Decentralized Consensus Optimization with Asynchrony and Delays

Tianyu Wu[1], Kun Yuan[2], Qing Ling[3], Wotao Yin[1], and Ali H. Sayed[2]

[1]Department of Mathematics, [2]Department of Electrical Engineering, University of California, Los Angeles
[3]Department of Automation, University of Science and Technology of China

***Abstract*—We propose an asynchronous, decentralized algorithm for consensus optimization. The algorithm runs over a network of agents, where the agents perform local computation and communicate with neighbors. We design the algorithm so that the agents can compute and communicate independently at different times and for different durations. This reduces the waiting time for the slowest agent or longest communication delay and also eliminates the need for a global clock.**

Mathematically, the algorithm involves both primal and dual variables, uses fixed step-size parameters, and provably converges to the exact solution under a bounded delay assumption and a random agent assumption. When running synchronously, the algorithm performs just as well as existing competitive synchronous algorithms such as PG-EXTRA, which diverges without synchronization. Numerical experiments confirm the theoretical findings and illustrate the performance of the proposed algorithm.

***Index Terms*—decentralized, asynchronous, delay, consensus optimization.**

I. Introduction and Related Work

This paper considers a connected network of n agents that cooperatively solve the *consensus optimization* problem

$$\begin{aligned} &\underset{x\in\mathbb{R}^p}{\text{minimize}} \quad \bar{f}(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x), \\ &\text{where } f_i(x) := s_i(x) + r_i(x),\ i = 1,\dots,n. \end{aligned} \tag{1}$$

We assume that the functions s_i and $r_i : \mathbb{R}^p \to \mathbb{R}$ are convex *differentiable* and possibly *nondifferentiable* functions, respectively. We call $f_i = s_i + r_i$ a *composite* objective function. Each s_i and r_i are kept private by agent $i = 1, 2, \cdots, n$, and r_i often serves as the regularization term or the indicator function to a certain constraint on the optimization variable $x \in \mathbb{R}^p$ that is common to all agents. Decentralized algorithms rely on agents' local computation, as well as the information exchange between agents. Such algorithms are generally robust to failure of critical relaying agents and scalable with network size.

In decentralized computation, especially with heterogeneous agents or due to processing and communication delays, it can be inefficient or impractical to synchronize multiple nodes and links. To see this, let $x^{i,k} \in \mathbb{R}^p$ be the local variable of agent i at iteration k, and let $X^k = [x^{1,k}, x^{2,k}, \dots, x^{n,k}]^\top \in \mathbb{R}^{n\times p}$ collect all local variables, where k is the iteration index. In a synchronous implementation, in order to perform an iteration that updates the entire X^k to X^{k+1}, all agents will need to wait for the slowest agent or be held back by the slowest communication. In addition, a clock coordinator is necessary for synchronization, which can be expensive and demanding to maintain in a large-scale decentralized network.

This work was supported in part by NSF grants CCF-1524250, DMS-1317602, ECCS-1407712, and ECCS-1462397, NSF China grant 61573331, and DARPA project N66001-14-2-4029. Emails: {wuty11,kunyuan,wotaoyin,sayed}@ucla.edu, qingling@mail.ustc.edu.cn

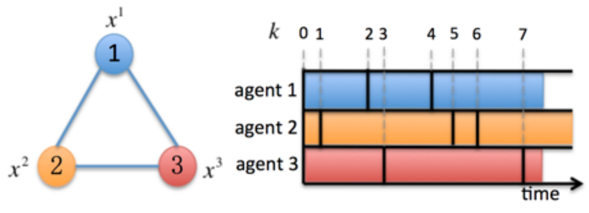

Fig. 1: Network and uncoordinated computing.

Motivated by these considerations, this paper proposes an asynchronous decentralized algorithm where actions by agents are not required to run synchronously. To allow agents to compute and communicate at different moments, for different durations, the proposed algorithm introduces delays into the iteration — the update of X^k can rely on delayed information received from neighbors. The information may be several-iteration out of date. Under uniformly bounded (but arbitrary) delays and that the next update is done by a random agent[1], this paper will show that the sequence $\{X^k\}_{k\geq 0}$ converges to a solution to problem (1) with probability one.

What can cause delays? Clearly, communication latency introduces delays. Furthermore, as agents start and finish their iterations independently, one agent may have updated its variables while its neighbors are working on their current iterations that still use old (i.e., delayed) copies of those variables; this situation is illustrated in Fig. 1. For example, before iteration 3, agents 1 and 2 have finished updating their local variables $x^{1,2}$ and $x^{2,1}$ respectively, but agent 3 is still relying on delayed neighboring variables $\{x^{1,0}, x^{2,0}\}$, rather than the updated variables $\{x^{1,2}, x^{2,1}\}$, to update $x^{3,3}$. Therefore, both computation and communication cause delays.

