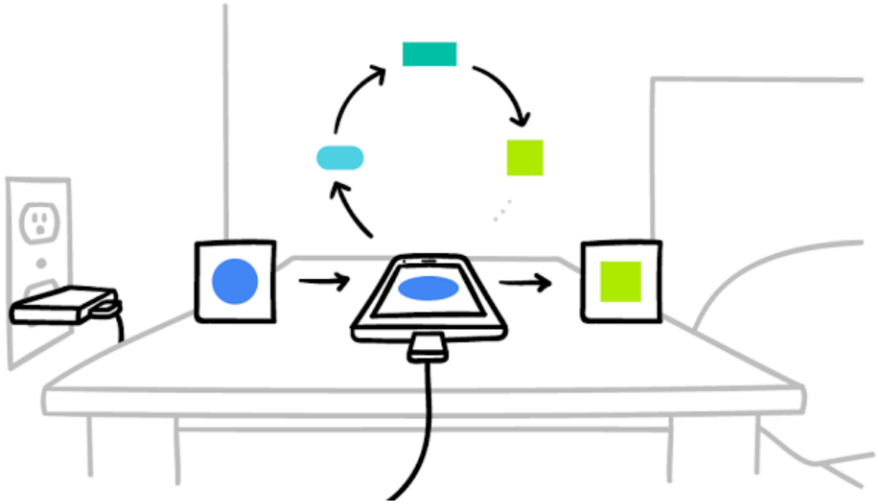
(Submitted on 8 Oct 2016)

We introduce a new and increasingly relevant setting for distributed optimization in machine learning, where the data defining the optimization are unevenly distributed over an extremely large number of nodes. The goal is to train a high-quality centralized model. We refer to this setting as Federated Optimization. In this setting, communication efficiency is of the utmost importance and minimizing the number of rounds of communication is the principal goal.

A motivating example arises when we keep the training data locally on users' mobile devices instead of logging it to a data center for training. In federated optimziation, the devices are used as compute nodes performing computation on their local data in order to update a global model. We suppose that we have extremely large number of devices in the network --- as many as the number of users of a given service, each of which has only a tiny fraction of the total data available. In particular, we expect the number of data points available locally to be much smaller than the number of devices. Additionally, since different users generate data with different patterns, it is reasonable to assume that no device has a representative sample of the overall distribution.

We show that existing algorithms are not suitable for this setting, and propose a new algorithm which shows encouraging experimental results for sparse convex problems. This work also sets a path for future research needed in the context of \federated optimization.

Deploying this technology to millions of heterogenous phones running Gboard requires a sophisticated technology stack. On device training uses a miniature version of TensorFlow. Careful scheduling ensures training happens only when the device is idle, plugged in, and on a free wi connection, so there is no impact on the phone's performance.



Your phone participates in Federated Learning only when it won't negatively impact your experience.

The system then needs to communicate and aggregate the model updates in a secure, efficient, scalable, and fault-tolerant way. It's only the combination of research with this infrastructure that makes the benefits of Federated Learning possible.

Federated learning works without the need to store user data in the cloud, but we're not stopping there. We've developed a Secure Aggregation protocol th cryptographic techniques so a coordinating server can only decrypt the average update if 100 or 1000s of users have participated — no individual phone's update can be inspected before averaging. It's the first protocol of its kind that is practical for deep-network-sized problems and real-world connectivity constraints. We designed Federated Averaging so the coordinating server only needs the average update, which allows Secure Aggregation to be used; however the protocol is general and can be applied to other problems as well. We're working hard on a production implementation of this protocol and expect to deploy it for Federated Learning applications in the near future.

Practical Secure Aggregation for Privacy-Preserving Machine Learning

Keith Bonawitz*, Vladimir Ivanov*, Ben Kreuter*, Antonio Marcedone†*,
H. Brendan McMahan*, Sarvar Patel*, Daniel Ramage*, Aaron Segal* and Karn Seth*
*Google, 1600 Amphitheatre Parkway Mountain View, California 94043
{bonawitz, vlivan, benkreuter, mcmahan, sarvar, dramage, asegal, karn}@google.com
†Cornell University, Ithaca, New York 14853
marcedone@cs.cornell.edu

More on Averaging

Ananda T. Suresh, Felix X. Yu, H. Brendan McMahan, and Sanjiv Kumar
Distributed Mean Estimation with Limited Communication
arXiv:1611.00429, 2016

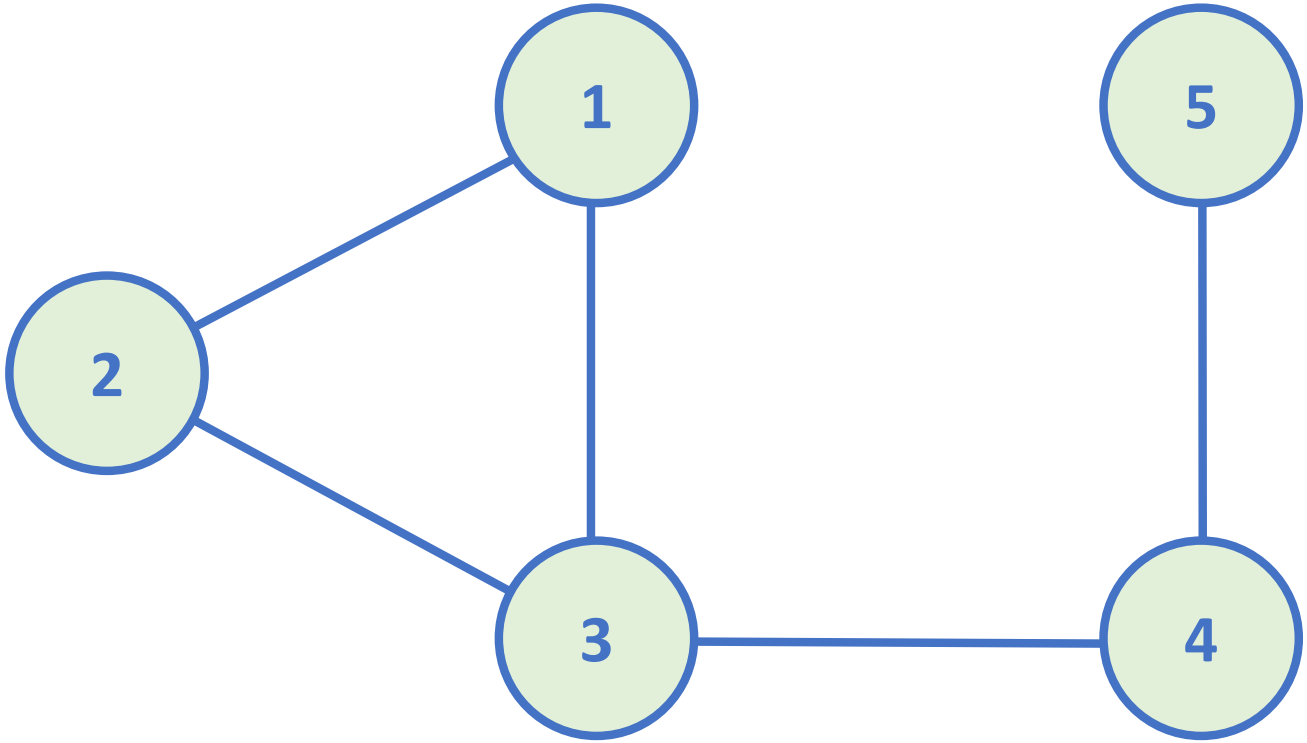
Jakub Konečný and Peter Richtárik
Randomized Distributed Mean Estimation: Accuracy vs Communication
arXiv:1611.07555, 2016

The (Standard) Randomized Gossip Algorithm

Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah
Randomized Gossip Algorithms
IEEE Transactions on Information Theory 52.6 (2006), pp. 2508–2530

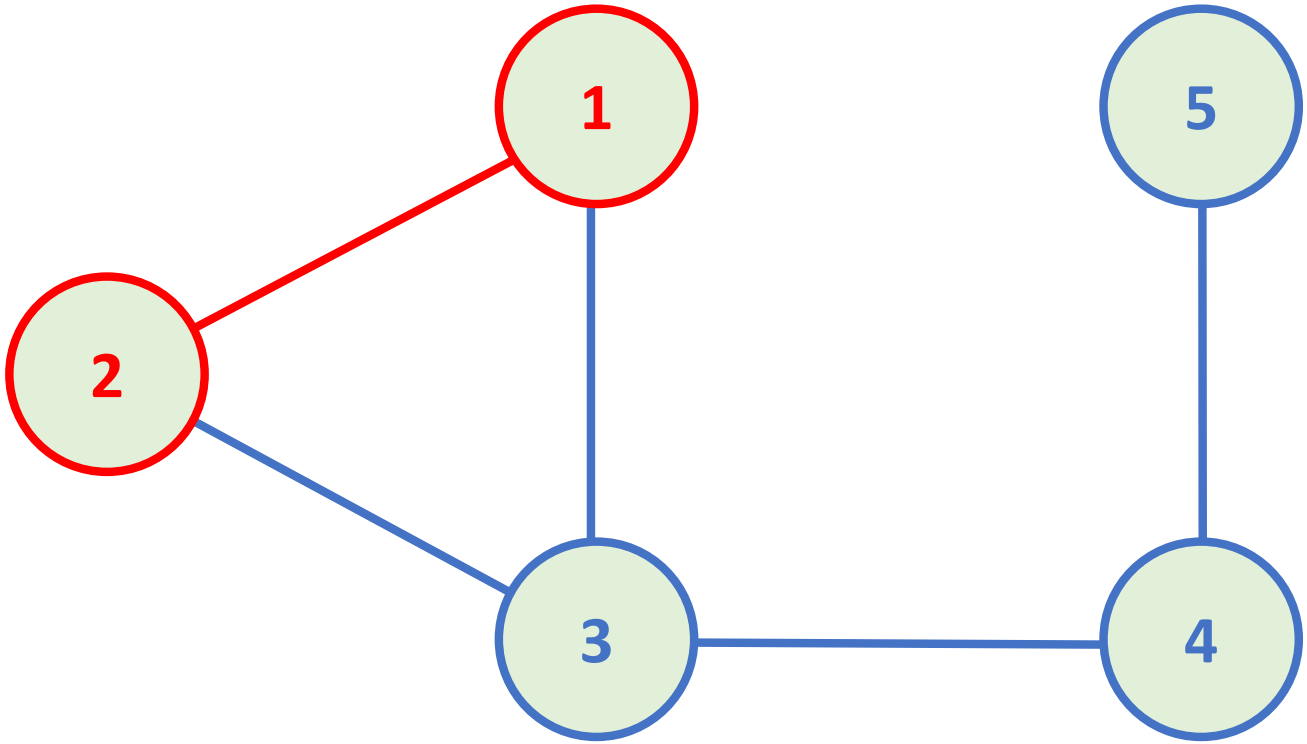
BGPS 2006

Randomized (Pairwise) Gossip Algorithm



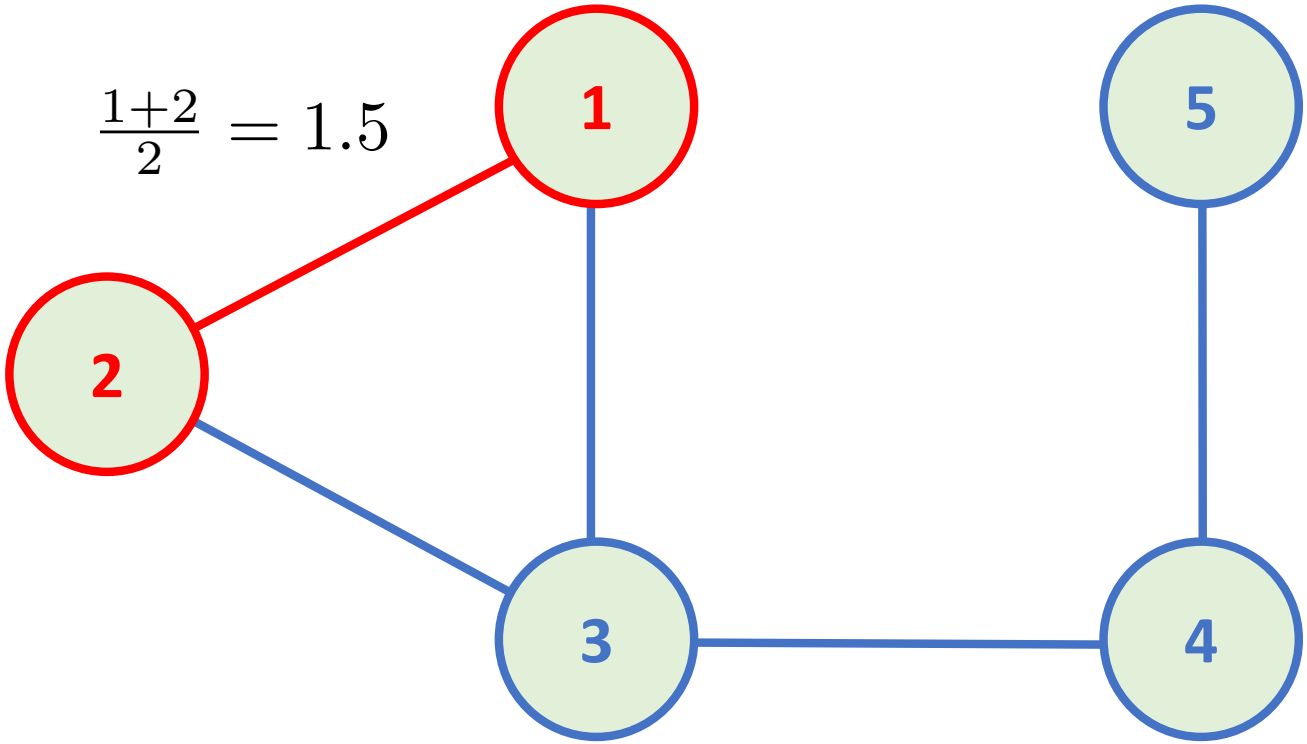
$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized (Pairwise) Gossip Algorithm



