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

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

- **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)
    $$x^{k} = WX^{k-1} = W^{2}X^{k-2} = \cdots = W^{k}X^{0}.$$  

- 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

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

- more general, **consensus minimization**

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

- 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 \(\alpha\) is fixed, converge to **approximate, non-consensual solution**

- so, use either a small \(\alpha\) or diminishing \(\alpha = O(1/k^\epsilon), \ \epsilon \in (0, 1]\)
EXTRA (Shi et al’14)

- much faster than DGD, converge with a fixed $\alpha$

- **three-point iteration**

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

- interpretation: **DGD with correction**

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

- 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:**

\[
\min_{x,y} f(x) + g(y) \quad \text{subject to} \quad Ax + By = b.
\]

\(f, g\) can be nonsmooth. Alternates two simpler subproblems.

- **ADMM reformulation for decentralized optimization:**

\[
\min_{\{x_i\}_{i \in V}, \{y_{ij}\}_{(i,j) \in E}} \sum_{i \in V} f_i(x_i)
\]

subject to \(x_i = y_{ij}, x_j = y_{ij}, \forall (i,j) \in E\)

- 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

<table>
<thead>
<tr>
<th></th>
<th>async speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>48-core workstation</td>
<td>5~30x</td>
</tr>
<tr>
<td>cluster</td>
<td>10~100x</td>
</tr>
<tr>
<td>decentralized</td>
<td>significant</td>
</tr>
</tbody>
</table>
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 :-(

![Graph showing convergence of Sync EXTRA and Async EXTRA (naive)]
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 $k$th 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 $T : \mathbb{R}^n \rightarrow \mathbb{R}^n$ 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.


- 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\textsuperscript{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 \( i_k \in \{1, \ldots, m\} \):
  \[
  x_{i_k}^{k+1} \leftarrow \begin{cases} 
  x_{i_k}^k - \eta_k S_i(x_{i_k}^{k-d_k}), & \text{if } i = i_k \\
  x_{i_k}^k, & \text{otherwise}.
  \end{cases}
  \]

- **assumptions**: nonexpansive \( T \), no locking \((d_k \text{ is a vector})\), atomic update

- **guarantee**: almost sure weak convergence under proper \( \eta_k \)

\textsuperscript{2}Peng-Xu-Yan-Y. SISC’16
Convergence results

Definitions: \( m \) is \# coordinates, \( \tau \) is the maximum delay.

**Theorem (convergence)**

Assume that \( T \) is nonexpansive and has a fixed point. Let \( (x_k^k)_{k \geq 0} \) 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, \( (x_k^k)_{k \geq 0} \) weakly converges to a fixed point of \( T \).

**Theorem (linear rate)**

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

\[
\mathbb{E}\|x_k^k - x^*\|^2 \leq c^k \cdot \|x_0 - x^*\|^2,
\]

with \( c < 1 \).
Unbounded delays with known distribution

- $j_{k,i}$: delay of $x_i$ at iteration $k$
- $\mathcal{P}_\ell := \Pr[\max_i\{j_{k,i}\} \geq \ell]$: 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 $\sum_\ell (\ell \mathcal{P}_\ell)^{1/2} < \infty$, set $H = \left(1 + \frac{1}{\sqrt{m}} \sum_\ell \mathcal{P}_\ell \mathcal{P}_\ell^{1/2} (\ell^{1/2} + \ell^{-1/2})\right)^{-1}$
2. if $\sum_\ell \ell \mathcal{P}_\ell^{1/2} < \infty$, set $H = \left(1 + \frac{2}{\sqrt{m}} \sum_\ell \mathcal{P}_\ell^{1/2}\right)^{-1}$

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

--- 

$^3$R.Hannah and W.Yin’16
Arbitrary unbounded delays

- \( j_{k,i} \): async delay of \( x_i \) at iteration \( k \)
- \( j_k = \max_i \{j_{k,i}\} \): max delay at iteration \( k \)
- \( \lim \inf j_k < \infty \): 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, \( x_{bnd}^k \rightarrow x^* \in \text{Fix}T \).