A. *Relationship to certain synchronous algorithms*

The proposed algorithm, if running synchronously, can be algebraically reduced to PG-EXTRA [1]; they solve problem

[1]The index i of the agent responsible for any kth update is random and independent of those responsible for the earlier updates $1,\dots,k-1$.

(1) with a fixed step -size parameter and are typically faster than algorithms using diminishing step sizes. Also, both algorithms generalize EXTRA [2], which only deals with differentiable functions. However, divergence (or convergence to wrong solutions) can be observed when one runs EXTRA and PG-EXTRA in the *asynchronous* setting, where the proposed algorithm works correctly. The proposed algorithm in this paper must use additional dual variables that are associated with edges, thus leading to a moderate cost of updating and communicating the dual variables.

The proposed algorithm is also very different from decentralized ADMM [2]–[4] except that both algorithms can use fixed parameters. Distributed consensus methods [5], [6] that rely on fixed step-sizes can also converge fast, albeit only to *approximate* solutions.

Several useful diffusion strategies [7]–[12] have also been developed for solving *stochastic* decentralized optimization problems where realizations of random cost functions are observed at every iteration. To keep adaptation and continuous learning alive, these strategies employ fixed step-sizes, and they converge fast to a small neighborhood of the true solution. The diffusion strategies operate on the primal domain, but they can outperform some primal-dual strategies in the stochastic setting due to the influence of gradient noise [13]. These studies focused on synchronous implementations. Here, our emphasis is on asynchronous networks, where delays are present and become critical, and also on deterministic optimization where convergence to the true solution of problem (1) is desired.

B. Related decentralized algorithms under different settings

Our setting of asynchrony is different from *randomized single activation*, which is assumed for the randomized gossip algorithm [14], [15]. Their setting activates only one edge at a time and does *not* allow any delay. That is, before each activation, computation and communication associated with previous activations must be completed, and only one edge in each neighborhood can be activated at any time. Likewise, our setting is different from *randomized multi-activation* such as [16], [17] for consensus averaging, and [18]–[21] for consensus optimization, which activate multiple edges each time and still do not allow any delay. These algorithms can be alternatively viewed as synchronous algorithms running in a sequence of varying subgraphs. Since each iteration waits for the slowest agent or longest communication, a certain coordinator or global clock is needed.

Our setting is also different from [22]–[25], in which other sources of asynchronous behavior in networks are allowed, such as different arrival times of data at the agents, random on/off activation of edges and neighbors, random on/off updates by the agents, and random link failures, etc. Although the results in these works provide notable evidence for the resilience of decentralized algorithms to network uncertainties, they do not consider delays.

We also distinguish our setting from the *fixed-communication-delay* setting [26], [27], where the information passing through each edge takes a fixed number of iterations to arrive. Different edges can have different such numbers, and agents can compute with only the information they have, instead of waiting. As demonstrated in [26], this setting can be transferred into *no communication delay* by replacing an edge with a chain of dummy nodes. Information passing through a chain of τ dummy nodes simulates an edge with a τ-iteration delay. The computation in this setting is still synchronous, so a coordinator or global clock is still needed. Other works [26], [28] consider *random communication delays* in their setting. However, they are only suitable for consensus averaging, not generalized for the optimization problem (1).

Our setting is identical to the setting outlined in Section 2.6 of [29], where the introduced *asynchronous decentralized ADMM* allows both computation and communication delays. Our algorithm, however, handles composite functions and avoids solving possibly complicated subproblems.

C. Contributions

This paper introduces an asynchronous, decentralized algorithm for problem (1) that provably converges to an optimal solution assuming that the next update is performed by a random agent and that communication is subject to arbitrary but bounded delays. If running synchronously, the proposed algorithm is as fast as the competitive PG-EXTRA algorithm except, for asynchrony, the proposed algorithm involves updating and communicating additional edge variable. When the proposed algorithm runs asynchronously, it eliminates waits and becomes significantly faster.

Our asynchronous setting is considerably less restrictive than the settings under which existing non-synchronous or non-deterministic decentralized algorithms are proposed. In our setting, the computation and communication of agents are uncoordinated. A global clock is not needed.

