

# Asynchronous Decentralized Online Learning

Published in NeurIPS 2021

Jiyan Jiang, Wenpeng Zhang, Jinjie Gu, Wenwu Zhu

Presenters: Haiguang Wang, Yufeng Chen, Yuxiang Hong

2025 年 11 月 7 日

# Outline

Outline

Introduction

AD-OCO Framework

AD-OGP Algorithm

Experiments

Conclusion

References





# Motivation and Background

- **Online learning:** Process large-scale streaming data efficiently
- **Existing decentralized algorithms:** Operate under synchronous setting
  - Leads to **straggler problem**: Fast learners wait for slow learners
- **This paper's purpose:** Design asynchronous counterpart of synchronous algorithms

## Key challenges:

- Each learner has five basic types of actions: prediction, weight update, message sending, message receiving, and model averaging.
- Asynchronization: Each learner executes its own actions independently ⇒ the orders of the actions of different learners are intermixed together.



# Contributions

**Novel Framework:** Asynchronous Decentralized Online Convex Optimization (AD-OCO)

- First formal framework that gives a complete characterization of the whole interaction dynamics of asynchronous decentralized online learning.
- Individual view: Specify the action sequence of each learner using its own indices.
- System view: Propose a novel event indexing system  $\Rightarrow$  mapping all learners into a single time axis.

**AD-OGP Algorithm:**

- A gossip-based communication scheme
  - Each learner is only allowed to communicate with its immediate neighbors.
  - Adopts "push-sum" strategy (a common way to realize asymmetric gossiping), which does not require any local synchronization among learners.
- A local update scheme build upon online gradient descent (OGD).
  - Design a novel **weighted projection** operation to avoid predictions outside the feasible domain.
- **Instantaneous model averaging:** Each learner performs model averaging as long as it's unoccupied and its receiving buffer is not empty.



## Theoretical Contributions:

- Conduct the **first** regret analysis of asynchronous decentralized online learning.
- Extend the graph augmentation technique to online setting to handle message delays.
- Disentangle the complex effects of predictions, weight update and model averaging via our proposed event indexing system.
- Derive a non-trivial regret bound of AD-OGP: it is the same order  $O(\sqrt{T})$  as that of its synchronous counterpart.
- Make an extra contribution to the convergence analysis of push-sum: reduce a factor of  $\sqrt{n}$  ( $n$  is the dimension of model parameters) in the convergence rate of push-sum compared to the previous results, and the final bound is now independent of  $n$ .



# Problem Formulation

## Network Structure:

- Undirected graph  $\mathcal{G} = (V, E)$  with vertex set  $V = \{1, \dots, m\}$
- Each learner  $i \in V$  maintains a local model  $w_i \in \mathcal{K}$  (convex compact decision set)
- Each learner  $i$  can only communicate with its immediate neighbors  
 $\mathcal{N}(i) = \{j \in V \mid (i, j) \in E\}$

**Individual View:** Specifies the action sequence of each individual learner.

- Learner  $i \in V$  is selected adversarially by the environment to make predictions at certain time point.
- Index the prediction rounds of learner  $i$  as  $\tau = 1, \dots, N_i$ , learner  $i$  predicts with its most recent model  $w_i^\tau$ .
- Environment reveals a convex loss function  $f_i^\tau : \mathcal{K} \mapsto \mathbb{R}$ , and learner suffers from the loss  $f_i^\tau(w_i^\tau)$ .
- After finishing the calculation, it uses the gradient to update its model  
 $w_i \leftarrow \mathcal{U}(w_i; \nabla f_i^\tau(w_i^\tau))$ , where  $\mathcal{U}$  is the local update scheme.



- Then it selects a subset of neighbors, and *sends* a copy of its updated model to each of them.
- Besides sending out its own model copies, learner  $i$  will also *receive* its neighbors' model copies at certain time points. These copies will be stored in receiving buffer  $\mathcal{B}_i$ .
- Whenever learner  $i$  is not occupied with gradient calculation or local update, and its buffer is not empty, it can *average* its model with the model copies stored in the buffer, i.e.,  $w_i \leftarrow \mathcal{A}(w_i; \mathcal{B}_i)$ , where  $\mathcal{A}$  is the model averaging scheme.



# System View and Event Indexing

**Global Event Sequence:** Due to synchronization, each learner executes its actions independently.  $\Rightarrow$  The order of actions of all learners are intermixed.

