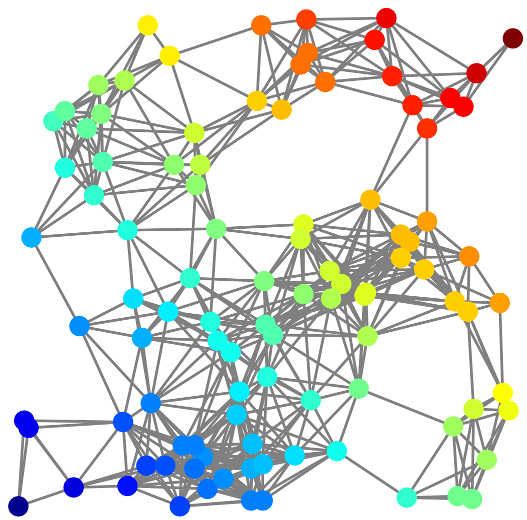
Consensus-Based Distributed Optimization

Communication-Computation Tradeoffs



Konstantinos Tsianos, Sean Lawlor, & Michael Rabbat

McGill

Separable Convex Optimization

Consider problems of the form

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n}\sum_{i=1}^{n} f_i(x) \\ \text{subject to} \quad & x \in \mathcal{X} \end{aligned}$$

where $f_i(x)$ are convex,

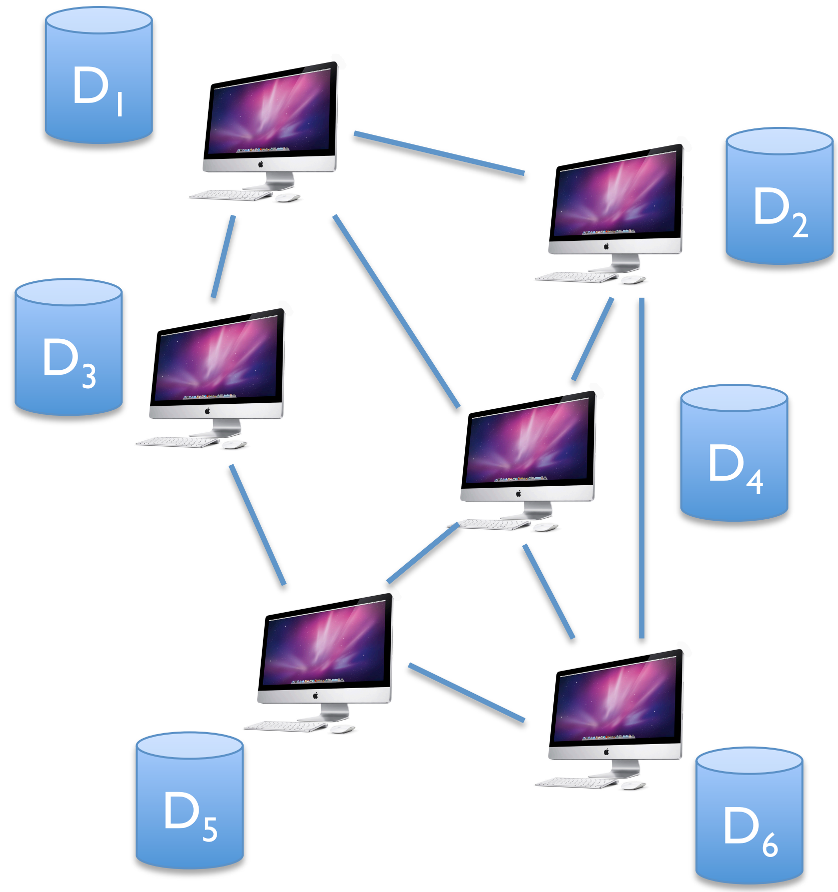
and $\mathcal{X} \subseteq \mathbb{R}^d$ is convex

Solve in a network where $f_i(x)$ only available at node i

Distributed Model Fitting

Fit a model to data at all nodes

$$\text{minimize} \sum_{i=1}^{n} \sum_{y \in \mathcal{D}_i} \ell(x, y)$$



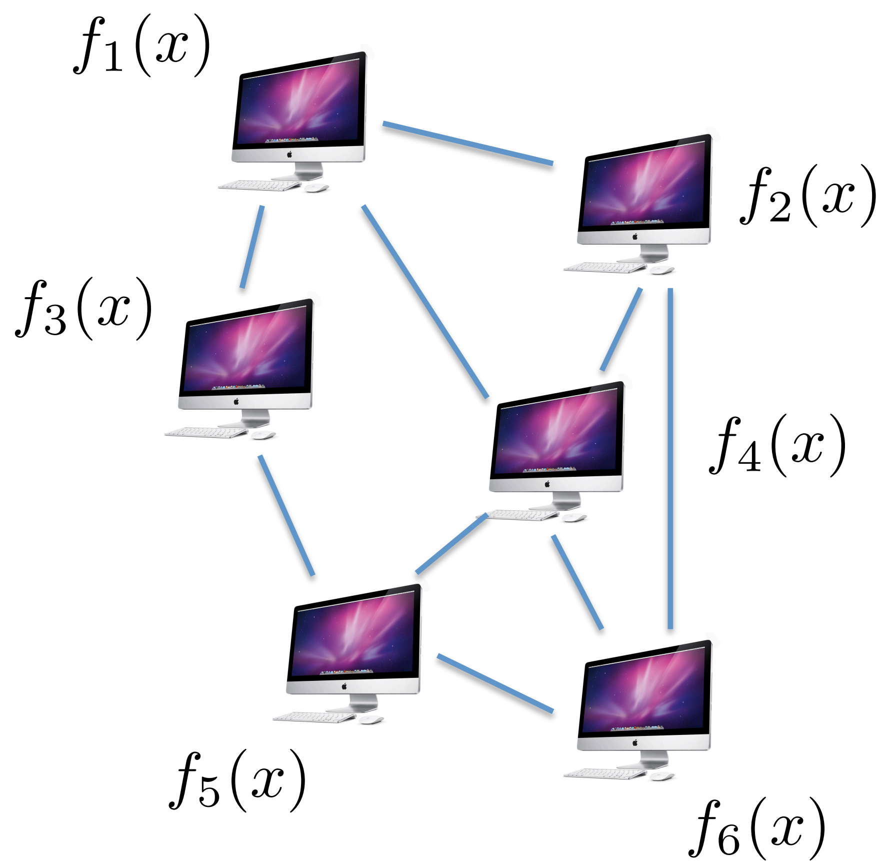
Distributed Model Fitting

Fit a model to data at all nodes

$$\text{minimize} \sum_{i=1}^{n} \underbrace{\sum_{y \in \mathcal{D}_i} \ell(x, y)}_{f_i(x)}$$