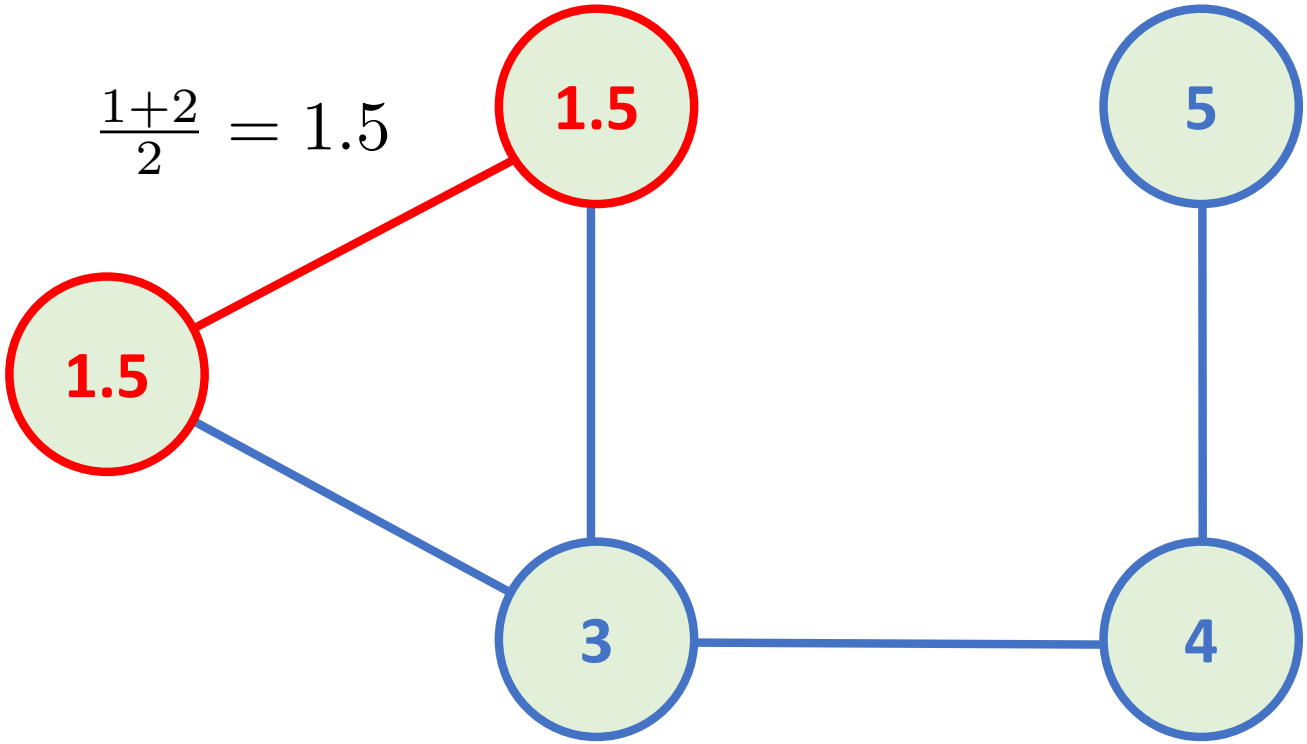
$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized (Pairwise) Gossip Algorithm



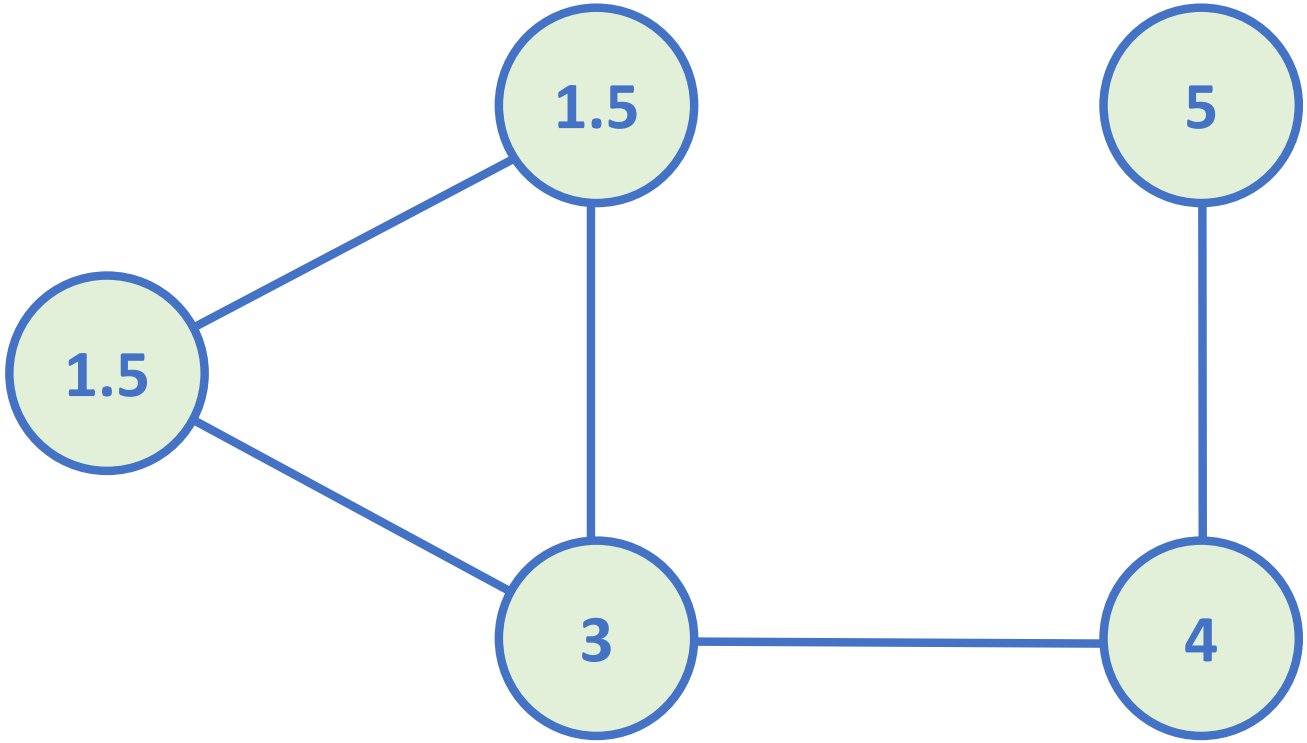
$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized (Pairwise) Gossip Algorithm



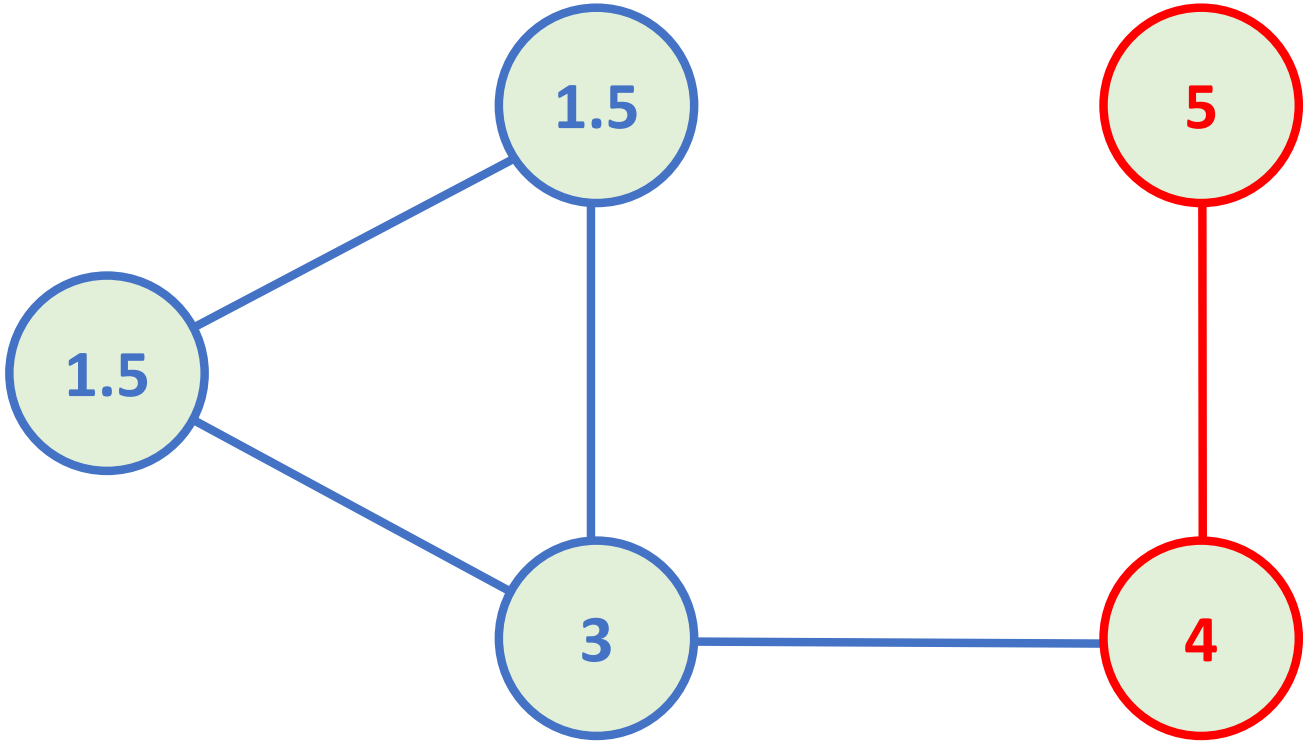
$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized (Pairwise) Gossip Algorithm