- We need to specify the complete orders of the actions of all learners by mapping them into a single time axis.
- We view either a **predictions** or a **local update** of any learner as an event, and use a virtual counter  $t$  to index these events.
- Let  $T = \sum_{i \in V} N_i$  be the total number of **predictions** of all learners, then the total number of **local updates** is also  $T$ .
- For each event  $t \in \{1, \dots, 2T\}$ , we use  $\delta_t \in \{0, 1\}$  to distinguish the event type:
  - $\delta_t = 0$  if it is a prediction event.
  - $\delta_t = 1$  if it is a local update event.
- We further use  $i_t$  to denote the learner that executes the event  $t$ , and  $f_t$  to denote the associated feedback.
- Then there exists some  $\tau \in \{1, \dots, N_{i_t}\}$  such that  $f_t = f_{i_t}^\tau$ .

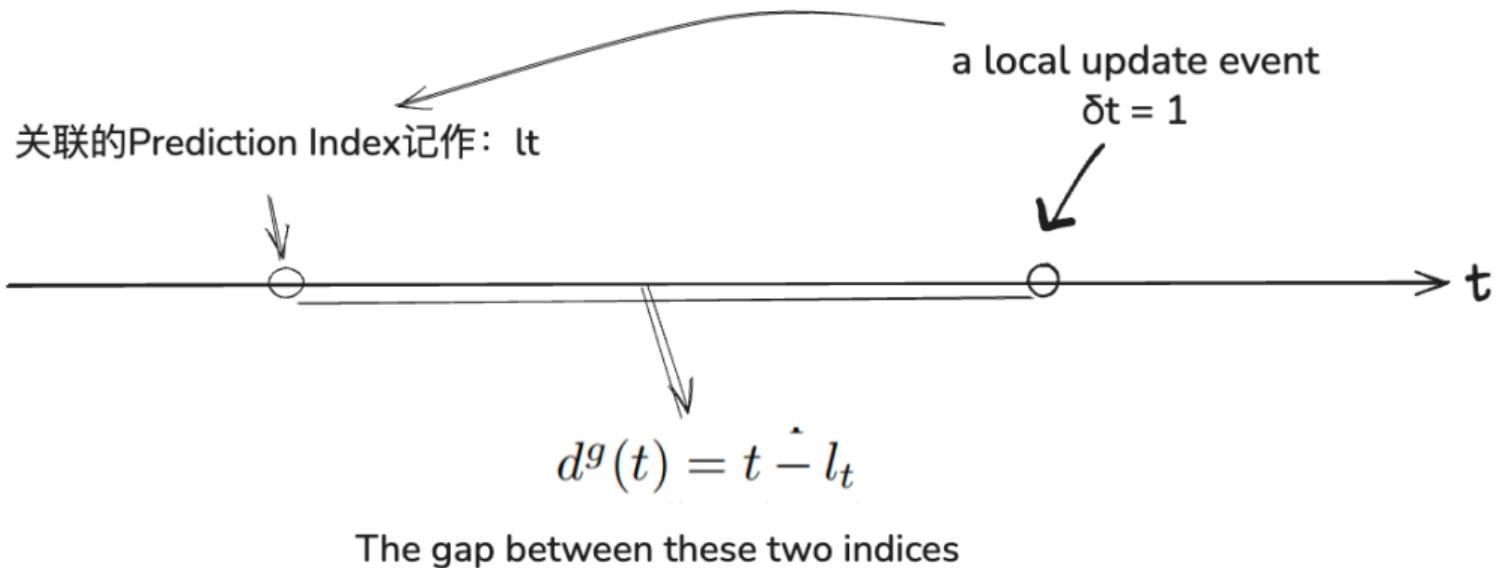


图: An example of Prediction and local update.



# Delays and Model Evolution

**Processing delay:** Since any prediction must occur before its corresponding local update, here we have  $t \geq 2$ ,  $l_t < t$  and  $d^g(t) > 0$ .

$$d^p(t) = \text{card}\{l_t < s < t \mid \delta_s = 1\}$$

i.e., the number of local updates between prediction and its associated local update.