Communicate over logical overlay network

$$G = (\{1, \ldots, n\}, E)$$



Distributed Primal Averaging

Operation at node i, first initialize $x_i(0) \in \mathbb{R}^d$

repeat:

communicate: send $x_i(t)$ to neighbors, receive $x_j(t)$

compute: $g_i(t) \in \partial f_i(x_i(t))$

$$x_i(t+1) = \Pi_{\mathcal{X}} \left[\sum_{j=1}^{n} P_{i,j} x_j(t) - \alpha_t g_i(t) \right]$$

until satisfying convergence criterion

Assume P doubly stochastic $P_{i,j} > 0 \Leftrightarrow (i,j) \in E$

Nedic and Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE T Auto Control,* 2009
Ram, Nedic, Veeravalli, "Distributed stochastic subgradient projection algorithms," *J Opt Theory & Apps*, 2010

Distributed Dual Averaging (DDA)

Operation at node i, first initialize $z_i(0), x_i(0) \in \mathbb{R}^d$

repeat:

communicate: send $z_i(t)$ to neighbors, receive $z_j(t)$

compute: $g_i(t) \in \partial f_i(x_i(t))$

$$z_i(t+1) = \sum_{j=1}^{n} P_{i,j} z_j(t) - g_i(t)$$

$$x_i(t+1) = \arg\min_{x \in \mathcal{X}} \left\{ \langle z, x \rangle + \frac{1}{a(t)} \|x\|_2^2 \right\}$$

until satisfying convergence criterion

Assume P doubly stochastic $P_{i,j} > 0 \Leftrightarrow (i,j) \in E$

Duchi, Agarwal, Wainwright, "Dual averaging for distributed optimization," *IEEE Trans Auto Control*, 2011

Convergence of DDA

DDA updates:

$$z_i(t+1) = \sum_{j=1}^{n} P_{i,j} z_j(t) - g_i(t)$$

$$x_i(t+1) = \arg\min_{x \in \mathcal{X}} \left\{ \langle z, x \rangle + \frac{1}{a(t)} \|x\|_2^2 \right\}$$

Theorem (Duchi, Agarwal, and Wainwright '11): For the running average,

$$\widehat{x}_i(T) = \frac{1}{T} \sum_{t=1}^{T} x_i(T)$$

we have

$$F(\widehat{x}_i(T)) - F^* \leq C \frac{\log(\sqrt{n}T)}{(1-\lambda_2)\sqrt{T}}$$

Communication-Computation Tradeoffs

Tsianos, Lawlor, and Rabbat, NIPS 2012

A Closer Look at DDA

Error after T iterations

$$\epsilon(T) = F(\widehat{x}_i(T)) - F^* \leq C \frac{\log(\sqrt{n}T)}{(1-\lambda_2)\sqrt{T}}$$

- Bound increases with network size
- Assume fixed data set $y_1, y_2, \ldots, y_m$

$$F(x) = \frac{1}{m}\sum_{j=1}^{m} l(x, y_j) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\left(\frac{n}{m}\sum_{j=1}^{m/n} l(x, y_{j,i})\right)}_{f_i(x)}$$

- (Sub)Gradient computation is *n* times faster

$$\nabla_x f_i(x) = \frac{n}{m}\sum_{j=1}^{m/n} \nabla_x l(x, y_{j,i})$$

Time Model

- **Computation:** Normalize time so that

$$1 \text{ time unit} = \text{time to compute } \sum_{j=1}^{m} l(x, y_j)$$

  – Then takes $1/n$ time for n nodes

- **Communication:** Define problem-specific constant

$$r = \text{ time to transmit } z_i(t) \text{ to one neighbor}$$

  – Assume graph is *k*-regular

$$\text{total time for one iteration} = \frac{1}{n} + kr \text{ time units}$$

Communication-Computation Tradeoff

- DDA error bound $\epsilon(T) = C\frac{\log(\sqrt{n}T)}{(1-\lambda_2)\sqrt{T}}$
- Assume a favorable topology (*G* = K_n or *k*-regular expander)

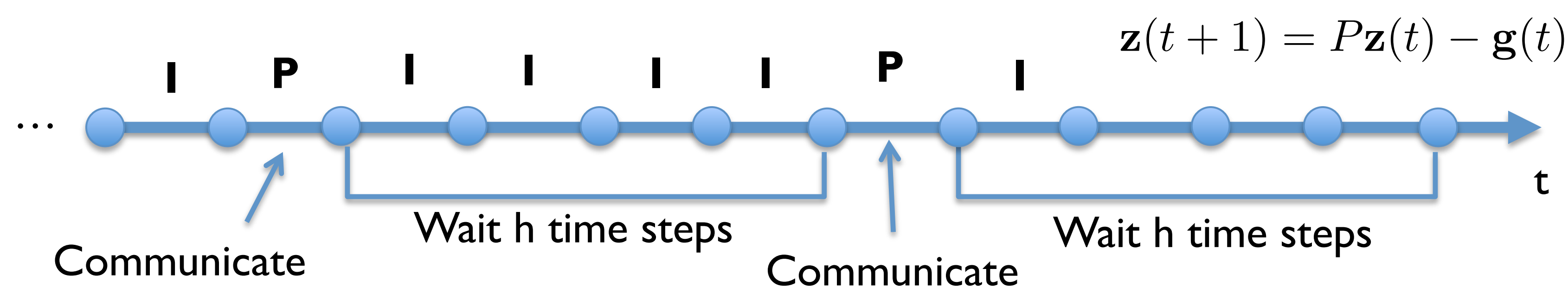
$$1 - \lambda_2 = \Theta(1) \quad \text{as } n \to \infty$$