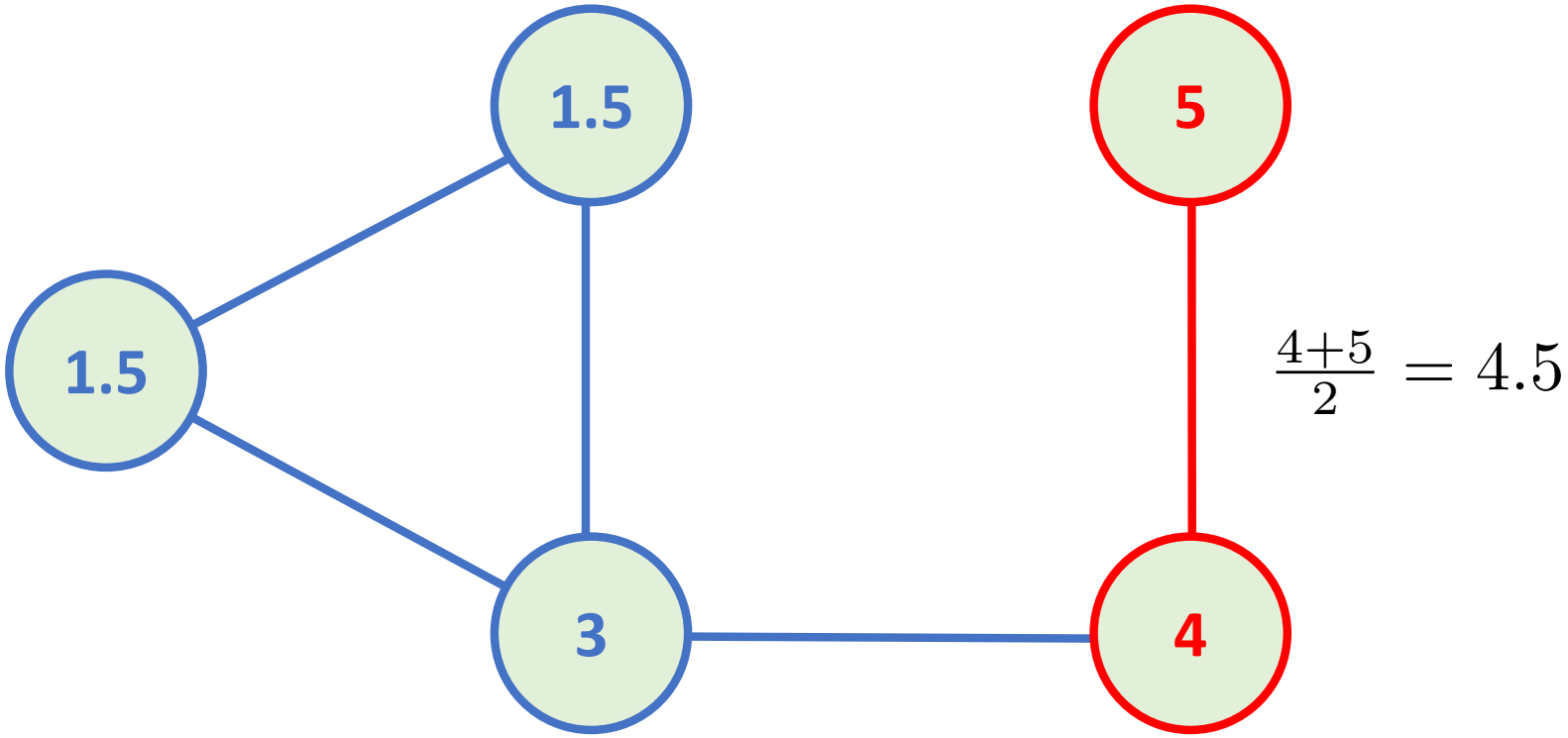
$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized (Pairwise) Gossip Algorithm



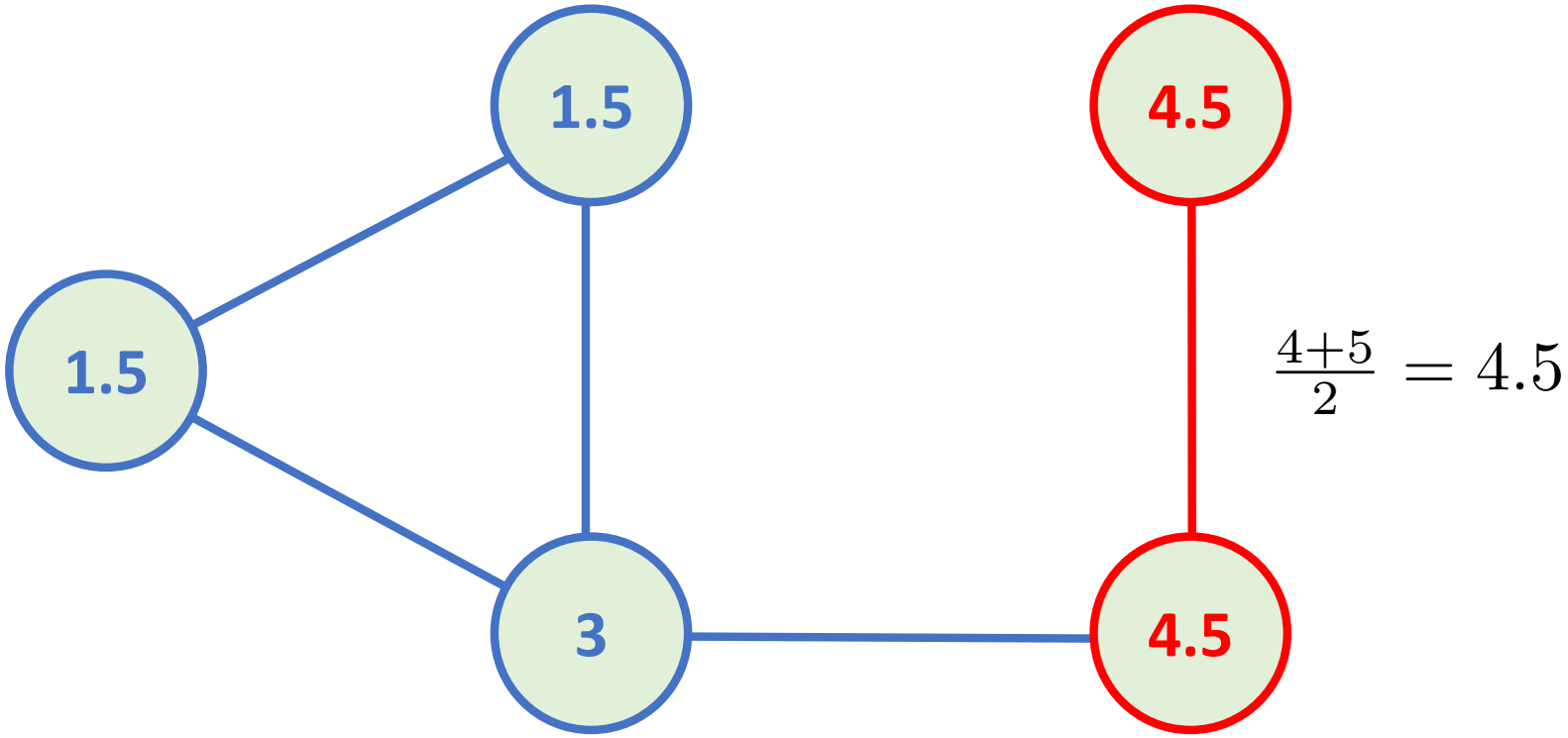
$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized (Pairwise) Gossip Algorithm



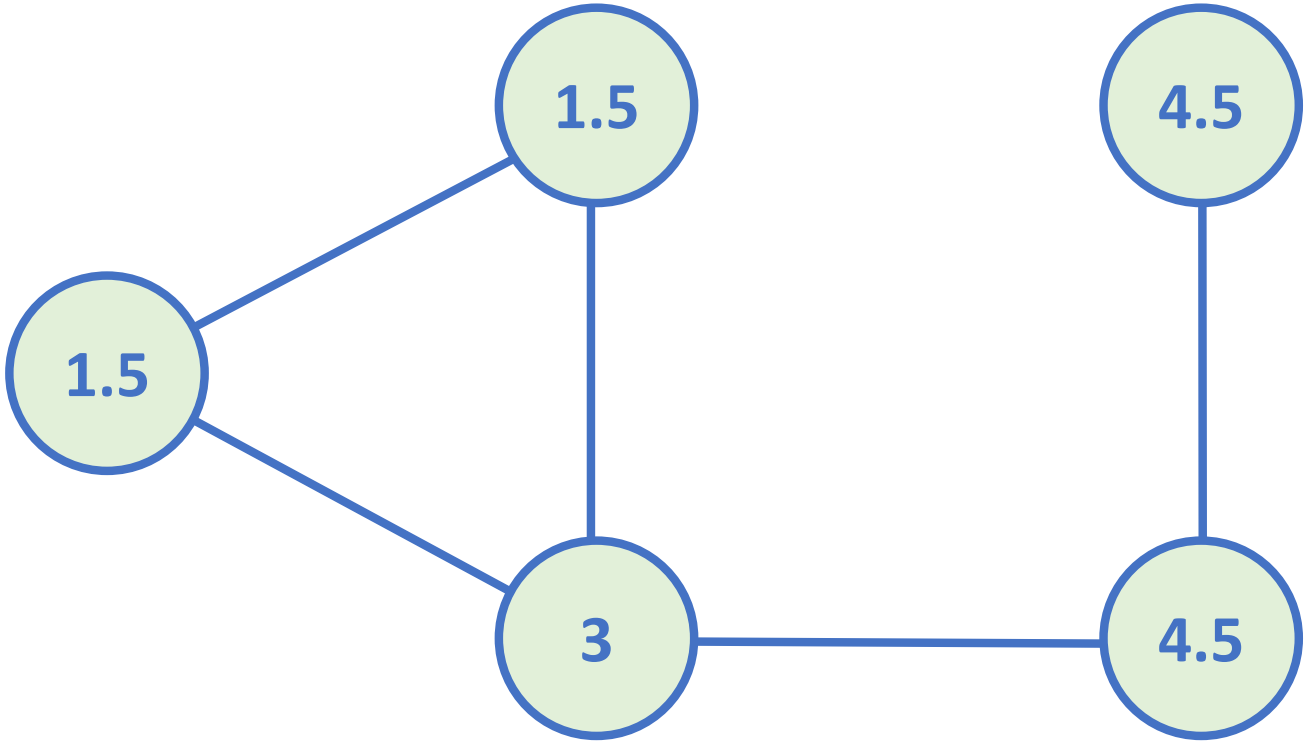
$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized (Pairwise) Gossip Algorithm



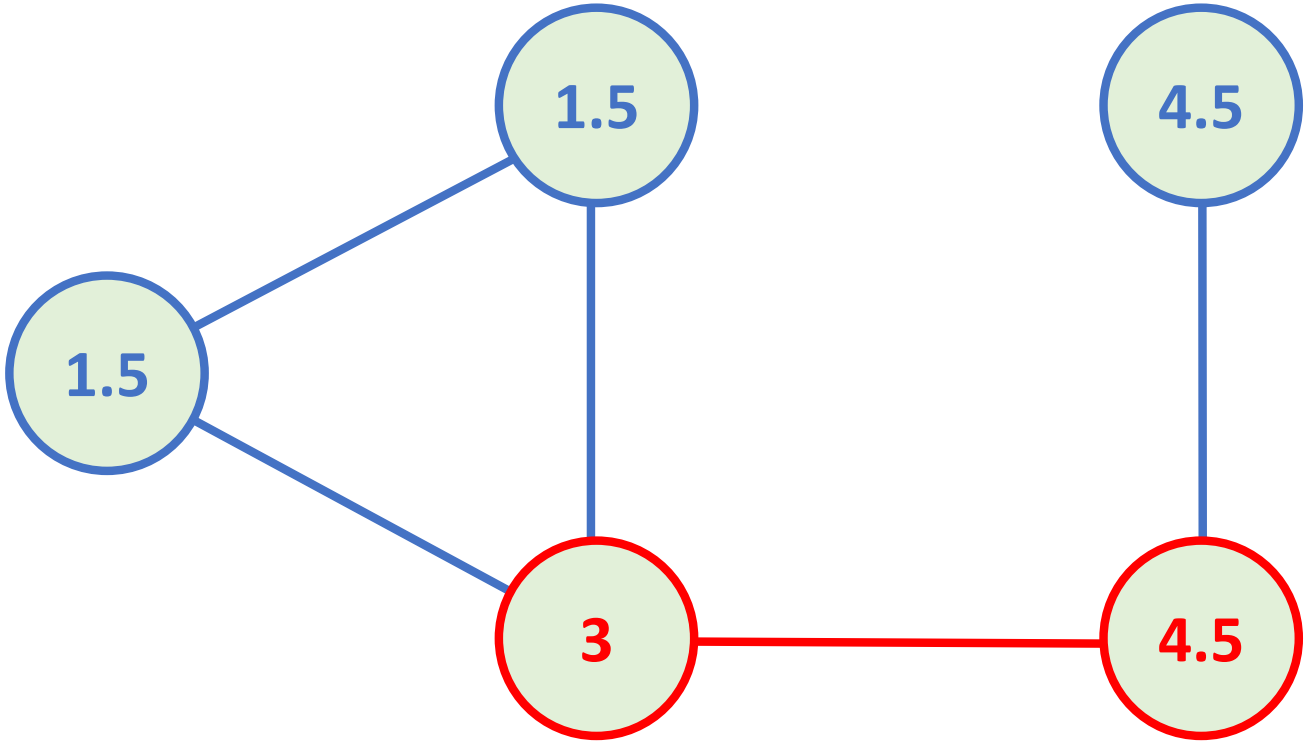
$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized (Pairwise) Gossip Algorithm