**Message delay:** After any local update event  $t$ , the learner  $i_t$  that executes this event will send out its model copy to a subset  $S_t$  of its neighbors  $\mathcal{N}(i_t)$ . We focus on the copy sent to any neighbor  $j \in S_t$ . We assume such copy is used to average learner  $j$ 's model between events  $r_{tj}$  and  $(r_{tj} + 1)$  for some  $r_{tj} \geq t$ . For the message sent from learner  $i_t$  to learner  $j$  after event  $t$ , we introduce a quantity termed the message delay  $d_{i_t}^m(t) = r_{tj} - t$ .

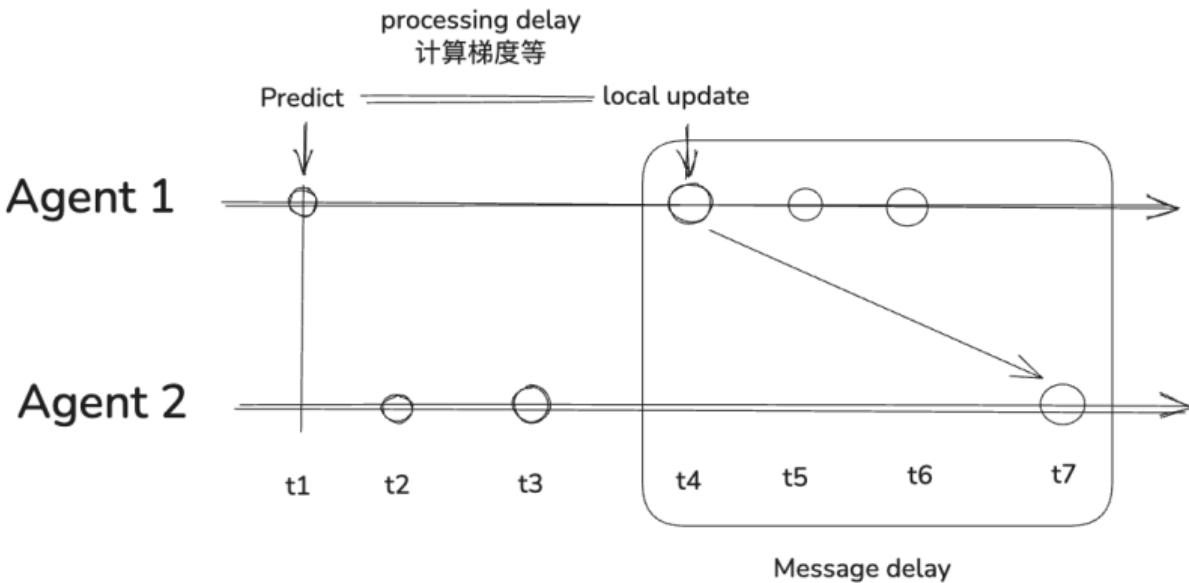


图: An example of Processing delay and Message delay.



# Model Evolution and Regret Definition

**Model Evolution:** We denote  $w_i(t)$  as the most recent model before event  $t$  occurs, and  $u_i(t)$  as the model immediately after event  $t$ .

- For an update event executed by learner  $i_t$ , its associated gradient is calculated based on learner  $i_t$ 's prediction using model  $w_{i_t}(l_t)$  at event  $l_t$ :

$$u_{i_t} = \mathcal{U}(w_{i_t}(t); \nabla f_{l_t}(w_{i_t}(l_t)))$$

The models of other learners do not change at  $i_t$ 's local update event, so  $u_j(t) = w_j(t)$  for  $j \neq i_t$ .

- Model averaging operation is not regarded as an event. It happens between  $t$  and  $t + 1$  if learner  $i$  is not occupied.
  - Model at the beginning:  $u_i(t)$
  - Model at the end:  $w_i(t + 1)$

So, the overall effect of model averaging on learner  $i$  between event  $t$  and  $t + 1$  can be expressed as:

$$w_i(t+1) = \begin{cases} u_i(t), & s = 0 \\ \mathcal{A}(\dots(\mathcal{A}(\mathcal{A}(u_i(t), \mathcal{B}_i^1(t)), \mathcal{B}_i^2(t))\dots), \mathcal{B}_i^s(t)), & s \geq 1 \end{cases}$$



**Regret Definition:** In AD-OCO, the regret w.r.t an arbitrary reference learner  $j \in V$  is defined as:

$$\text{Regret}_j = \sum_{t \in \mathcal{P}_T} f_t(\mathbf{w}_j(t)) - \sum_{t \in \mathcal{P}_T} f_t(\mathbf{w}^*)$$

where  $\mathcal{P}_T = \{t \in \{1, \dots, 2T\} \mid \delta_t = 0\}$  denotes the set of all prediction events, and  $w^* \in \operatorname{argmin}_{w \in \mathcal{K}} \sum_{t \in \mathcal{P}_T} f_t(w)$  refers to a **fixed model** chosen with hindsight.