- Time to reach ϵ accuracy is

$$\tau(\epsilon) \approx \frac{C^2}{\epsilon^2}(\frac{1}{n} + kr) \text{ time units}$$

  – If communication is free (*r* = 0): perfect linear speedup
  – If *G* = *Kn* : minimal time when $n = 1/\sqrt{r}$
  – If *G* is *k*-regular expander, get diminishing returns with increasing *n*

Sparse Communication



- If each node transmits once every h iterations we prove that

$$\epsilon = C_h \frac{\log(\sqrt{n}T)}{\sqrt{T}}, \quad C_h = \sqrt{c_1 + c_2 h}$$

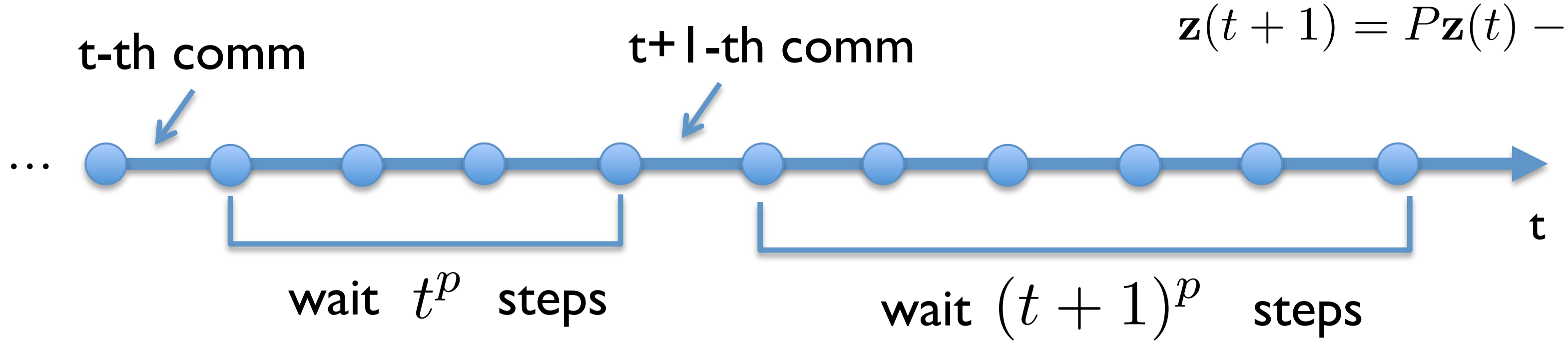
- Of the T iterations, $H_T = \frac{T}{h}$ involve communication, so

$$\tau(\epsilon) = \frac{T}{n} + \frac{T}{h} kr = \frac{C_h^2}{\epsilon^2}\left(\frac{1}{n} + \frac{kr}{h}\right) \quad \text{time units}$$

- There is an optimal $h_{opt} = c_3\sqrt{nkr}$
- Complete Graphs: $\tau(\epsilon) = O(n)$
- Expander Graphs: $\tau(\epsilon) = \frac{c_5}{\sqrt{n}} + c_6$

Increasingly Sparse Communication

$$\mathbf{z}(t+1) = P\mathbf{z}(t) - \mathbf{g}(t)$$



- To reach ϵ accuracy will take $\tau(\epsilon) = O\left(T\left(\frac{1}{n} + \frac{kr}{T^{\frac{1}{p+1}}}\right)\right)$ where $T = \left(\frac{C_p}{\epsilon}\right)^{\frac{2}{1-2p}}$
- For constant *k*, arbitrarily close to linear speedup $O\left(\frac{T}{n}\right)$
- The rate is slower in number or iterations than when communicating every iteration:

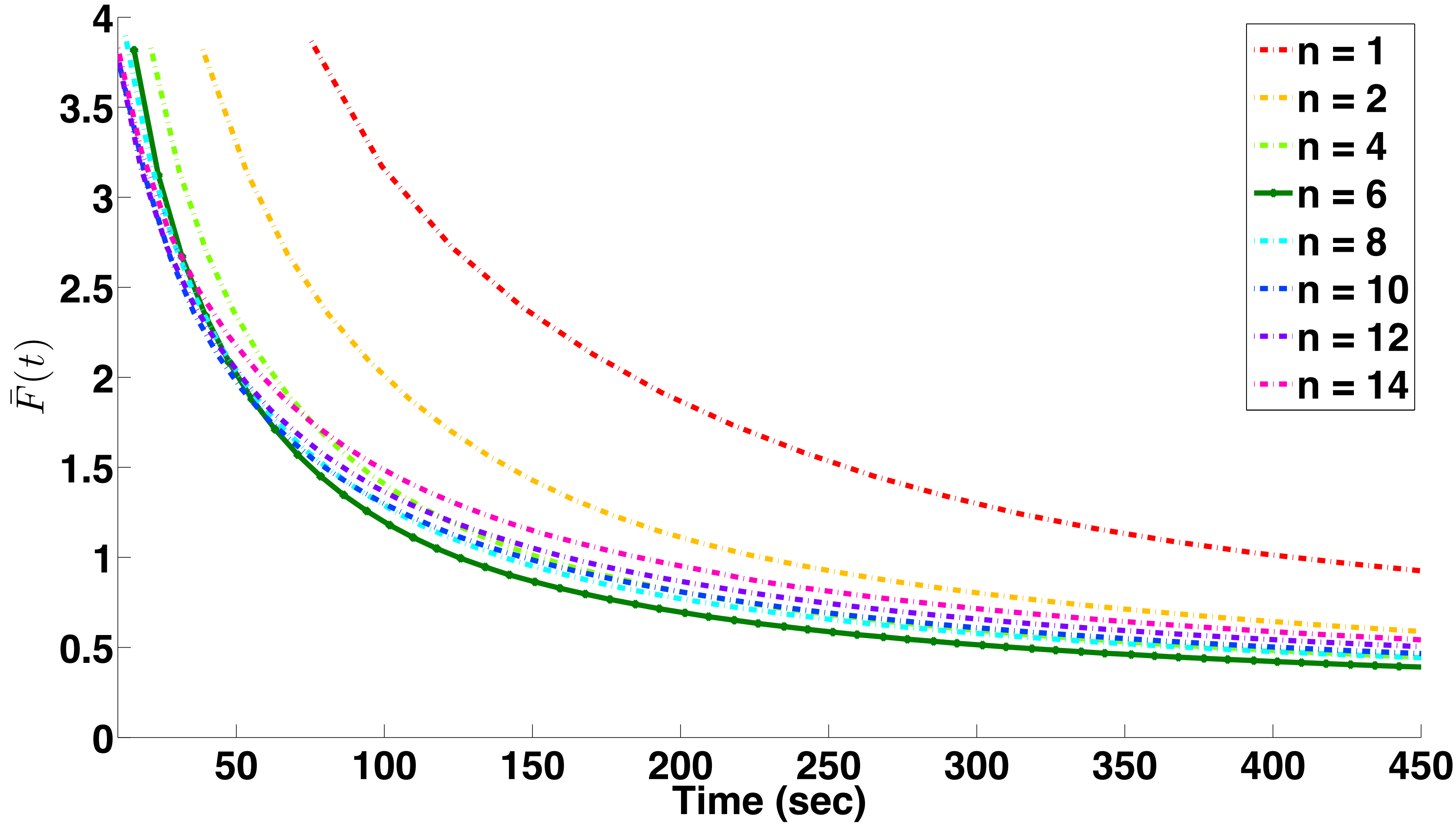
$$\frac{1}{\epsilon^2} \quad \text{VS} \quad \frac{1}{\epsilon^{\frac{2}{1-2p}}}$$

but the algorithm scales better with *n*

Experimental Evaluation

- Cluster with 14 nodes, complete graph,
- Network transmits 11 mb/sec
- Learn a distance metric $d_A(u,v) = \sqrt{(u-v)^T A(u-v)}$
  - 1 cpu needs 29 seconds to compute $\nabla F(w)$
  - Sending/receiving 1 gradient takes 0.85 seconds
    - Gradient dimension: 614657
    - Gradient size: 4.7 MB
  - Communication/Computation trade-off: $r = \frac{0.85}{29} = 0.0293$
  - Complete graph optimal size is $n = \frac{1}{\sqrt{r}} = 5.8$

Metric Learning Problem



Network of 6 cpus is the fastest. Theory predicts 5.8.

Non-smooth Minimization

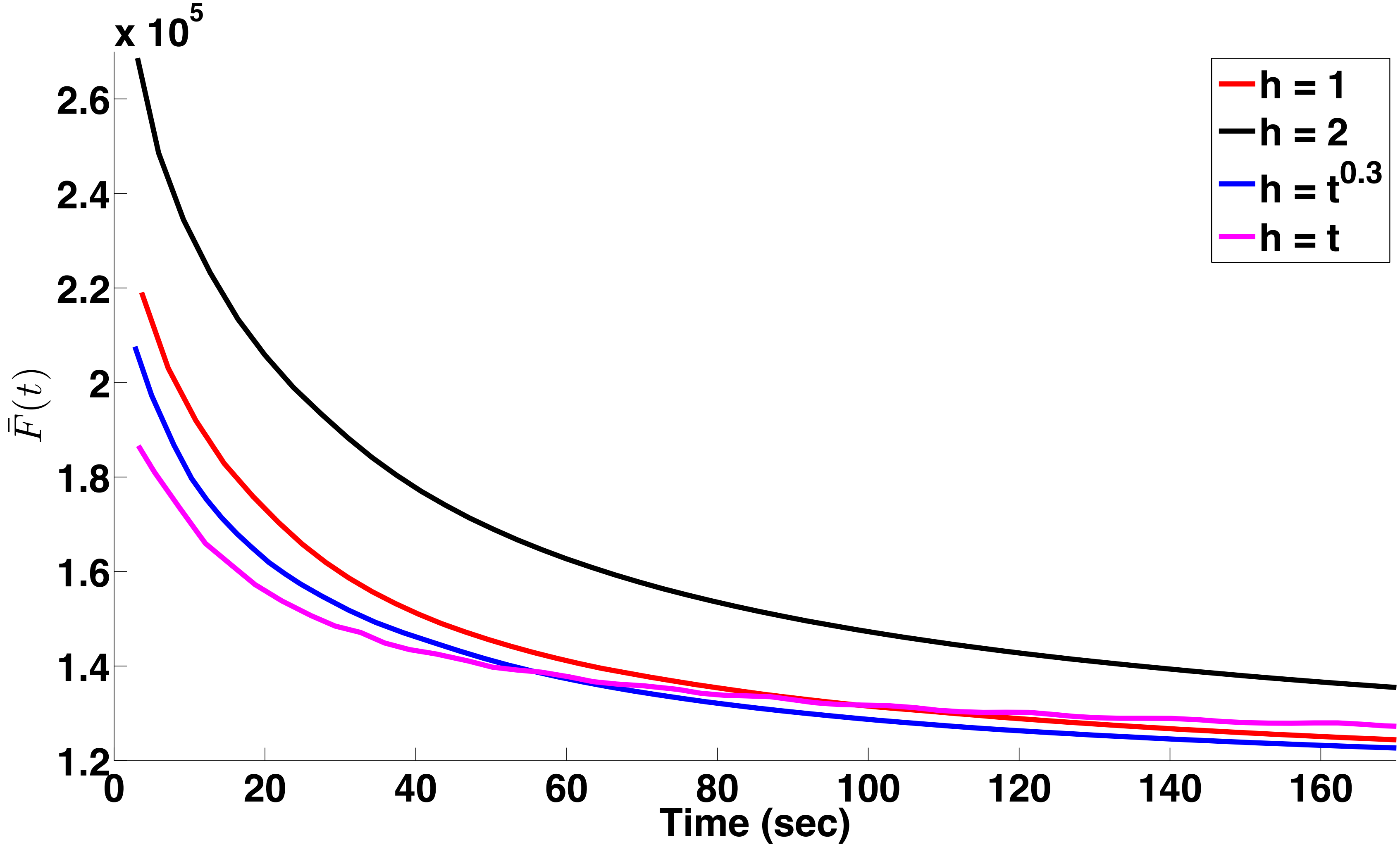
$$F(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w), \quad w \in R^{10,000}, M = 15,000$$

