Decentralized Optimization under Asynchrony and Delays

Wotao Yin (UCLA/Math)

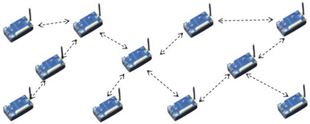
joint with: Tianyu Wu (Math), Kun Yuan (EE), Qing Ling (USTC/Auto), Ali Sayed (EE)

Emerging Wireless Networks @ IPAM — February 7, 2017

Background

Decentralized optimization

- $G = (V, E)$ has $n = |V|$ agents and is strongly connected



- **consensus optimization**: find a *consensus solution* to

$$\underset{x_1,\dots,x_n \in \mathbb{R}^p}{\text{minimize}} \sum_{i=1}^{n} f_i(x_i) \quad \text{subject to } x_i = x_j, \ \forall \, \text{edge}\, (i,j).$$

- **assumption**: no center, only neighbors can communicate
- **benefits**: no long-dist communication, privacy, fault tolerance

Applications

- Sensor networks (signal processing, tracking, ...)
- Distributed control (UAVs, cars, ...)
- Distributed learning (classification, dictionary learning, ...)

Consensus averaging (product of stochastic matrices)

- **data:** each agent i has a number y_i
- **goal:** compute $\bar{y} = \sum_{i=1}^{n} y_i$
- if G is a complete graph, then **trivial** to solve.
- in general, an **iterative algorithm**:
  - **strategy:** average with neighbors's iterates

$$x_i^{k+1} \leftarrow \sum_{j \in N_i \cup \{i\}} w_{ij} x_j^k$$

  - thus, entire network iterates (matrix-form)

$$\mathbf{x}^k = W\mathbf{x}^{k-1} = W^2\mathbf{x}^{k-2} = \cdots = W^k\mathbf{x}^0.$$

  - product of sto matrices (Touri-Nedic'12), optimize W (Lin-Boyd'04), PushSum (Kempe et al'03) ...

Decentralized gradient descent

- **consensus average** is equivalent to

$$\underset{x_1,\ldots,x_n}{\text{minimize}} \sum_{i=1}^{n} |x_i - y_i|^2 \quad \text{subject to } x_i = x_j, \ \forall\,\text{edge } (i,j).$$

- more general, **consensus minimization**

$$\underset{x_1,\ldots,x_n}{\text{minimize}} \sum_{i=1}^{n} f_i(x_i) \quad \text{subject to } x_i = x_j, \ \forall\,\text{edge } (i,j).$$

- decentralized gradient descent (Nedic-Ozdaglar'09, related to *diffusion*):

$$x_i^{k+1} \leftarrow \sum_{j \in N_i \cup \{i\}} w_{ij} x_j^k - \alpha \nabla f_i(x_i^k)$$

- easy to implement, easy to generalize
- if α is fixed, converge to *approximate, non-consensual solution*
- so, use either a small α or diminishing $\alpha = O(1/k^{\epsilon})$, $\epsilon \in (0,1]$

EXTRA (Shi et al'14)

- much faster than DGD, converge with a fixed α
- **three-point iteration**

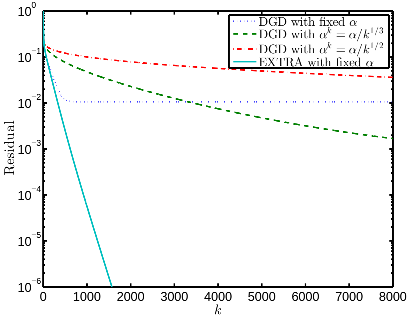
$$\mathbf{x}^{k+1} \leftarrow (W+I)\mathbf{x}^{k} - \frac{1}{2}(W+I)\mathbf{x}^{k-1} - \alpha\Big(\nabla \mathbf{f}(\mathbf{x}^{k}) - \nabla \mathbf{f}(\mathbf{x}^{k-1})\Big)$$

- interpretation: **DGD with correction**

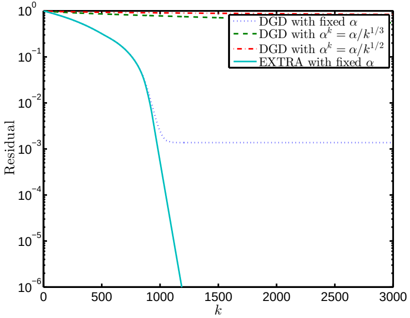
$$\mathbf{x}^{k+1} \leftarrow W\mathbf{x}^{k} - \alpha \nabla \mathbf{f}(\mathbf{x}^{k}) + \underbrace{\sum_{i=0}^{k-1} \frac{1}{2}(W-I)\mathbf{x}^{i}}_{\text{correction}}$$