**Goal:** Learners aim to make a prediction sequence  $\{\mathbf{w}_i(t)\}_{i \in V, t \in \mathcal{P}_T}$ , such that the regret of any reference learner  $j$  grows sublinearly with time, i.e.,

$$\lim_{T \rightarrow \infty} \text{Regret}_j / T = 0$$

---

**Protocol 1** Asynchronous Decentralized Online Convex Optimization (**AD-OCO**)
 

---

- 1: **Input:** Time horizon  $T$ , convex set  $\mathcal{K}$ , local update scheme  $\mathcal{U}$ , and model averaging scheme  $\mathcal{A}$ .
- 2: **Initialize:** Local models  $\mathbf{w}_j(1) \in \mathcal{K}, \forall j \in V$ .
- 3: **for**  $t = 1, \dots, 2T$  **do**
- 4:   **if**  $\delta_t = 0$  **then** *//for a prediction event*
- 5:     A learner  $i_t \in V$  is selected by the environment adversarially to predict with  $\mathbf{w}_{i_t}(t)$ .
- 6:     Learner  $i_t$  receives a convex function  $f_t$  from environment, and suffers the loss  $f_t(\mathbf{w}_{i_t}(t))$ .
- 7:     Learner  $i_t$  starts to compute the gradient  $\nabla f_t(\mathbf{w}_{i_t}(t))$ .
- 8:   **else** *//for a local update event*
- 9:     Learner  $i_t$  performs a local update:  
$$\mathbf{u}_{i_t}(t) = \mathcal{U}(\mathbf{w}_{i_t}(t); \nabla f_{l_t}(\mathbf{w}_{i_t}(l_t))).$$
- 10:    Learner  $i_t$  selects a subset  $\mathcal{S}_t$  of its neighbors  $\mathcal{N}(i_t)$  and sends  $\mathbf{u}_{i_t}(t)$  to each  $j \in \mathcal{S}_t$ .
- 11:   **end if**
- 12:   **for**  $i \in V$  **do**
- 13:     Learner  $i$  performs model averaging using model copies  $\mathcal{B}_i^1(t), \dots, \mathcal{B}_i^s(t)$  if it is *unoccupied*:  
$$\mathbf{w}_i(t+1) = \begin{cases} \mathbf{u}_i(t), & s = 0; \\ \mathcal{A}(\cdots(\mathcal{A}(\mathcal{A}(\mathbf{u}_i(t), \mathcal{B}_i^1(t)), \mathcal{B}_i^2(t)) \cdots), \mathcal{B}_i^s(t)), & s \geq 1. \end{cases}$$
- 14:   **end for**
- 15: **end for**

---

图: The AD-OGP Algorithm Framework (Algorithm 1).



# Algorithm Overview

**Initial State:** Each node  $i$  maintains a pair of variables  $(x_i, y_i)$ , and predicts with

$$w_{i_t}(t) = \frac{x_{i_t}(t)}{y_{i_t}(t)}.$$

- $x_i$  is a **weighted model**,  $y_i$  is a **weight scalar** (always  $> 0$ ). Both are used for push-sum.
- Initialization:  $x_i = w_0 \in \mathcal{K}$  ( $\mathcal{K}$  is a convex compact decision set),  $y_i = 1$ .