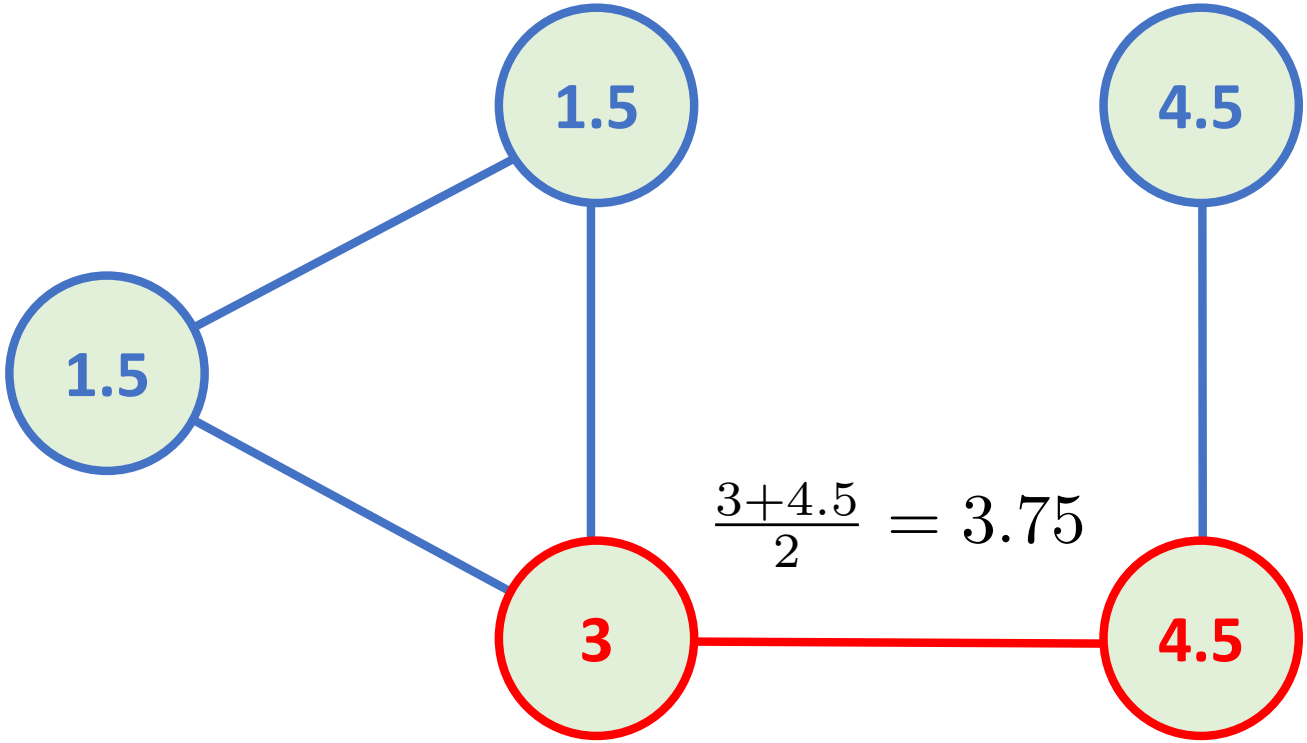
$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized (Pairwise) Gossip Algorithm



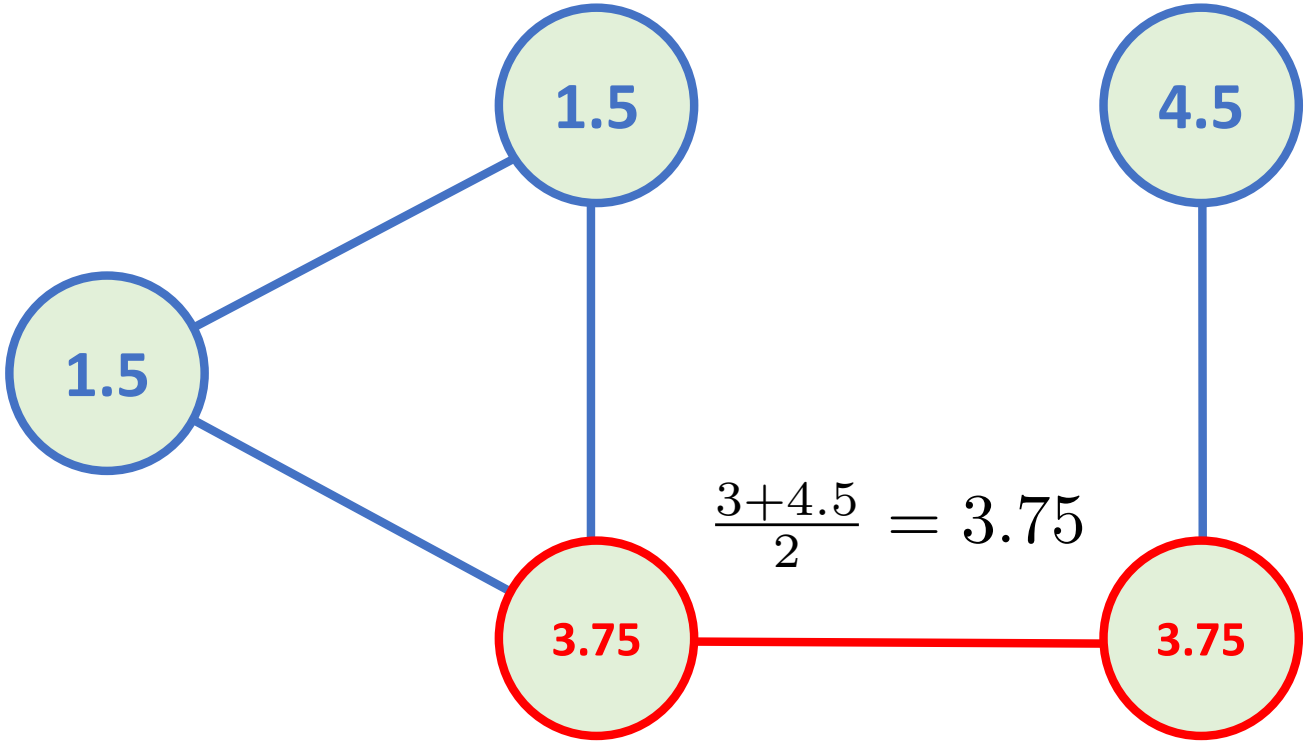
$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized (Pairwise) Gossip Algorithm



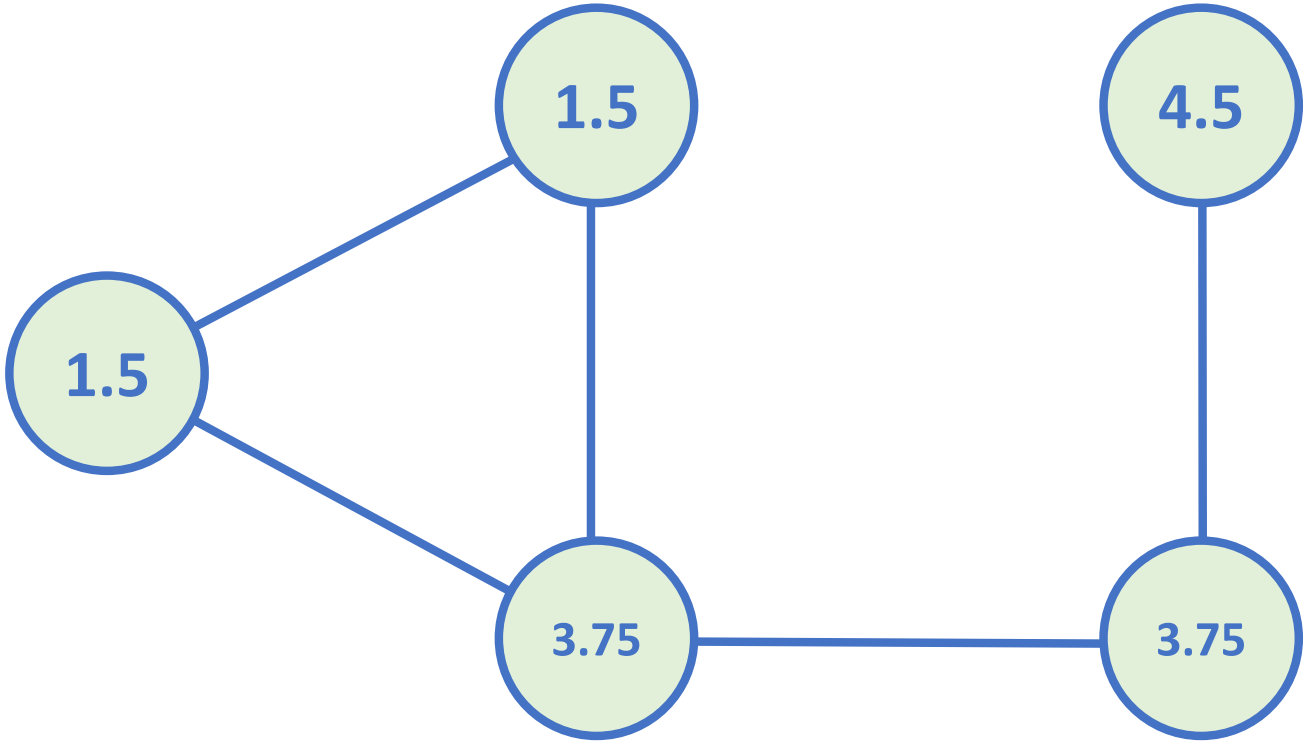
$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized (Pairwise) Gossip Algorithm



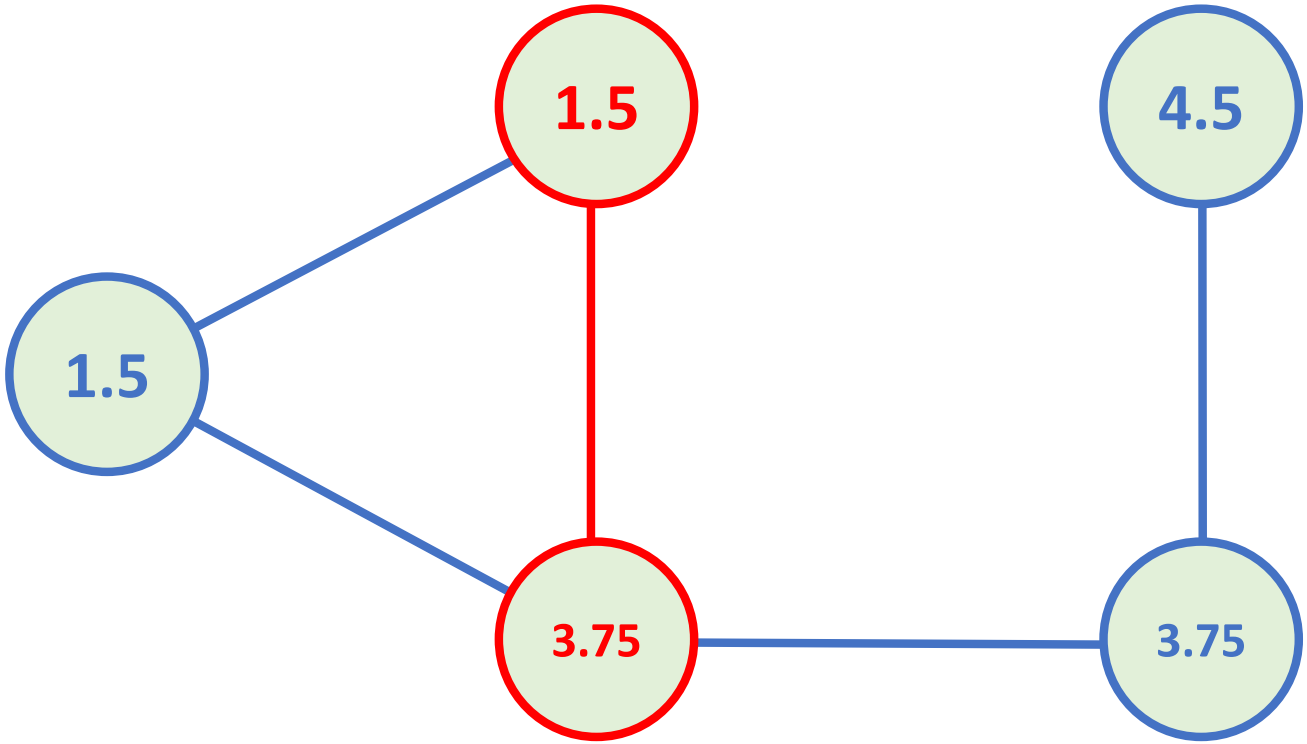
$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized (Pairwise) Gossip Algorithm