$$f_i(w) = \sum_{j=1}^{M} \max\left(l^1(w, x_{j|i}), l^2(w, x_{j|i})\right),$$

$$l^\xi(w, x_{j|i}) = (w - x_{j|i}^\xi)^T (w - x_{j|i}^\xi), \; \xi \in \{1, 2\}$$

- Complete graph of 10 nodes
  - $r = 0.00089$
  - $h_{opt} = 1$
  - For h=2 each node communicates $H_T = 55$ times
  - For p=0.3 each node communicates $H_T = 53$ times

Non-smooth Minimization



Practical Considerations

Tsianos, Lawlor, and Rabbat, Allerton 2012

Distributed Dual Averaging

Operation at node i, first initialize $z_i(0), x_i(0) \in \mathbb{R}^d$

repeat:

communicate: send $z_i(t)$ to neighbors, receive $z_j(t)$

compute: $g_i(t) \in \partial f_i(x_i(t))$

$$z_i(t+1) = \sum_{j=1}^{n} P_{i,j} z_j(t) - g_i(t)$$

$$x_i(t+1) = \arg\min_{x \in \mathcal{X}} \left\{ \langle z, x \rangle + \frac{1}{a(t)} \|x\|_2^2 \right\}$$

until satisfying convergence criterion

Assume P doubly stochastic $P_{i,j} > 0 \Leftrightarrow (i,j) \in E$

Duchi, Agarwal, Wainwright, "Dual averaging for distributed optimization," *IEEE T Auto Control*, 2011

Consensus-Based Distributed Optimization

General operation:

repeat:

communicate

compute

until (convergence)

$$z_i(t+1) = \sum_{j=1}^{n} P_{i,j} z_j(t) - g_i(t)$$

Synchronous or Asynchronous ?

Push-Pull or Push (or Pull) ?

Doubly stochastic *P* ?

Tsitsiklis, Bertsekas, Athans, "Distributed asynchronous gradient optimization algs" *IEEE T Auto Control,* 1986
Nedic and Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE T Auto Control,* 2009
Ram, Nedic, Veeravalli, "Distributed stochastic subgradient projection algorithms," *J Opt Theory & Apps*, 2010
Duchi, Agarwal, Wainwright, "Dual averaging for distributed optimization," *IEEE T Auto Control,* 2011
Chen and Sayed, "Diffusion adaptation strategies for distributed optimization," *IEEE T Sig Proc*, 2012
Jakovetic, Xavier, and Moura, "Fast distributed gradient methods," submitted, 2012

Synchronous or Asynchronous?

- Need neighbors values to update

$$z_i(t+1) = \sum_{j=1}^{n} P_{i,j} z_j(t) - g_i(t) \qquad P_{i,j} > 0 \Leftrightarrow (i,j) \in E$$

- Could wait to receive values from all neighbors
  - **But** then the whole network moves at the pace of the slowest node
- Motivates asynchronous communications
- Implication: time-varying update weights $P_{i,j}(t)$
- Allows to also model:
  - Communication delays
  - Time-varying inter-communication intervals

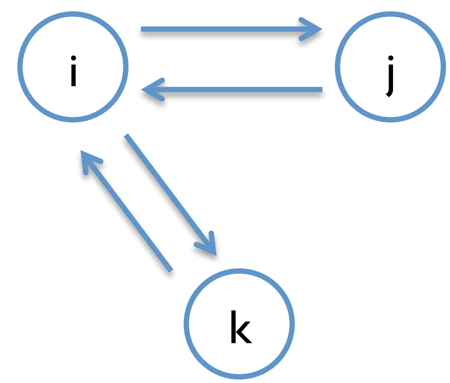
Push-Pull vs. Push (or Pull)

- Pairwise Push-Pull protocols cause deadlocks

- Need to finish one update before processing the next

$$z_i(t+1) = z_j(t+1) = \frac{z_i(t) + z_j(t)}{2}$$

$$z_k(t+1) = z_k(t) \quad \text{for } k \neq i, j$$



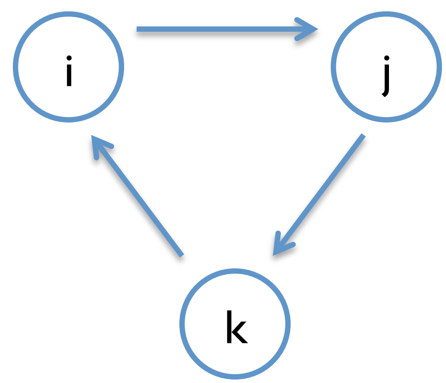
Push-Pull vs. Push (or Pull)

- Pairwise Push-Pull protocols cause deadlocks

- Need to finish one update before processing the next

$$z_i(t+1) = z_j(t+1) = \frac{z_i(t) + z_j(t)}{2}$$

$$z_k(t+1) = z_k(t) \quad \text{for } k \neq i, j$$



- Motivates using Push-only protocol

Doubly-Stochastic Weights *P* ?

- Resigned to using asynchronous push protocols

$$z_i(t+1) = \sum_{j=1}^{n} P_{i,j} z_j(t) - g_i(t)$$

Doubly-Stochastic Weights *P* ?

- Resigned to using asynchronous push protocols

$$z(t+1) = P(t)z(t)$$

Doubly-Stochastic Weights *P* ?