## AD-OGP Algorithm Steps:

---

### Algorithm 1 Asynchronous Decentralized Online Gradient-Push (AD-OGP)

---

- 1: **Input:** Learning rate  $\eta$ , initial model  $w_0 \in \mathcal{K}$
- 2: **Initialize:** For each learner  $i \in V$ ,  $x_i \leftarrow w_0$ ,  $y_i \leftarrow 1$ . Empty receiving buffer  $\mathcal{B}_i$ .
- 3: **Loop for each event**  $t \in \{1, \dots, 2T\}$ :
  - 4:   **If**  $\delta_t = 0$  (prediction event) for learner  $i_t$ :
    - 5:      $w_{i_t}(t) \leftarrow x_{i_t}(t)/y_{i_t}(t)$
    - 6:     Make prediction with  $w_{i_t}(t)$ , receive loss function  $f_t$ .
  - 7:   **If**  $\delta_t = 1$  (local update event) for learner  $i_t$ :
    - 8:     Let  $l_t$  be the prediction event corresponding to  $f_t$ .
    - 9:     Compute gradient  $q_t = \nabla f_{l_t}(w_{i_t}(l_t))$ .



# Push-Sum Communication Strategy

**Challenge:** Decentralized model averaging typically involves linear averaging with neighbors. In an asynchronous and asymmetric setting, traditional doubly-stochastic averaging is not feasible.

- *Why?*

- Doubly-stochastic matrices require symmetric interactions ( $p_{ij} = p_{ji}$ ) and global synchronization ( $v(t+1) = Pv(t)$ ).
- In asynchronous asymmetric gossiping, learner  $i$  sends its model to  $j$  without waiting for acknowledgment or  $j$ 's model. This makes symmetric weights impossible to negotiate.
- Lack of global rounds means  $\sum_j p_{ij} = 1$  (row stochastic) and  $\sum_i p_{ij} = 1$  (column stochastic) cannot be maintained simultaneously in a global snapshot.
- Column stochasticity ( $P_{ii} = \gamma_i, \forall j \in \mathcal{N}(i), P_{ji} = \gamma_i$ ) can be enforced locally for outgoing messages, but row stochasticity cannot.

- *How Push-sum addresses this?*

- Each learner maintains a tuple  $(x_i, y_i)$ .  $x_i$  carries model information,  $y_i$  is an auxiliary scalar.
- After local update, learner  $i$  calculates  $\gamma_i = 1/(|\mathcal{N}(i)| + 1)$ , updates  $x'_i = \gamma_i x_i$ ,  $y'_i = \gamma_i y_i$ , and sends  $(x'_i, y'_i)$  to all neighbors.
- Neighbors accumulate these received quantities into their  $(x, y)$  pair (Instantaneous Model Averaging).



# Regret Bound Analysis

## Assumptions:

- **Assumption 1:** The message delays are bounded by some integer  $D^{msg} \geq 0$ .
- **Assumption 2:** Each learner performs local updates at least once every  $\Gamma_d$  steps for some integer  $\Gamma_d > 0$ .

## Definitions:

- $\mathcal{Q}_{s,t} = \mathcal{Q}_T \cap (s, t)$  denotes local updates between events  $s$  and  $t$ .
- For any  $t \in \mathcal{Q}_T$ ,  $d^p(t) = |\mathcal{Q}_{l_t, t}|$ .
- $g_t = \nabla f_{l_t}(x_{i_t}(l_t)/y_{i_t}(l_t))$ ,  $\hat{g}_t = \nabla f_{l_t}(x_j(l_t)/y_j(l_t))$  for local update gradients.
- Total processing delay:  $D^{proc} = \sum_{t \in \mathcal{Q}_T} d^p(t)$ .

## Key Techniques for Analysis:

- **Graph Augmentation Technique:** For each real node  $i$ , create  $D^{msg}$  virtual nodes. Message passing from  $i$  to  $j$  is viewed as a delay-free path on this augmented graph. This helps to handle message delays.
- **Mixing Analysis:** After a sufficient number of time steps (related to network diameter  $\mathcal{D}$  and max delay  $D^{msg}$ , denoted as  $B$ ), the multi-step transition matrix  $\mathcal{Q}_{s,s+B}$  ensures that information from any node propagates to any other real node.