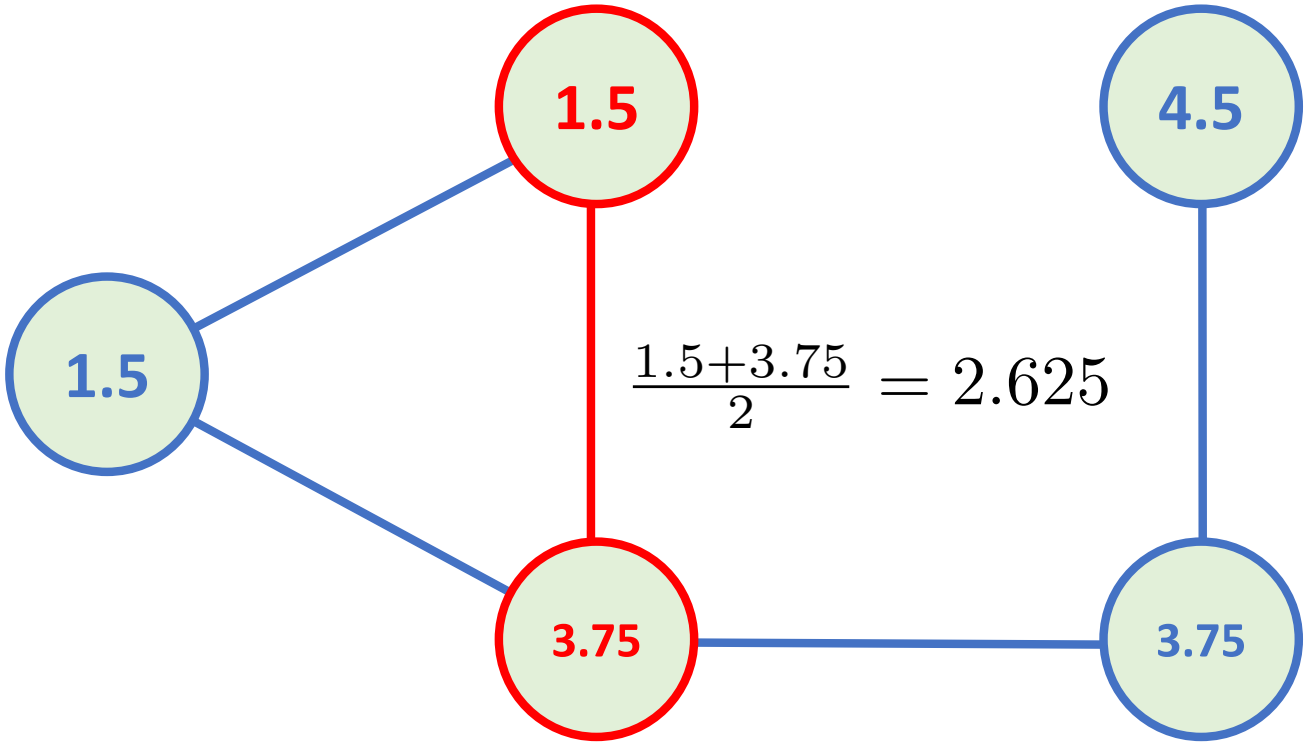
$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized (Pairwise) Gossip Algorithm



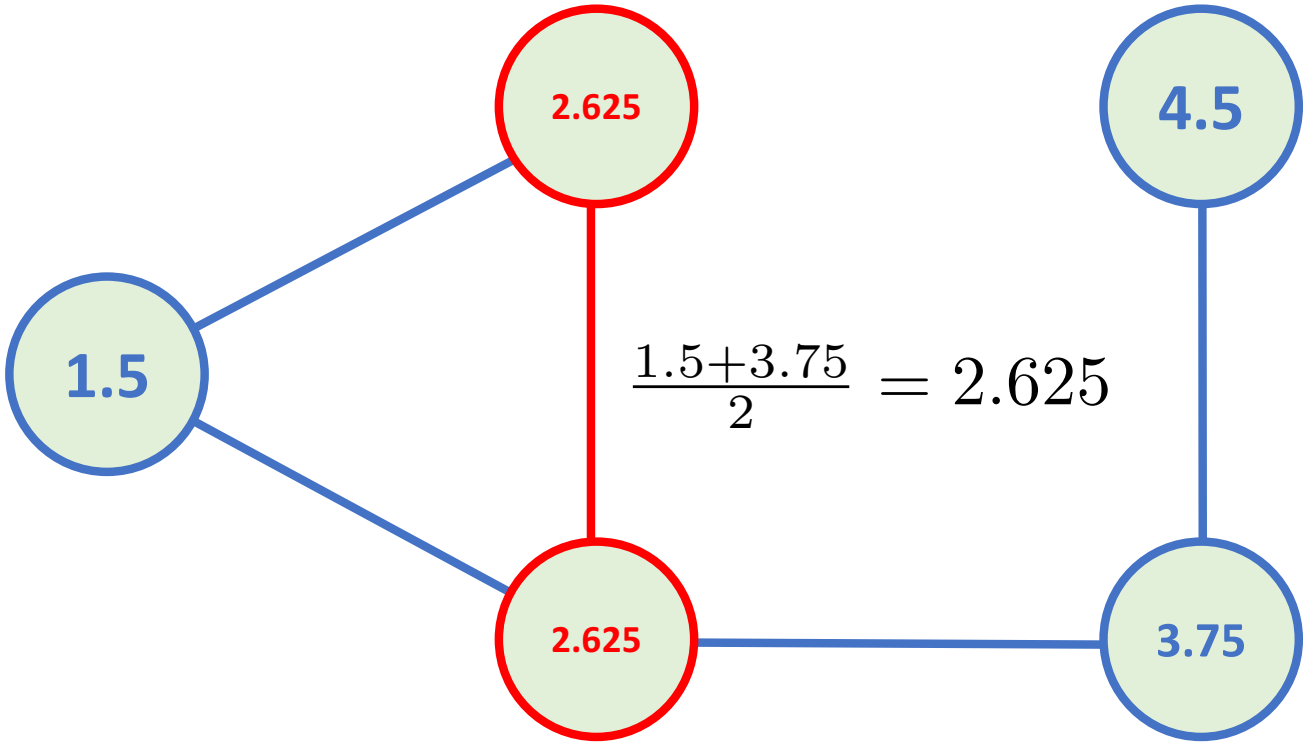
$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized (Pairwise) Gossip Algorithm



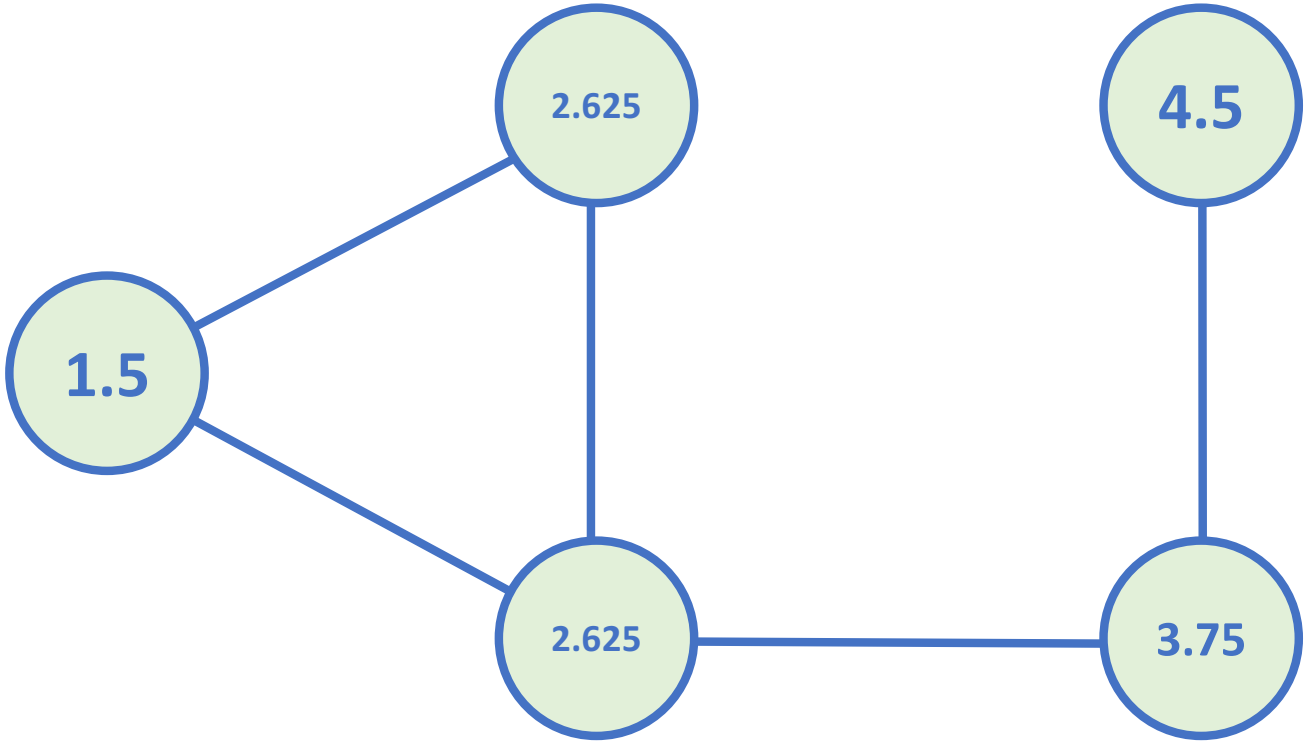
$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized (Pairwise) Gossip Algorithm



$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized (Pairwise) Gossip Algorithm



$$\text{Average} = \frac{1+2+3+4+5}{5} = 3$$

Randomized Gossip Algorithm (Formalized)

1. Initialize $x_i^0 = c_i$ for all $i = 1, 2, \ldots, n$
2. For $t \geq 0$ iterate:
   (a) Pick a random edge (i, j)
   (b) Set $x_i^{t+1} \leftarrow \frac{x_i^t + x_j^t}{2}$
   (c) Set $x_j^{t+1} \leftarrow \frac{x_i^t + x_j^t}{2}$
   (d) Set $x_u^{t+1} = x_u^t$ for all $u \notin \{i, j\}$

Randomized Gossip
=
Optimization Algorithm

Averaging via Optimization

nodes

Private node values

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\sum_{i=1}^{n}(x_i - c_i)^2 \\ \text{subject to} \quad & x_1 = x_2 = \cdots = x_n \end{aligned}$$