- also from **linearized ADMM** or **monotone operator splitting**
- generalized to proximable functions PG-EXTRA (Shi et al'15) and Nesterov acceleration (Ye et al'15)

Example: decentralized least squares $f_i = \|A_i x_i - b_i\|^2$



Example: decentralized sum of Huber functions $f_i = h(A_i x_i - b_i)$



Decentralized ADMM (Schizas et al'08)

- **ADMM**:

$$\underset{x,y}{\text{minimize}}\ f(x) + g(y) \quad \text{subject to } Ax + By = b.$$

  f, g can be nonsmooth. Alternates two simpler subproblems.

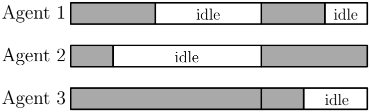
- **ADMM reformulation for decentralized optimization** ():

$$\begin{aligned} \underset{\{x_i\}_{i\in V},\{y_{ij}\}_{(i,j)\in E}}{\text{minimize}} \quad & \textstyle\sum_{i\in V} f_i(x_i) \\ \text{subject to} \quad & x_i = y_{ij},\ x_j = y_{ij},\ \forall (i,j) \in E \end{aligned}$$

- ADMM alternates between two steps
  - update x_i while fixing y_{ij}, by each agent
  - update y_{ij} and dual var while fixing x_i's, between each edge (i,j)
- also very fast

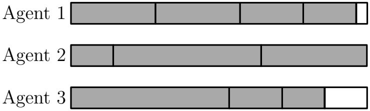
Go Asynchronous

Sync versus Async



Synchronous

(wait for the slowest)



Asynchronous

(non-stop, no wait)

How to synchronize?

Use:

- global clock, coordinator
- barrier, memory locks, semaphore, mutex, interrupt mask
- conditional variables, atomic variables, while-loop wait

which lead to *synchronization overheads*

Speed comparisons

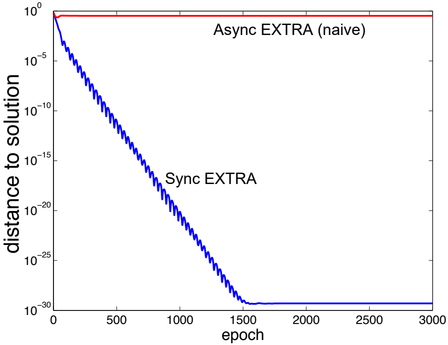
CPU speed $\gg$ streaming speed $\gg$ response speed

	async speedup
48-core workstation	5$\sim$30x
cluster	10$\sim$100x
decentralized	significant

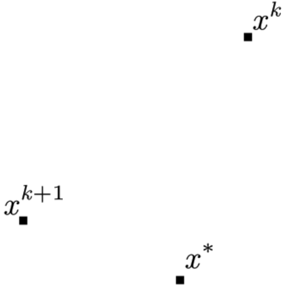
Naive approach work?

Keep existing algorithms, just add **random activation** and/or **delays**

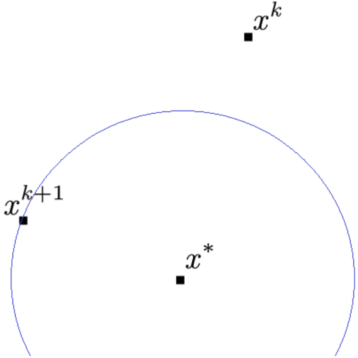
- async DGD: still converge :-)
- async EXTRA or D-ADMM: fails to converge :-(



Async does not always work

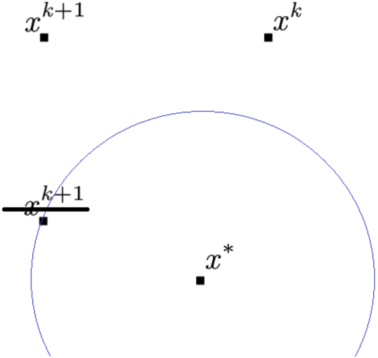


Async does not always work



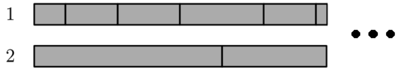
Async does not always work

x_2 update is delayed; distance to solution increases!



Async does not always work

If x_1 is updated more frequently than x_2



then a divergent example is easy to find.

How to make async work?