# Regret Bound Derivation

## General Regret Upper Bound:

$$\text{Regret}_j \leq \frac{m}{2\eta} \|\mathbf{w}_0 - \mathbf{w}^*\|^2 + \frac{2\eta}{m} \sum_{t \in \mathcal{Q}_T} \sum_{s \in \mathcal{Q}_{l_t}, t+1} g_t g_s + \eta \sum_{t \in \mathcal{Q}_T} \frac{g_t^2}{y_{i_t}(t)} \\ + 2\eta \sum_{t \in \mathcal{Q}_T} \sum_{s \in \mathcal{Q}_{1, l_t}} \lambda^{\lfloor \frac{l_t-s}{2B} \rfloor} (g_t + 2\hat{g}_t) \frac{g_s}{y_{i_s}(t)} + 2\eta \sum_{t \in \mathcal{Q}_T} \sum_{s \in \mathcal{Q}_{1, t+1}} \lambda^{\lfloor \frac{t-s}{2B} \rfloor} g_t \frac{g_s}{y_{i_s}(t)},$$

where  $\lambda = 1 - m\alpha^4$ ,  $\alpha = (1/m)^B$ ,  $B = (\mathcal{D} + 1)(D^{msg} + \Gamma_d)$ .

**Final Regret Bound** (assuming  $g_t \leq G, \hat{g}_t \leq G, \forall t \in \mathcal{Q}_T$  for  $G < \infty$ ):

$$\text{Regret}_j \leq \frac{m}{2\eta} \|\mathbf{w}_0 - \mathbf{w}^*\|^2 + \frac{2\eta}{m} G^2 (T + D^{proc}) + \frac{2\eta}{m} \left( \frac{8 + \alpha^4}{\alpha^5} \right) B G^2 T$$

By setting  $\eta = (mF/2G)(D^{proc} + T + CBT)^{-1/2}$ , where  $C = (8 + \alpha^4)/\alpha^5$ , the final regret is  $O((CBT + T + D^{proc})^{1/2})$ .



# Experimental Setup

## Two Large-Scale Real-World Datasets:

- **Higgs dataset**: A benchmark dataset in high-energy physics for binary classification, consisting of 11 million instances with 28 features. Uses logical loss.
- **Poker-hand dataset**: A commonly used dataset in automatic rule generation for 10-class classification, with 1 million instances and 25 features. Uses multivariate logistic loss.

*These datasets, with different tasks (binary vs. multi-class) and domains, test the algorithm's scalability and generalization.*

## Comparison and Verification Goals:

- Compare AD-OGP with its synchronous counterpart (D-OGP) to show the advantage of synchronization.
- Verify the effectiveness of asymmetric gossiping and instantaneous model averaging.
- Verify our theoretical regret bound.

## Network Topologies for Comparison:

- Complete graph (high connectivity)
- Watts-Strogatz graph (random graph with medium connectivity)
- Ring graph (low connectivity)

# The Benefit of Asynchronization

Missing image: 21f7f0f0-b57a-4a08-96ce-504e3e0842ba:image.png

# Effectiveness of Asymmetric Gossiping and Instantaneous Model Averaging

## 4.3.1 The Effectiveness of Asymmetric Gossiping

Missing image: 944ac58f-0294-4c3a-88c4-fc0b5f2be828:image.png

# Verification of the Theoretical Bound

Missing image: f2167018-059a-4570-aeb0-669cfb26a2be:image.png



# Conclusions and Future Work

## Key Contributions:

- In this paper, we present the first systematic study of asynchronous decentralized online learning.
- We begin by formulating the framework of Asynchronous Decentralized Online Convex Optimization (AD-OCO), which gives a complete characterization of the whole interaction dynamics.
- Then we devise the Asynchronous Decentralized Online Gradient-Push (AD-OGP) algorithm, which is fully asynchronous and includes two novel innovations: weighted projection and instantaneous model averaging.
- Theoretically, we provide the first regret analysis paradigm for asynchronous decentralized online learning, deriving a non-trivial regret bound of  $O(\sqrt{T})$ .
- Finally, we conduct extensive experiments to demonstrate the benefit of synchronization, verify the effectiveness of the two innovations, and corroborate the theoretical bound.
- Our work paves the way for future research investigating asynchronous decentralized online learning.

## Limitations and Future Work:



# References

-  Anonymous Author 1, Anonymous Author 2, Anonymous Author 3. (2023).  
**Asynchronous Decentralized Online Convex Optimization.**  
In Submission to Major Machine Learning Conference.
-  Tsianos, K. I., Lawlor, S., & Rabbat, M. G. (2012).  
**Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning.**  
In 2012 50th Annual Allerton Conference on Communication, Control, and Computing.
-  Kempe, D., Dobra, A., & Gehrke, J. (2003).  
**Gossip-based computation of aggregate information.**  
In 44th Annual IEEE Symposium on Foundations of Computer Science.