We want all nodes to find the same number; the **average of the private values:**

$$\bar{c} \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} c_i$$

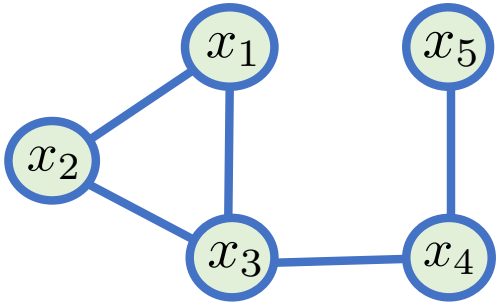
Lemma

$$x^* = \bar{c} \cdot \mathbf{1}$$

The solution is $x_i^* = \bar{c}$ for all i

Encoding Constraints via the Incidence Matrix of the Graph

$$x_1 = x_2 = x_3 = x_4 = x_5$$

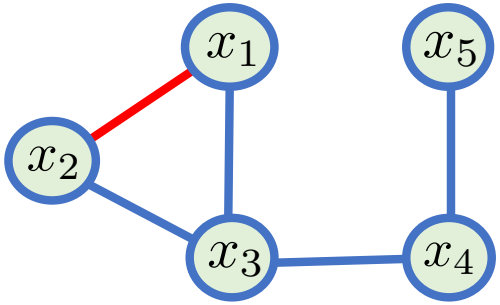


Incidence matrix of the graph

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Encoding Constraints via the Incidence Matrix of the Graph

$$x_1 = x_2 = x_3 = x_4 = x_5$$

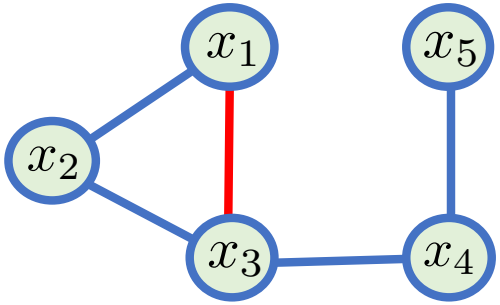


Incidence matrix of the graph

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Encoding Constraints via the Incidence Matrix of the Graph

$$x_1 = x_2 = x_3 = x_4 = x_5$$

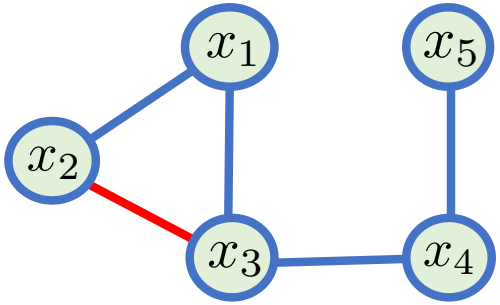


Incidence matrix of the graph

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Encoding Constraints via the Incidence Matrix of the Graph

$$x_1 = x_2 = x_3 = x_4 = x_5$$

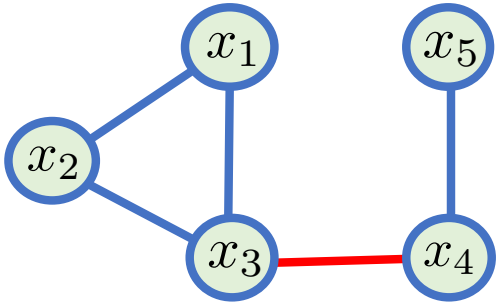


Incidence matrix of the graph

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Encoding Constraints via the Incidence Matrix of the Graph

$$x_1 = x_2 = x_3 = x_4 = x_5$$

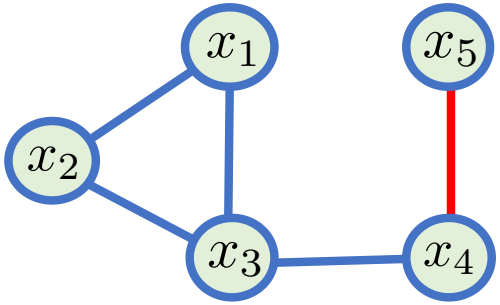


Incidence matrix of the graph

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Encoding Constraints via the Incidence Matrix of the Graph

$$x_1 = x_2 = x_3 = x_4 = x_5$$



Incidence matrix of the graph

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Optimization Problem

Recall:

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Incidence matrix of the graph

$$\text{minimize} \quad P(x) \quad \text{subject to} \quad Ax = 0,\ x \in \mathbb{R}^n$$

$$P(x) = \frac{1}{2}\|x - c\|^2 = \frac{1}{2}\sum_{i=1}^{n}(x_i - c_i)^2$$

More Reading

Robert Mansel Gower and P.R.
Randomized Iterative Methods for Linear Systems
SIAM Journal on Matrix Analysis and Applications 36(4):1660-1690, 2015

GR 2015a

Robert Mansel Gower and P.R.
Stochastic Dual Ascent for Solving Linear Systems
arXiv:1512.06890, 2015

GR 2015b

Randomized Gossip as an optimization method + duality

Nicolas Loizou and P.R.
A New Perspective on Randomized Gossip Algorithms
In *Proceedings of The 4th IEEE Global Conference on Signal Processing*, 2016

LR 2016

P.R. and Martin Takáč
Stochastic Reformulations of Linear Systems: Algorithms and Convergence Theory
arXiv:1706.01108, 2017

RT 2017

Randomized Gossip is:

- Stochastic Gradient Descent
- Stochastic Newton Method
- Stochastic Proximal Point Method
- Stochastic Fixed Point Method
- Stochastic Projection Method

Randomized Gossip
=
Dual Optimization Algorithm

Dual Problem

Primal Problem

$$\min_{x \in \mathbb{R}^n} \left\{ P(x) \stackrel{\text{def}}{=} \frac{1}{2}\|x - c\|^2 \quad \text{subject to} \quad Ax = 0 \right\}$$

Incidence matrix

nodes

Vector of private values stored at the nodes

Dual Problem

$$\max_{y \in \mathbb{R}^m} D(y) \stackrel{\text{def}}{=} -c^\top A^\top y - \frac{1}{2}\|A^\top y\|^2$$

edges

Duality mapping: $x(y) \stackrel{\text{def}}{=} c + A^\top y$

$x(0) = c$

Lemma [GR 2015b]

$$D(y^*) - D(y) = \frac{1}{2}\|x^* - x(y)\|^2$$

Randomized Gossip: Dual View

Unit basis vector in $\mathbb{R}^m$ corresponding to edge (i,j)

Dual Method

$$y^{t+1} = y^t + \lambda^t e_{ij} \quad \text{where} \quad \lambda^t = \arg\max_{\lambda \in \mathbb{R}} D(y^t + \lambda e_{ij})$$

$y^0 = 0$

Theorem [GR 2015b]

Mapping the iterates of the dual method via the duality mapping

$$x^t \leftarrow x(y^t)$$

gives the standard randomized gossip method

Dual Theory

Algebraic connectivity of the graph, i.e., smallest nonzero eigenvalue of the Laplacian: $L = A^\top A$

Theorem [HKLRG 2017]

$$\mathbb{E}[D(y^*) - D(y^k)] \leq \left(1 - \frac{\alpha}{2m}\right)^k [D(y^*) - D(y^0)]$$

- Follows by applying the lemma: $D(y^*) - D(y) = \frac{1}{2}\|x^* - x(y)\|^2$
- First done in [GR 2015b]

Part 2
Privacy Preserving Randomized Gossip Algorithms

Private Gossip via Binary Oracle

Private Gossip via Binary Oracle

1. Initialize $x_i^0 = c_i$ for all $i = 1, 2, \ldots, n$
2. For $t \geq 0$ iterate:
   (a) Pick a random edge (i, j)
   (b) If $x_i^t \leq x_j^t$, set
   - $x_i^{t+1} \leftarrow x_i^t + \lambda^t$
   - $x_j^{t+1} \leftarrow x_j^t - \lambda^t$

   (c) If $x_i^t > x_j^t$, set
   - $x_i^{t+1} \leftarrow x_i^t - \lambda^t$
   - $x_j^{t+1} \leftarrow x_j^t + \lambda^t$

   (d) Set $x_u^{t+1} = x_u^t$ for all $u \notin \{i, j\}$

"Stepsize"

In (standard) randomized gossip we instead had:

$$\text{Set } x_i^{t+1} \leftarrow \frac{x_i^t + x_j^t}{2}$$
$$\text{Set } x_j^{t+1} \leftarrow \frac{x_i^t + x_j^t}{2}$$

Privacy protection

Node *i* only learns binary Information about *j:* whether his/her value is smaller or larger

Implementation

Secure multi-party protocol between nodes

Theory: General Stepsizes

Dual function

$$\beta^k = \sum_{t=0}^{k} \left(\lambda^t\right)^2$$

Theorem [HKLRG 2017]

$$\min_{t=0,1,\ldots,k} \mathbb{E}[L^t] \leq \frac{D(y^*) - D(y^0)}{\alpha^k} + \frac{\beta^k}{\alpha^k}$$

$$L^t \stackrel{\text{def}}{=} \frac{1}{m} \sum_{\text{edges } (i,j)} |x_i^t - x_j^t|$$

Algebraic connectivity

$$\alpha^k = \sum_{t=0}^{k} \lambda^t$$

Proposition [HKLRG 2017] $\sqrt{\alpha}\|x^t - x^*\| \leq mL^t \leq \sqrt{mn}\|x^t - x^*\|$

Theory: Constant Stepsizes

Corollary (constant stepsize)

$\mathcal{O}(1/k)$

error

$$\lambda^t = \lambda^0 > 0 \quad \Rightarrow \quad \min_{t=0,1,\ldots,k} \mathbb{E}[L^t] \leq \frac{D(y^*) - D(y^0)}{\lambda^0(k+1)} + \lambda^0$$

Corollary (optimal constant stepsize)

$\mathcal{O}(1/\sqrt{k})$

$$\lambda^t = \sqrt{\frac{D(y^*) - D(y^0)}{k+1}} \quad \Rightarrow \quad \min_{t=0,1,\ldots,k} \mathbb{E}[L^t] \leq 2\sqrt{\frac{D(y^*) - D(y^0)}{k+1}}$$

Theory: Adaptive Stepsizes

Impractical: global information is needed

Theorem [HKLRG 2017]

$$\lambda^t = f(x_1^t, x_2^t, \ldots, x_n^t)$$

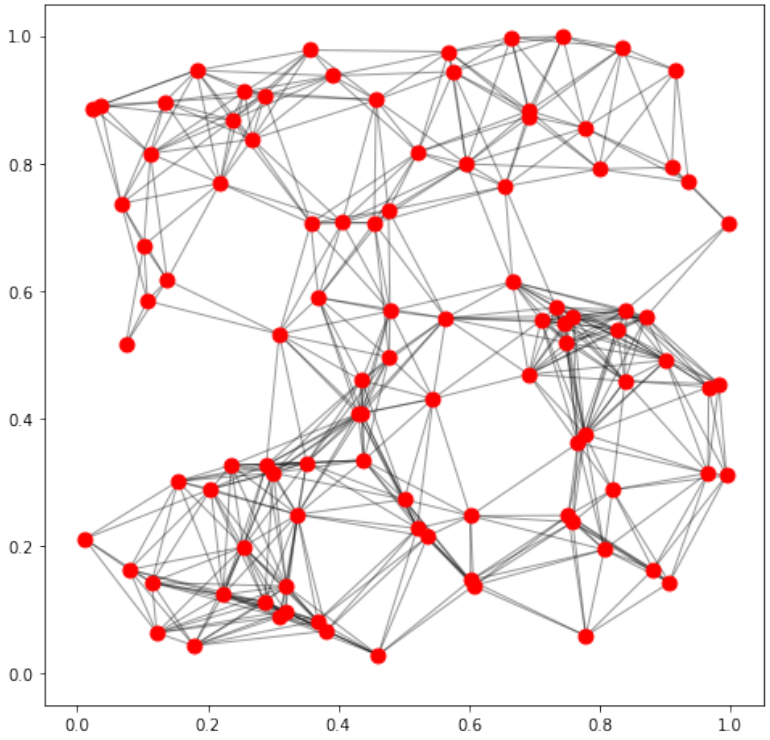
$$\mathbb{E}[\|x^t - x^*\|^2] \leq \left(1 - \frac{\alpha(G)}{2m^2}\right)^t \|x^0 - x^*\|^2$$