- 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:

\[ z^{k+1} = T(z^k) \]

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

- define \( k \)th iteration at completion of \( k \)th update, by some agent \( i_k \)

- async iteration:

\[
\begin{align*}
    z^{k+1}_i &= \begin{cases} 
        z_i^k + \eta (T - I)_i (\hat{z}^k), & i = i_k, \\
        z_i^k, & i \neq i_k.
    \end{cases}
\end{align*}
\]

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

- analysis: find the weakest assumptions so that \( z^k \to z^* \).
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 *delays*, no global clock
Async EXTRA

- \( s_i \) are convex Lipschitz-differentiable, \( r_i \) are proximable

- **formulation:**

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{n} s_i(x^i) + \sum_{i=1}^{n} r_i(x^i), \\
\text{subject to} & \quad x^1 = x^2 = \cdots = x^n.
\end{align*}
\]
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;

\[
\begin{aligned}
\text{computing:} \\
\tilde{x}^{i,k+1} &= \text{prox}_{\alpha r_i} \left( \sum_{j \in \mathcal{N}_i} w_{ij} x^{j,k-\tau^k_j} - \alpha \nabla s_i(x^{i,k-\tau^k_i}) - \sum_{e \in \mathcal{E}_i} v_{ei} y^{e,k-\delta^k_e} \right) \\
\tilde{y}^{e,k+1} &= y^{e,k-\delta^k_e} + (v_{ei} x^{i,k-\tau^k_i} + v_{ej} x^{j,k-\tau^k_j}), \quad \forall e \in \mathcal{L}_i;
\end{aligned}
\]

\[
\begin{aligned}
\text{damped updates:} \\
x^{i,k+1} &= x^{i,k} + \eta_i \left( \tilde{x}^{i,k+1} - x^{i,k-\tau^k_i} \right), \\
y^{e,k+1} &= y^{e,k} + \eta_i \left( \tilde{y}^{e,k+1} - y^{e,k-\delta^k_e} \right), \quad \forall e \in \mathcal{L}_i,
\end{aligned}
\]
Deriving Async EXTRA

- **original matrix form:**

\[
\begin{align*}
\text{minimize} & \quad s(X) + r(X), \\
\text{subject to} & \quad (I - W)X = 0.
\end{align*}
\]

- **scaled incidence matrix** \( V \) such that \( V^T V = \frac{1}{2} (I - W) \)

- **same problem, but constraints (metric) are changed:**

\[
\begin{align*}
\text{minimize} & \quad s(X) + r(X), \\
\text{subject to} & \quad VX = 0.
\end{align*}
\]
• apply **Condat-Vu primal-dual splitting**: 
\[
\begin{align*}
Y^{k+1} &= Y^k + VX^k, \\
X^{k+1} &= \text{prox}_{\alpha r}[X^k - \alpha \nabla s(X^k) - V^\top (2Y^{k+1} - Y^k)].
\end{align*}
\]

• **eliminate** $Y^{k+1}$ for 2nd line, use $W = I - 2V^TV$, arrive at 
\[
\begin{align*}
Y^{k+1} &= Y^k + VX^k, \\
X^{k+1} &= \text{prox}_{\alpha r}[WX^k - \alpha \nabla s(X^k) - V^\top Y^k].
\end{align*}
\]

• this final iteration operator is **nonexpansive**, thus **ARock** is applicable
Simulation setup

- **10 nodes** randomly placed in a $30 \times 30$ area

- **14 edges** $(i, j)$ under $\text{dist}(i, j) < 15$

- 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 $\ell_1$ compressed sensing
Decentralized classification (sparse log. regression)
Decentralized matrix completion (Ling et al’11)

<table>
<thead>
<tr>
<th>Time (ms)</th>
<th>Relative error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sync-EXTRA</td>
<td></td>
</tr>
<tr>
<td>Async-EXTRA</td>
<td></td>
</tr>
<tr>
<td>Sync</td>
<td></td>
</tr>
<tr>
<td>Async</td>
<td></td>
</tr>
</tbody>
</table>
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.