- use a nice *iteration operator* T, for example,
  - $\Rightarrow$ sufficient objective descent, or
  - $\Rightarrow$ nonexpansive toward the solution
- skillfully select i_k (index of kth coordinate update)
- or both

History of Async Algorithms

Brief history of async algorithms

- **1969** – a linear equation solver by Chazan and Miranker;
- **1978** – fixed-point problems by Baudet under the **absolute-contraction**[1]
- next 20–30 years, many papers on linear, nonlinear and differential equations
- **1989** – *Parallel and Distributed Computation: Numerical Methods* by Bertsekas and Tsitsiklis.
- **2000** – Review by Frommer and Szyld.
- **1991** – gradient-projection itr assuming a local linear-error bound by Tseng
- **2001** – domain decomposition assuming strong convexity by Tai & Tseng

[1] An operator $T : \mathbb{R}^n \to \mathbb{R}^n$ is absolute-contractive if $|T(x) - T(y)| \leq P|x - y|$, component-wise, where $|x|$ denotes the vector with components $|x_i|,\ i = 1, \ldots, n$, and $P \in \mathbb{R}_+^{n \times n}$ and $\rho(P) < 1$.

Brief history of async algorithms

- **2011** – Hogwild!, JellyFish, Lian'15, Parallel stochastic gradient descent.
- **2013** – Liu eta al'13, Liu-Wright'14, Sto coordinate descent (CD) for convex composite minimization.
- **2015** – Hsieh et al., Sto dual CD for regression problems.
  ARock: sto coordinate update for fixed-point problems, ADMM etc.
- **2016** – Davis, Cannelli et al: Nonconvex sto-CD. Hannah-Y.'16, Peng at al.'16: "unbounded" delay.
- Decentralized: **async EXTRA**, Eisen et al'16: quasi-Newton
- Async sto (splitting/distributed/incremental) methods: Wei-Ozdaglar'13, Iutzeler et al'13, Zhang-Kwok'14, Hong'14, Chang et al'15

ARock framework

ARock[2]: Async-parallel coordinate update

- problem: $x = T(x)$, where $x = (x_1, \ldots, x_m)$
- sub-operator $S_i(x) := x_i - (T(x))_i$
- **algorithm:** each agent randomly picks $i_k \in \{1, \ldots, m\}$:

$$x_i^{k+1} \leftarrow \begin{cases} x_i^k - \eta_k S_i(x^{k-d_k}), & \text{if } i = i_k \\ x_i^k, & \text{otherwise.} \end{cases}$$

- **assumptions:** nonexpansive T, no locking (d_k is a vector), atomic update
- **guarantee:** almost sure weak convergence under proper η_k

[2] Peng-Xu-Yan-Y. SISC'16

Convergence results

Definitions: m is # coordinates, τ is the maximum delay.

Theorem (convergence)

Assume that T is nonexpansive and has a fixed point. Let $(x^k)_{k\geq 0}$ be the sequence generated by ARock with the step sizes $\eta_k \in [\eta_{\min}, \frac{1}{1+2\frac{\tau}{\sqrt{m}}})$, $\forall k$. Then, with probability one, $(x^k)_{k\geq 0}$ weakly converges to a fixed point of T.

Theorem (linear rate)

If S is quasi-μ-strongly monotone if $\langle x-y, Sx-Sy\rangle \geq \mu\|x-y\|^2$ for any $x \in \mathcal{H}$ and $y \in \mathrm{zer}S := \{y \in \mathcal{H} : Sy = 0\}$, then with certain fixed step size

$$\mathbb{E}\|x^k - x^*\|^2 \leq c^k \cdot \|x^0 - x^*\|^2, \text{ with } c < 1.$$

Unbounded delays[3] with known distribution

- $j_{k,i}$: delay of x_i at iteration k
- $P_\ell := \Pr[\max_i\{j_{k,i}\} \geq \ell]$: iteration-independent distribution of max delay
- $\exists\, B \ni \forall k, |j_{k,i} - j_{k,i'}| < B$: x_i's delays are evenly old at each iteration

Theorem

Assume that T is nonexpansive and has a fixed point. Fix $c \in (0,1)$. Use fixed step size $\eta = cH$ for either of the following cases:

1. *if $\sum_\ell (\ell P_\ell)^{1/2} < \infty$, set $H = \left(1 + \frac{1}{\sqrt{m}} \sum_\ell P_\ell^{1/2}(\ell^{1/2} + \ell^{-1/2})\right)^{-1}$*
2. *if $\sum_\ell \ell P_\ell^{1/2} < \infty$, set $H = \left(1 + \frac{2}{\sqrt{m}} \sum_\ell P_\ell^{1/2}\right)^{-1}$*

Then, with probability one, $x^k \rightharpoonup x^* \in \mathrm{Fix}T$.