Gives a "bound" on the "limits" of the binary oracle

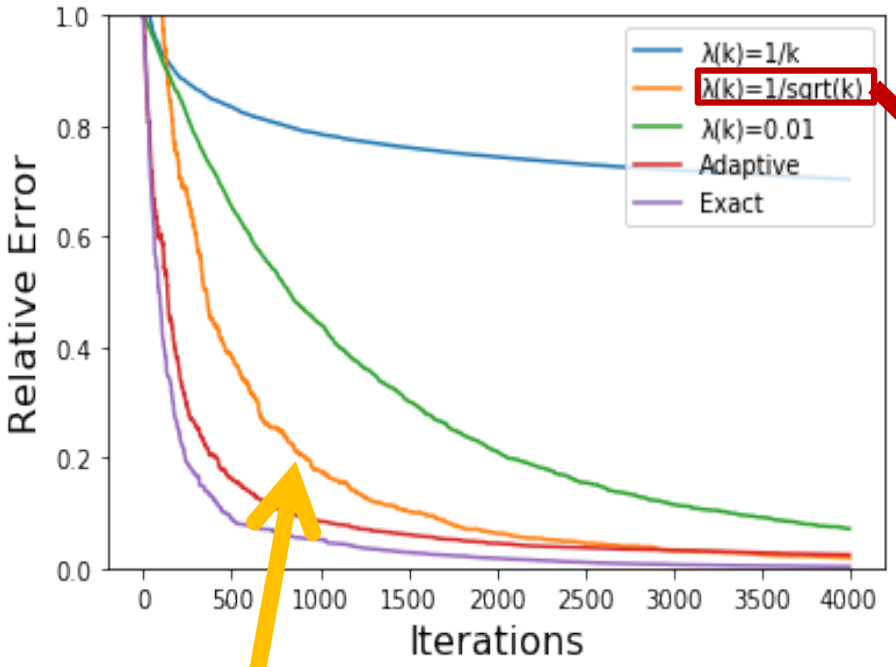
Standard randomized gossip has m instead of m^2

Experimental Setup

- 2D Random Geometric Graph on 100 nodes
- Important in wireless sensor network modelling
- Nodes placed uniformly in a square; edges between close-by nodes

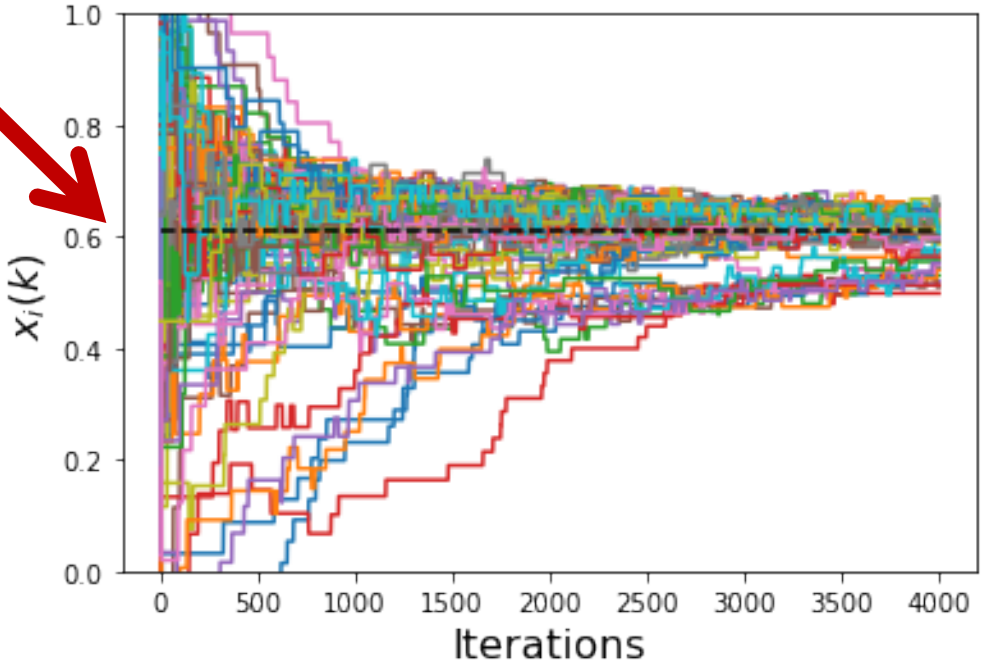


Experiment



Best non-adaptive stepsize

Evolution of values in nodes



Private Gossip via ϵ-Gap Oracle

Private Gossip via eps-Gap Oracle

1. Initialize $x_i^0 = c_i$ for all $i = 1, 2, \ldots, n$
2. For $t \geq 0$ iterate:
   (a) Pick a random edge (i, j)
   (b) If $x_i^t \leq x_j^t - \epsilon$, set
   - $x_i^{t+1} \leftarrow x_i^t + \epsilon/2$
   - $x_j^{t+1} \leftarrow x_j^t - \epsilon/2$

   (c) If $x_i^t > x_j^t + \epsilon$, set
   - $x_i^{t+1} \leftarrow x_i^t - \epsilon/2$
   - $x_j^{t+1} \leftarrow x_j^t + \epsilon/2$

   (d) Set $x_u^{t+1} = x_u^t$ for all $u \notin \{i, j\}$

Binary oracle had 0 here

Binary oracle had λ^t here

Privacy protection

Node *i* only learns that his/her value is larger or smaller by a fixed margin than that of node *j*

Implementation

Secure multi-party protocol between nodes

Theory: General Stepsizes

Dual function

Theorem [HKLRG 2017]

$\mathcal{O}(1/k)$

$$\mathbb{E}\left[\frac{1}{k}\sum_{t=0}^{k-1}\Delta^t(\epsilon)\right] \leq \frac{4\left(D(y^*)-D(y^0)\right)}{k\epsilon^2}$$

$$\Delta^t(\epsilon) \overset{\text{def}}{=} \frac{1}{m}\sum_{\text{edges } (i,j)} \Delta_{ij}^t(\epsilon)$$

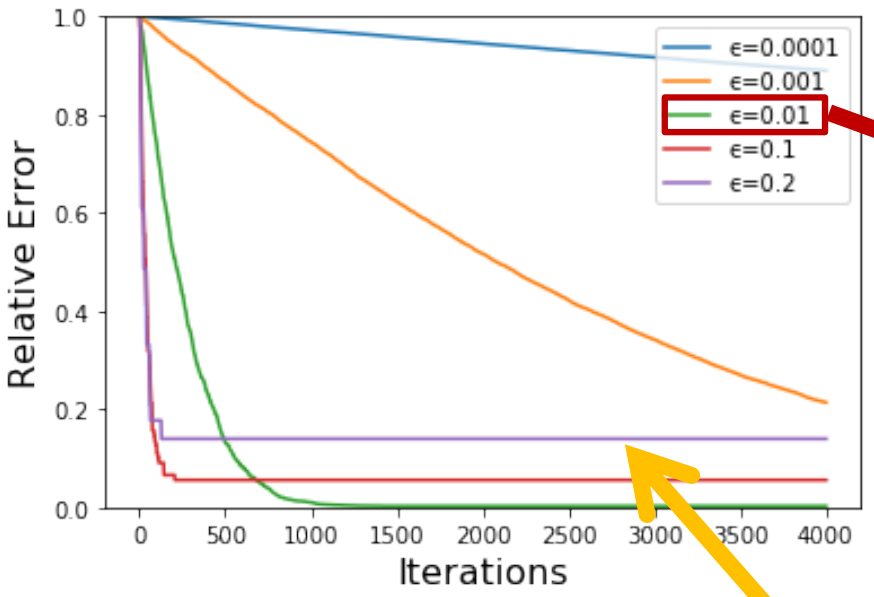
$$\Delta_{ij}^t(\epsilon) = \begin{cases} 1, & |x_i^t - x_j^t| \geq \epsilon, \\ 0, & \text{otherwise.} \end{cases}$$

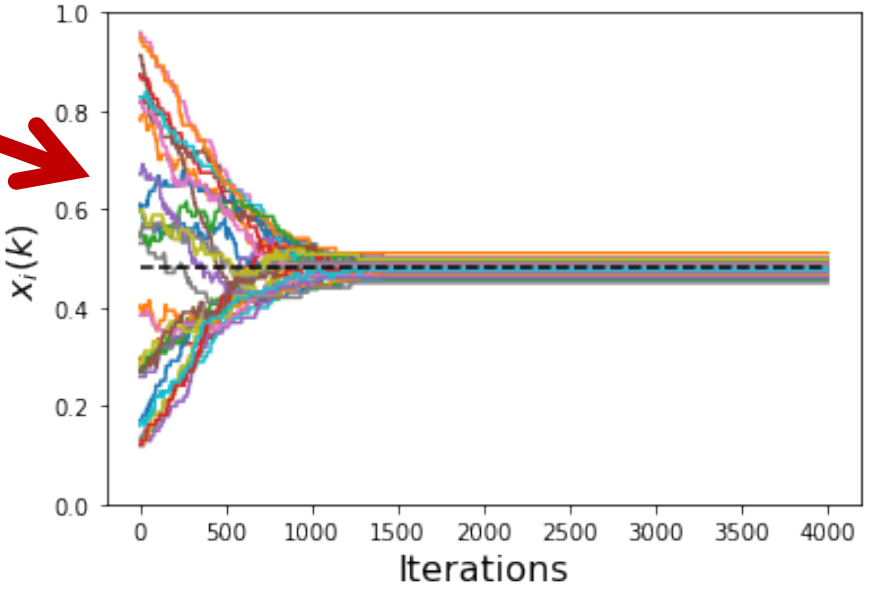
Observation $\quad \epsilon \cdot \Delta^t(\epsilon) \leq L^t$

$$L^t = \frac{1}{m}\sum_{\text{edges } (i,j)} |x_i^t - x_j^t|$$

Experiment

Evolution of node values





After some time, relative error is not decreasing anymore

Private Gossip via Controlled Noise Insertion

Private Gossip with Controlled Noise Insertion

"Do standard gossip except nodes lie about their private value"

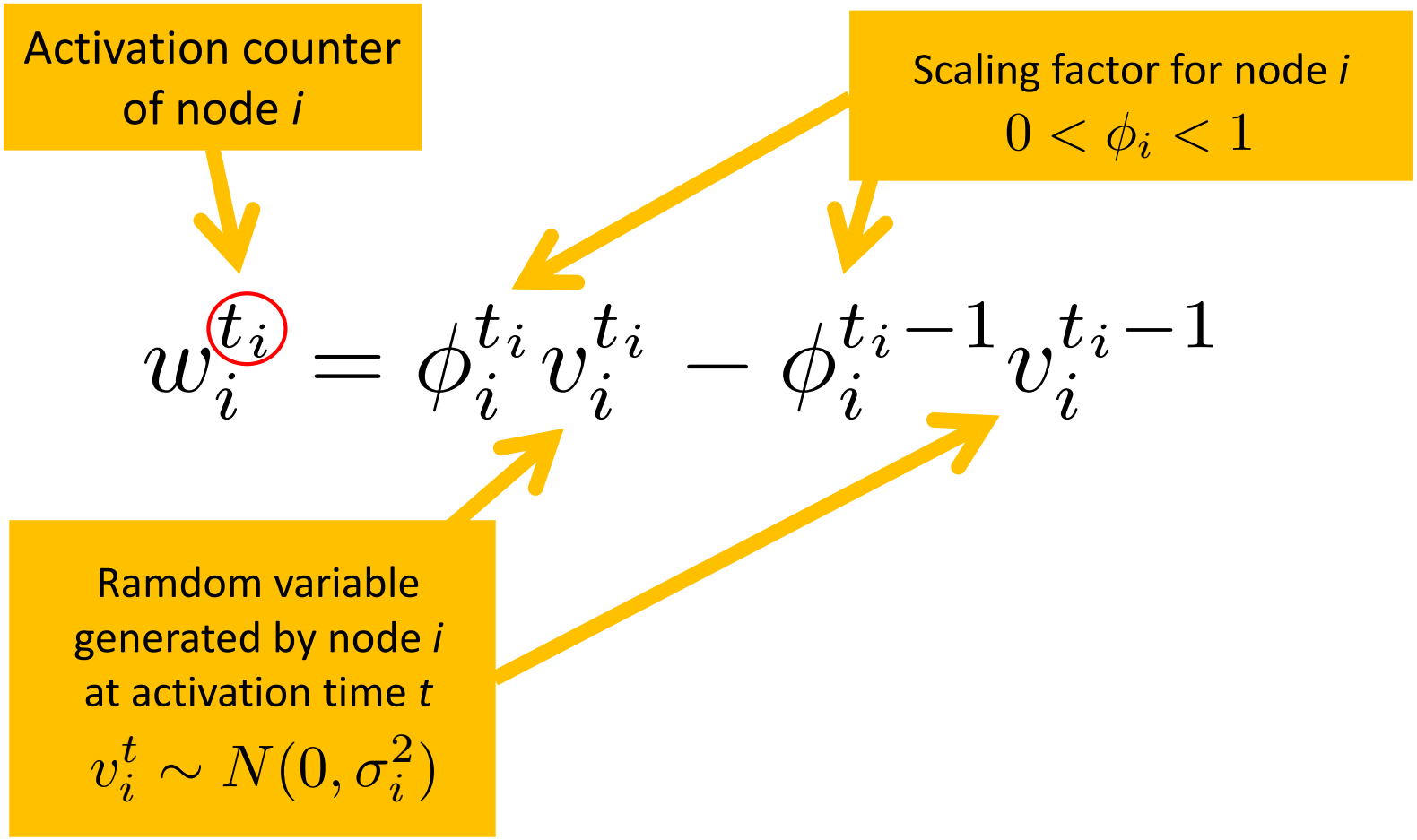
1. Initialize $x_i^0 = c_i$ for all $i = 1, 2, \ldots, n$
2. For $t \geq 0$ iterate:
   (a) Pick a random edge (i, j)
   (b) Set $x_i^{t+1} \leftarrow \frac{(x_i^t + w_i^{t_i}) + (x_j^t + w_j^{t_j})}{2}$
   (c) Set $x_j^{t+1} \leftarrow \frac{(x_i^t + w_i^{t_i}) + (x_j^t + w_j^{t_j})}{2}$
   (d) Set $x_u^{t+1} = x_u^t$ for all $u \notin \{i, j\}$