- Resigned to using asynchronous push protocols

$$z(t+1) = P(t)z(t)$$

- Need $\prod_{t=1}^{\infty} P(t) \to \frac{1}{n}\mathbf{1}\mathbf{1}^T$ for unbiased optimization

$$\text{minimize} \quad \frac{1}{n}\sum_{i=1}^{n} f_i(x) \qquad \text{NOT} \qquad \text{minimize} \quad \sum_{i=1}^{n} \pi_i f_i(x)$$

Doubly-Stochastic Weights *P* ?

- Resigned to using asynchronous push protocols

$$z(t+1) = P(t)z(t)$$

- Need $\prod_{t=1}^{\infty} P(t) \to \frac{1}{n}\mathbf{1}\mathbf{1}^T$ for unbiased optimization

$$\text{minimize} \quad \frac{1}{n}\sum_{i=1}^{n} f_i(x) \qquad \text{NOT} \qquad \text{minimize} \quad \sum_{i=1}^{n} \pi_i f_i(x)$$

- But asynchronous push protocols cannot be doubly stochastic
  - Each node controls a row or column of P, but not both
  - (Both would require synchronous coordination)

Push-Sum Distributed Averaging

- Initialize $z_i(0) \in \mathbb{R}^d, w_i(0) = 1$
- Send $\big(P_{i,j} z_i(t), P_{i,j} w_i(t)\big)$ to neighbor (P column stochastic)
- Receive $\big\{\big(P_{j,i} z_j(t'), P_{j,i} w_j(t')\big)\big\}$ from neighbors j
  - Buffer incoming messages while sending and computing
- Update

$$z_i(t+1) = \sum_{\text{queue}} P_{j,i} z_j(t') \qquad w_i(t+1) = \sum_{\text{queue}} P_{j,i} w_j(t')$$

- Theorem: $$\frac{z_i(t+1)}{w_i(t+1)} \longrightarrow \frac{1}{n}\sum_{i=1}^{n} z_i(0)$$

Kempe, Dobra, Gherke, "Gossip-based computation of aggregate information" *FOCS,* 2003
Bénézit Blondel, Thiran, Tsitsiklis, Vetterli, "Weighted gossip," *ISIT*, 2010
Tsianos, Lawlor, Rabbat, "Push-sum distributed dual averaging," *CDC*, 2012
Dominguez-Garcia, Hadjicostis, Vaidya, "Robust average consensus over packet dropping links," *CDC*, 2012

Experiments

- n=15 nodes
  - Open MPI v1.4.4
  - Armadillo v2.3.91 (linked to LAPACK and BLAS)

- Test problem:

$$f_i(x) = \sum_{j=1}^{M} (x - c_{j|i})^T (x - c_{j|i})$$

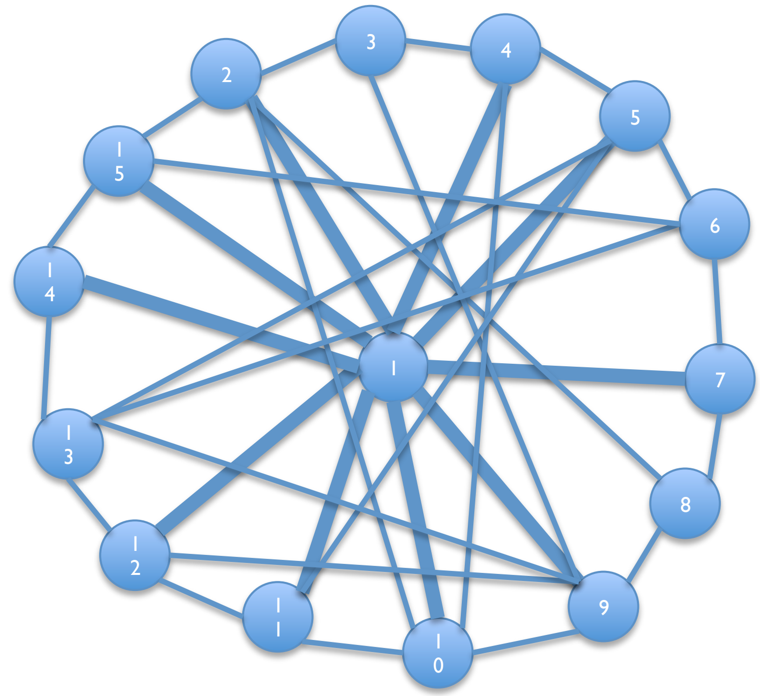
$$x, c_{j|i} \in \mathbb{R}^{5,000}$$

$$M = 500$$

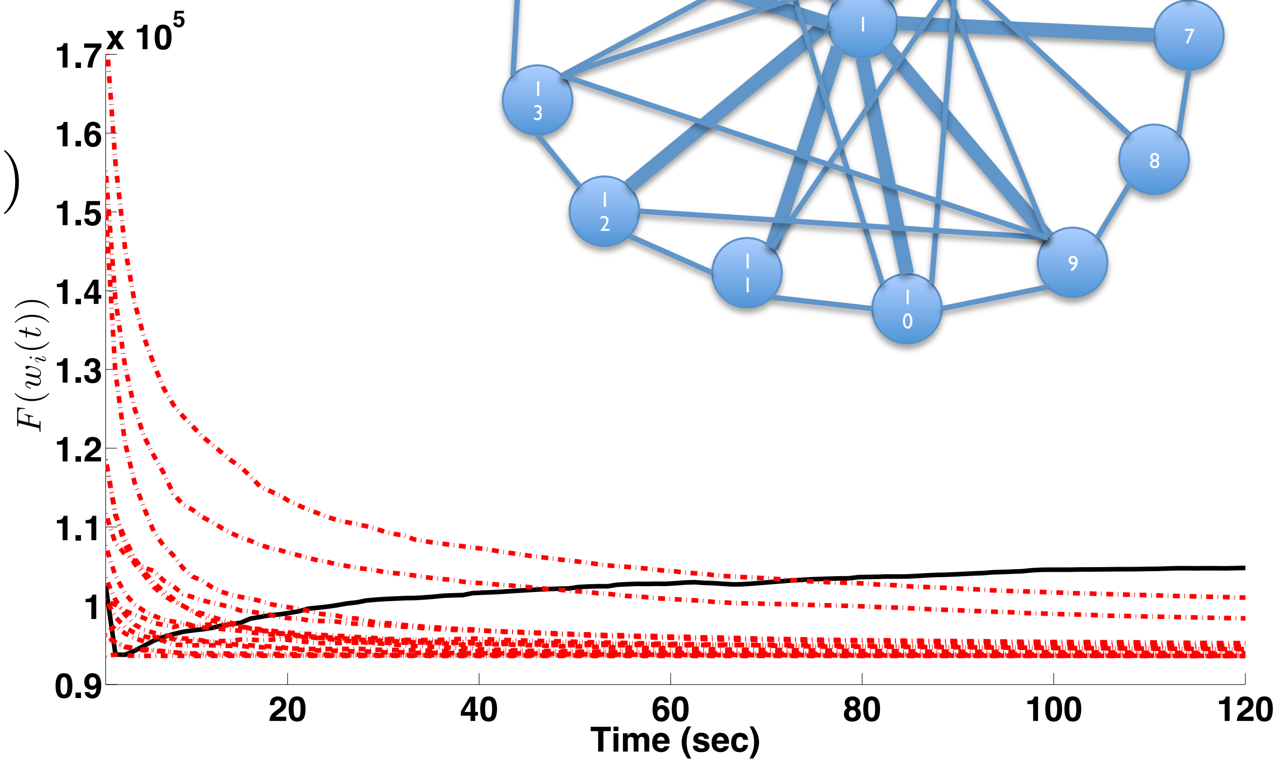
$$\mathcal{X} = \mathcal{B}(0, 2 \max_{i,j} \|c_{j|i}\|)$$

Unbalanced Network Topology

- One node communicates more than others

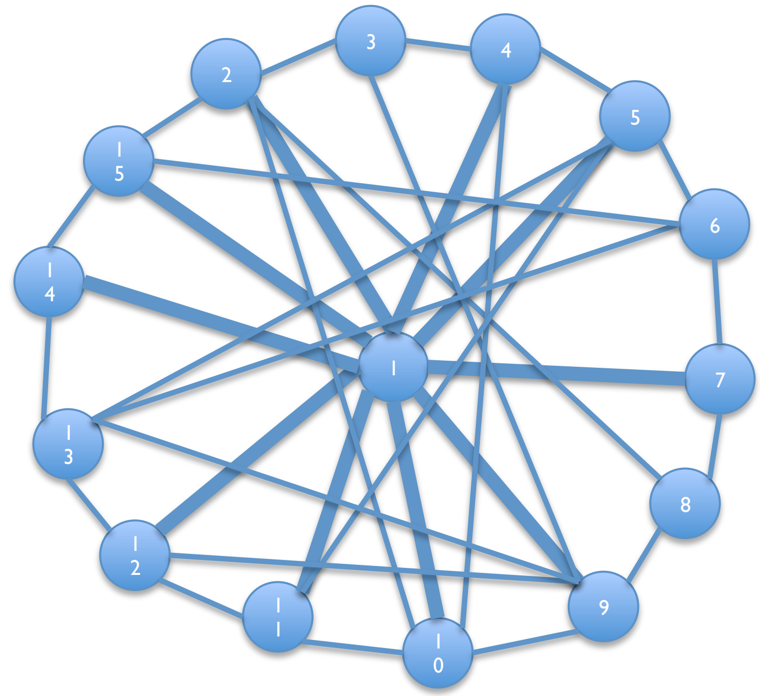


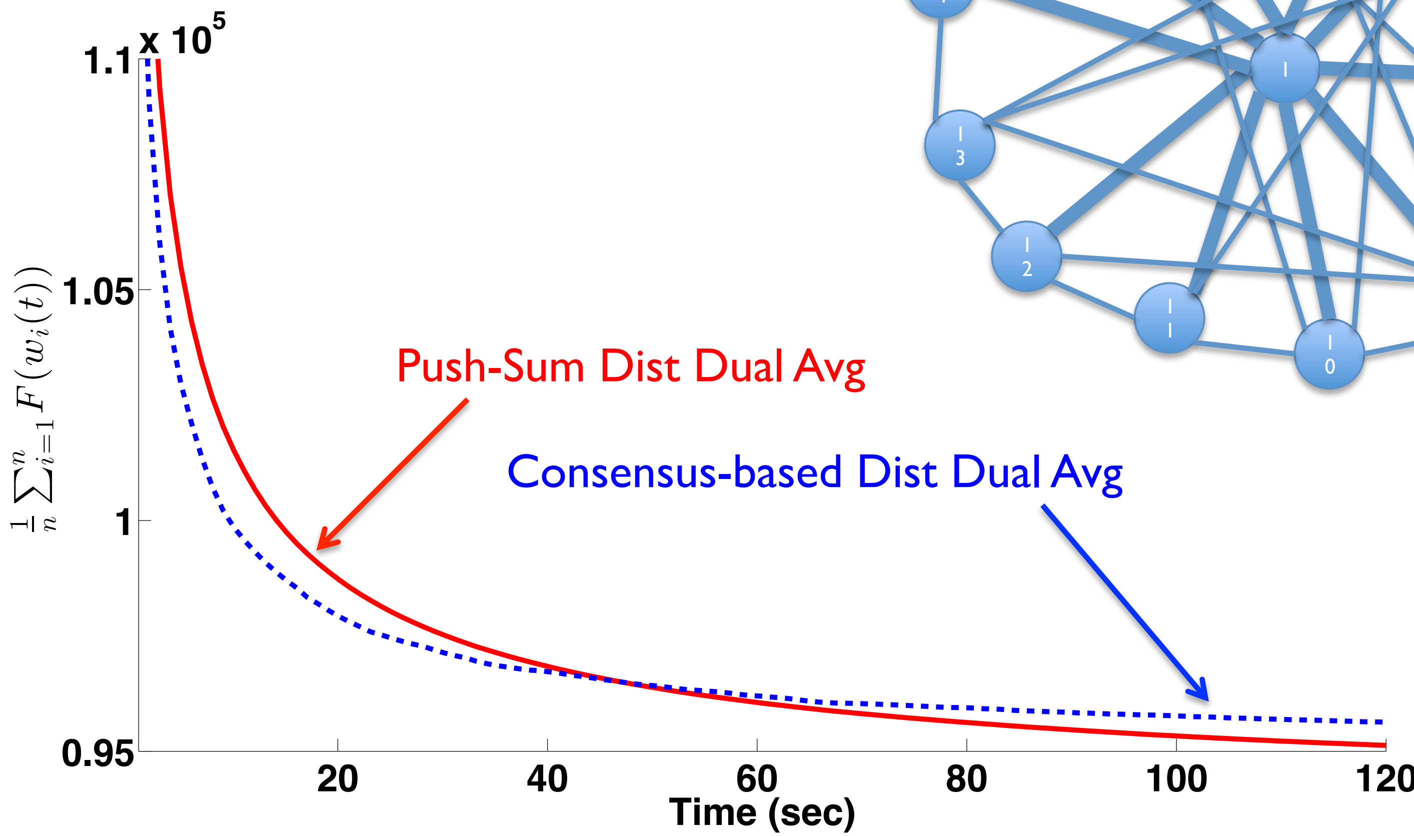
$$F(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$