[3] R.Hannah and W.Yin'16

Arbitrary unbounded delays[4]

- $j_{k,i}$: async delay of x_i at iteration k
- $j_k = \max_i\{j_{k,i}\}$: max delay at iteration k
- $\liminf j_k < \infty$: all but finitely many iterations have a bounded delay

Theorem

Assume that T is nonexpansive and has a fixed point. Fix $c \in (0,1)$ and $R > 1$. Use step sizes

$$\eta_k = c\left(1 + \frac{R^{j_k - 1/2}}{\sqrt{m}(R-1)}\right)^{-1}.$$

Then, with probability one, $x^k_{\text{bnd}} \rightharpoonup x^* \in \mathrm{Fix}T$.

- Optionally optimize R based on $\{j_k\}$.

[4] R.Hannah and W.Yin'16

Convergence theory: take-home messages

- Has rigorous convergence guarantees, even without memory locks
- **Speed** depends on load balance (LB):
  - good LB: async is *noticeably faster* than sync
  - mediocre LB: async is *significantly faster* than sync
  - poor LB: async is *order-of-magnitude faster* than sync
- **Scalability**: assume: m blocks and good LB
  - blocks fully coupled: scale up to $p \sim \sqrt{m}$ agents
  - blocks loosely coupled: scale up to $p \sim m$ agents

Async EXTRA

Async EXTRA overview

- *uncoordinated*: agents start and complete at any time
- delays Okay, either bounded or unbounded
- however, random activations and independent delays

Async EXTRA technical overview

- rewrite sync-EXTRA into a fixed-point iteration:

$$\mathbf{z}^{k+1} = \mathcal{T}(\mathbf{z}^k)$$

  where $\mathcal{T}$ has some *contractive property*, $\mathbf{z}^k = (\mathbf{x}^k, \text{dual var}^k)$

- **define kth iteration** at completion of kth update, by some agent i_k

- **async iteration**:

$$\mathbf{z}_i^{k+1} = \begin{cases} \mathbf{z}_i^k + \eta(\mathcal{T} - I)_i(\hat{\mathbf{z}}^k), & i = i_k, \\ \mathbf{z}_i^k, & i \neq i_k. \end{cases}$$

  where $\hat{\mathbf{z}}^k$ is delayed information and $\eta \leq 1$ is damping.

- **analysis:** find the weakest assumptions so that $\mathbf{z}^k \to \mathbf{z}^*$.

Terminology of async decentralized algorithms

- **single activation**: a random agent/edge at a time, but no overlap or delay (Boyd et al'06, Dimakis et al'10)
- **multi-activation**: many random agents/edges at a time, no overlap or delay (Iutzeler et al'13, Lorenzo et al'12, Wei-Ozdaglar'13, Hong-Chang'15)
- **zero delay**: random activations, link failures, arrivals of information (Nedic-Olshevsky'15, Zhao-Sayed'15)
- **fixed delay**: coordinated, no delay after adding dummy nodes/edges (Tsianos-Rabbat'12)
- **ideal**: *uncoordinated* (start an update at any time, run for any duration), allowing *delay*s, no global clock

Async EXTRA

- s_i are convex Lipschitz-differentiable, r_i are proximable
- **formulation:**

$$\begin{aligned} \underset{x^1,\cdots,x^n \in \mathbb{R}^p}{\text{minimize}} \quad & \sum_{i=1}^n s_i(x^i) + \sum_{i=1}^n r_i(x^i), \\ \text{subject to} \quad & x^1 = x^2 = \cdots = x^n. \end{aligned}$$

Async EXTRA algorithm

Algorithm 1: Async EXTRA

initialization: $\{x^{i,0}\}, \{y^{e,0}\}$, counter $k = 0$;

while *every agent i asynchronously* **do**

  compute (30) with whatever information it has;

  send $x^{i,k+1}$ and $\{y^{e,k+1}\}_{e \in \mathcal{L}_i}$ to neighbors;

$$
\begin{cases}
\text{computing:} \\
\widetilde{x}^{i,k+1} = \mathbf{prox}_{\alpha r_i}\Big(\sum_{j \in \mathcal{N}_i} w_{ij} x^{j,k-\tau_j^k} - \alpha \nabla s_i(x^{i,k-\tau_i^k}) - \sum_{e \in \mathcal{E}_i} v_{ei} y^{e,k-\delta_e^k}\Big) \\
\widetilde{y}^{e,k+1} = y^{e,k-\delta_e^k} + \left(v_{ei} x^{i,k-\tau_i^k} + v_{ej} x^{j,k-\tau_j^k}\right), \ \forall e \in \mathcal{L}_i; \\
\text{damped updates:} \\
x^{i,k+1} = x^{i,k} + \eta_i \left(\widetilde{x}^{i,k+1} - x^{i,k-\tau_i^k}\right), \\
y^{e,k+1} = y^{e,k} + \eta_i \left(\widetilde{y}^{e,k+1} - y^{e,k-\delta_e^k}\right), \ \forall e \in \mathcal{L}_i,
\end{cases}
\tag{1}
$$