Standard gossip = red stuff is zero

Red stuff = Noise

Structure of the Noise

(Whenever node *i* is activated, it adds the structured noise to its value as a privacy measure)



Total Noise Eventually Vanishes

Theorem [HKLRG 2017]

Total noise in the system vanishes over time:

$$\lim_{t\to\infty} \mathbb{E}\left(\bar{c} - \frac{1}{n}\sum_{i=1}^{n} x_i^t\right)^2 = 0$$

Theory

$$\rho = 1 - \frac{\alpha(\mathcal{G})}{2m}$$

Theorem [HKLRG 2017]

$$\mathbb{E}[D(y^*) - D(y^k)] \leq \rho^k \left(D(y^*) - D(y^0)\right) + \frac{\sum \left(d_i \sigma_i^2\right)}{4m} \sum_{t=1}^{k} \rho^{k-t} \psi^t$$

degree of node *i*

Weighted sum of exponentials:

$$\psi^t = \frac{1}{\sum_{i=1}^{n} \left(d_i \sigma_i^2\right)} \sum_{i=1}^{n} d_i \sigma_i^2 \left(1 - \frac{d_i}{m}\left(1 - \phi_i^2\right)\right)^t$$

noise decrease rate

$\Rightarrow$ ψ^t depends only on biggest of $1 - \frac{d_i}{m}\left(1 - \phi_i^2\right)$ for large t

$\Rightarrow$ Increasing ϕ_i for other i does not influence the bound

Theory

degree of node *i*

$$1 - \frac{d_i}{m}\left(1 - \phi_i^2\right) \quad \text{is a constant}$$

$$\forall i : \gamma \leq d_i$$

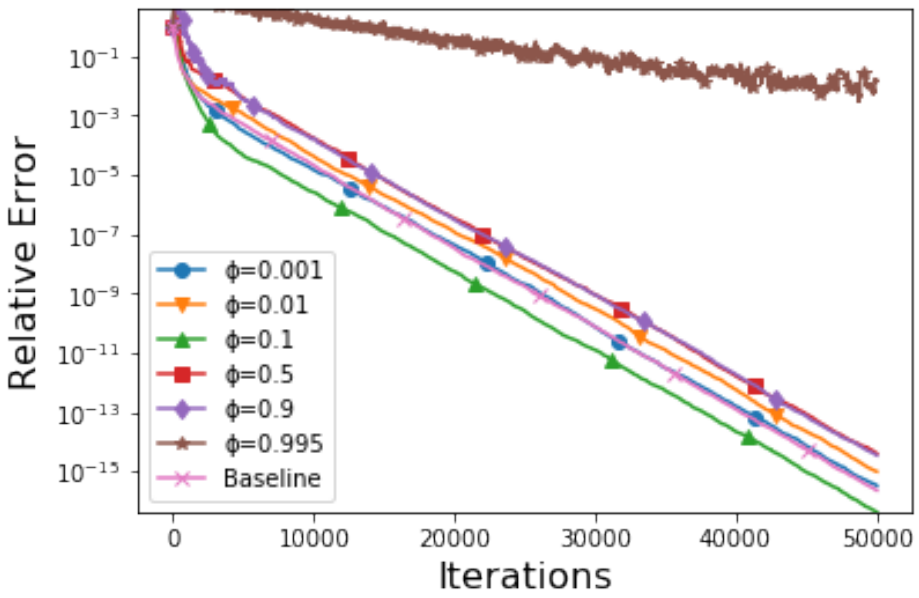
$$\phi_i = \sqrt{1 - \frac{\gamma}{d_i}}$$

Corollary

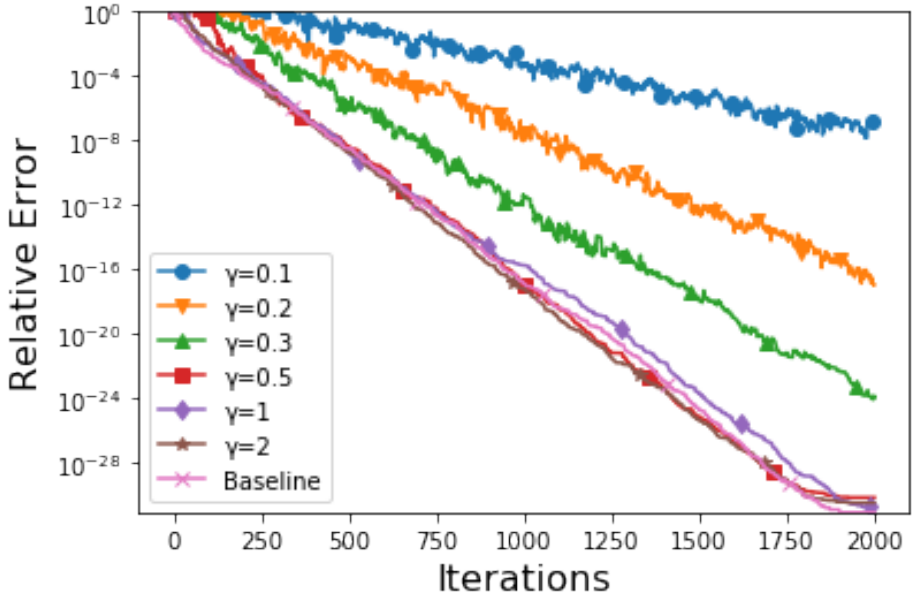
$$\mathbb{E}[D(y^*) - D(y^k)] \leq \left(1 - \min\left(\frac{\alpha(\mathcal{G})}{2m}, \frac{\gamma}{m}\right)\right)^k \left(D(y^*) - D(y^0) + \frac{\sum_{i=1}^n \left(d_i \sigma_i^2\right)}{4m} k\right)$$

Effect of the Noise Decrease Rate on Convergence

All nodes have the same noise decrease rate



Noise decrease rate driven by γ from the theory

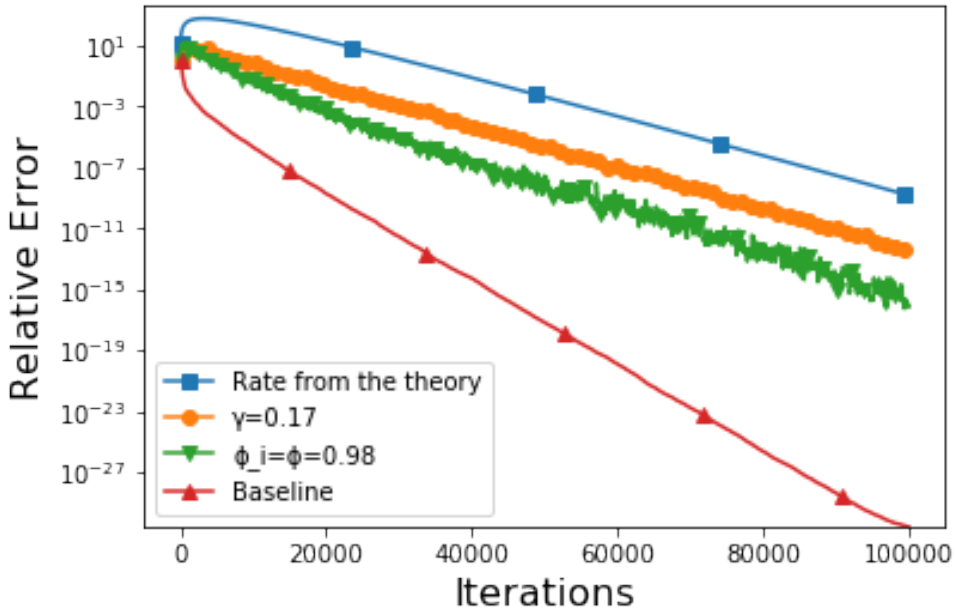


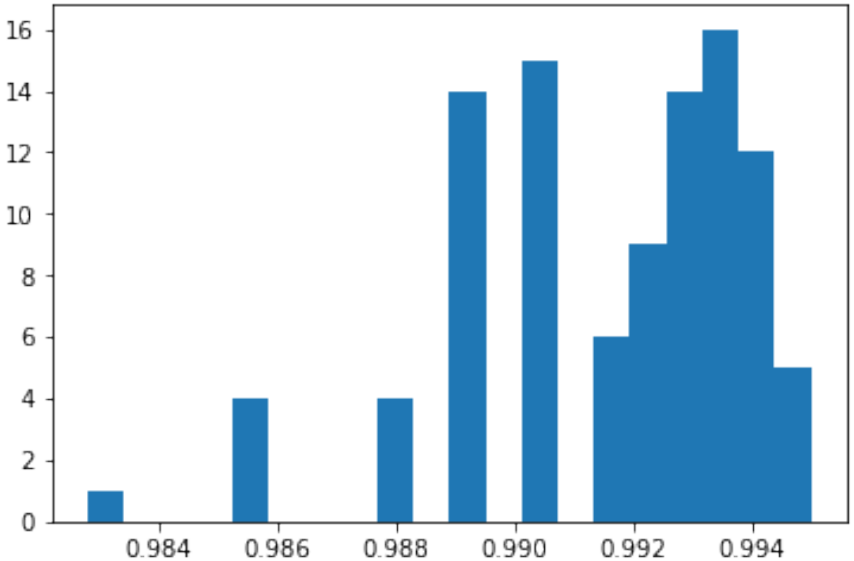
Comparing with Theory & Standard Gossip

Maximal noise decrease rate

Comparison with theory and Standard Gossip

Histogram of maximal values ϕ_i not violating convergence rate





Summary

Summary

- Introduced the average consensus problem and mentioned
  - Sensor networks
  - Social networks
  - Federated learning
- Reviewed the standard randomized gossip algorithm
- Introduced 3 new "privacy preserving" randomized gossip algorithms:
  - PRG with Binary Oracle
  - PRG with Gap Oracle
  - PRG with Controlled Noise Insertion
- Proved bounds on # iterations for various measures of success
- Did not prove any formal privacy guarantees!

Convergence Results

Algorithm	Success Measure	Rate
Standard Randomized Gossip	$\mathbb{E}\left[\frac{1}{2}\|x^k - x^*\|^2\right]$	$\left(1 - \frac{\alpha}{2m}\right)^k$
Private Randomized Gossip (Binary Oracle)	$\min_{t \le k} \mathbb{E}\left[\frac{1}{m} \sum_{\text{edges } (i,j)} \|x_i^t - x_j^t\|\right]$	$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$
Private Randomized Gossip (Gap Oracle)	$\mathbb{E}\left[\frac{1}{k}\sum_{t=0}^{k-1} \Delta^t(\epsilon)\right]$	$\mathcal{O}\left(\frac{1}{k\epsilon^2}\right)$
Private Randomized Gossip (Controlled Noise Insertion)	$\mathbb{E}\left[\frac{1}{2}\|x^k - x^*\|^2\right]$	$\left(1 - \frac{\min\{\alpha, 2\gamma\}}{2m}\right)^k$

THE END