Consensus vs. Push-Sum

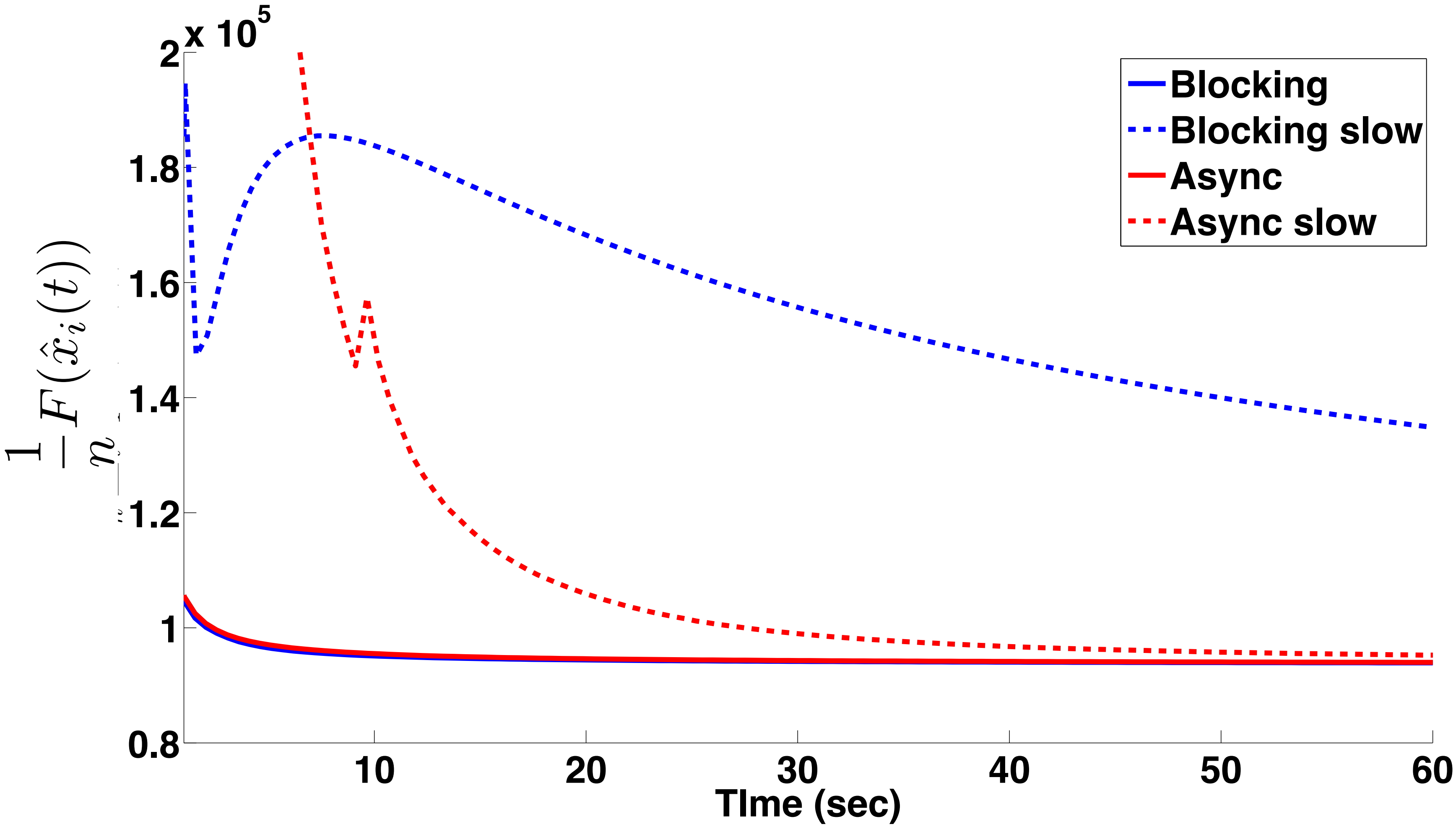
- Delays bias objective





Synchronous vs. Asynchronous

- G = K_n



Summary

- Communication costs can greatly affect the performance of distributed algorithms
- Comparing performance in terms of iterations can be deceiving
  - Iterations involve communication and computation
  - Tradeoff is problem- and system-specific
- Communication becomes less important with time
  - Have something interesting "to say" before communicating
- Communication protocols: averaging, asynchronous, push-based

michael.rabbat@mcgill.ca
http://www.ece.mcgill.ca/~mrabba1