Deriving Async EXTRA

- **original matrix form:**

$$\begin{aligned} \underset{X \in \mathbb{R}^{n\times p}}{\text{minimize}} \quad & \mathbf{s}(X) + \mathbf{r}(X), \\ \text{subject to} \quad & (I - W)X = 0. \end{aligned}$$

- **scaled incidence matrix** V such that $V^T V = \frac{1}{2}(I - W)$
- **same problem, but constraints (metric) are changed:**

$$\begin{aligned} \underset{X \in \mathbb{R}^{n\times p}}{\text{minimize}} \quad & \mathbf{s}(X) + \mathbf{r}(X), \\ \text{subject to} \quad & VX = 0. \end{aligned}$$

- apply **Condat-Vu primal-dual splitting**:

$$\begin{cases} Y^{k+1} = Y^k + VX^k, \\ X^{k+1} = \mathbf{prox}_{\alpha\mathbf{r}}[X^k - \alpha\nabla\mathbf{s}(X^k) - V^\top(2Y^{k+1} - Y^k)]. \end{cases}$$

- **eliminate** Y^{k+1} for 2nd line, use $W = I - 2V^TV$, arrive at

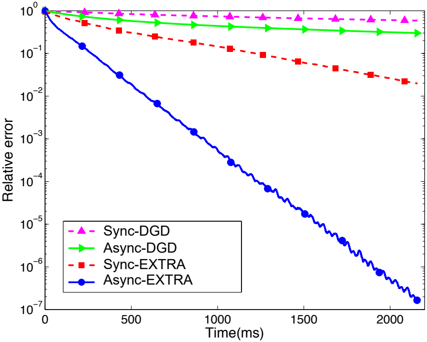
$$\begin{cases} Y^{k+1} = Y^k + VX^k, \\ X^{k+1} = \mathbf{prox}_{\alpha\mathbf{r}}[WX^k - \alpha\nabla\mathbf{s}(X^k) - V^\top Y^k]. \end{cases}$$

- this final iteration operator is **nonexpansive**, thus **ARock** is applicable

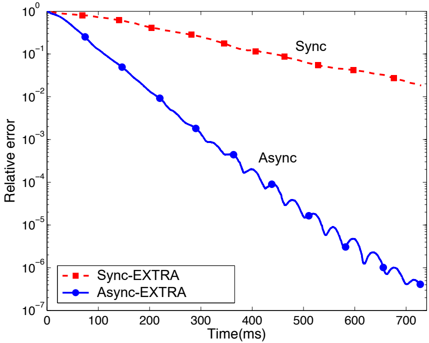
Simulation setup

- **10 nodes** randomly placed in a 30×30 area
- **14 edges** (i, j) under $\mathrm{dist}(i, j) < 15$
- simulated **computing times** $\sim \exp(1/(2 + |\bar{\mu}|))$ and $\bar{\mu} \sim N(0, 1)$
- simulated **communication delays** $\sim \exp(1/0.6)$
- **compare rel.errors**: sync-DGD, async-DGD, sync-EXTRA, async-EXTRA

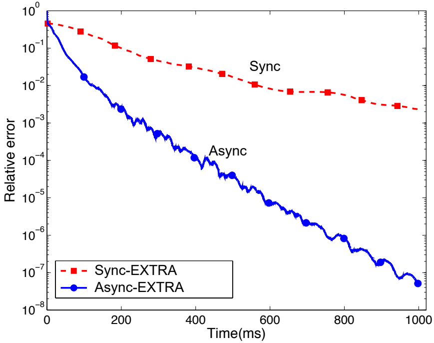
Decentralized ℓ_1 compressed sensing



Decentralized classification (sparse log. regression)



Decentralized matrix completion (Ling et al'11)



Summary

async algorithm:

- async reduces idle time, communication congestion, coordination
- surprising many parallel algorithms still work if async'd

not presented: using *expander graphs* reduces communication in decentralized optimization (Chow et al'16)

Thank you!

Acknowledgements: NSF

All papers in this talk can be found online. Paper of this work.