D. Notation

Each agent i holds a local variable $x^i \in \mathbb{R}^p$, whose value at iteration k is denoted by $x^{i,k}$. We introduce variable X to stack all local variables x^i:

$$X := \begin{bmatrix} — & (x^1)^\top & — \\ & \vdots & \\ — & (x^n)^\top & — \end{bmatrix} \in \mathbb{R}^{n\times p}. \quad (2)$$

We further define functions

$$\mathbf{s}(X) := \sum_{i=1}^{n} s_i(x^i), \quad \mathbf{r}(X) := \sum_{i=1}^{n} r_i(x^i), \quad (3)$$

as well as

$$\mathbf{f}(X) := \sum_{i=1}^{n} f_i(x^i) = \mathbf{s}(X) + \mathbf{r}(X). \quad (4)$$

The gradient of $\mathbf{s}(X)$ is defined as

$$\nabla\mathbf{s}(X) := \begin{bmatrix} — & \left(\nabla s_1(x^1)\right)^\top & — \\ & \vdots & \\ — & \left(\nabla s_n(x^n)\right)^\top & — \end{bmatrix} \in \mathbb{R}^{n\times p}. \quad (5)$$

The inner product on $\mathbb{R}^{n\times p}$ is defined as

$$\langle X, \widetilde{X}\rangle := \mathrm{tr}(X^\top \widetilde{X}) = \sum_{i=1}^{n}(x^i)^\top \widetilde{x}^i, \quad \forall X, \widetilde{X} \in \mathbb{R}^{n\times p}. \quad (6)$$

II. Algorithms

Consider a strongly connected undirected network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with $|\mathcal{V}| = n$ and $|\mathcal{E}| = m$. By convention, all edges $(i,j) \in \mathcal{E}$ obey $i < j$. To each edge $(i,j) \in \mathcal{E}$, we assign a weight $w_{ij} > 0$, which is used by agent i to scale the data x^j it receives from agent j. Likewise, let $w_{ji} = w_{ij}$ for agent j. If $(i,j) \notin \mathcal{E}$, then $w_{ij} = w_{ji} = 0$. For i, $\mathcal{N}_i$ denotes the neighborhood of agent i (including i itself), and $\mathcal{E}_i$ denotes the set of all edges connected to i.

Let $W = [w_{ij}] \in \mathbb{R}^{n\times n}$ denote the weight matrix, which is symmetric and assumed to be doubly stochastic. Such W can be generated through the maximum-degree [30] or Metropolis-Hastings rules [30]. It is easy to verify that $\mathrm{null}\{I - W\} = \mathrm{span}\{\mathbb{1}\}$. Introduce the diagonal matrix $D \in \mathbb{R}^{m\times m}$ with diagonal entries $D_{e,e} = \sqrt{w_{ij}/2}$ for each edge $e = (i,j)$. Let $C = [c_{ei}] \in \mathbb{R}^{m\times n}$ be the incidence matrix of $\mathcal{G}$, and define

$$V := DC \in \mathbb{R}^{m\times n} \quad (7)$$

as the scaled incidence matrix and $V = [v_{ei}]$. It is easy to verify the following statement:

Proposition 1 (Matrix factorization identity)**.** *When W and V are generated according to the above description, it holds that*

$$V^\top V = \frac{1}{2}(I - W). \quad (8)$$

■

A. Proposed primal-dual algorithm

Let us reformulate Problem (1). First, it is equivalent to

$$\begin{aligned}&\underset{\{x_{(1)},\cdots,x_{(n)}\}}{\text{minimize}} && \sum_{i=1}^{n} s_i(x^i) + \sum_{i=1}^{n} r_i(x^i),\\ &\text{subject to} && x^1 = x^2 = \cdots = x^n.\end{aligned} \quad (9)$$

Since $\mathrm{null}\{I - W\} = \mathrm{span}\{\mathbb{1}\}$, Problem (9) is equivalent to

$$\begin{aligned}&\underset{X\in\mathbb{R}^{n\times p}}{\text{minimize}} && \mathbf{s}(X) + \mathbf{r}(X)\\ &\text{subject to} && (I - W)X = 0.\end{aligned} \quad (10)$$

By Proposition 1, Problem (10) is further equivalent to

$$\begin{aligned}&\underset{X\in\mathbb{R}^{n\times p}}{\text{minimize}} && \mathbf{s}(X) + \mathbf{r}(X)\\ &\text{subject to} && VX = 0,\end{aligned} \quad (11)$$

which can be reformulated into the saddle-point problem

$$\max_{Y\in\mathbb{R}^{m\times p}} \min_{X\in\mathbb{R}^{n\times p}} \mathbf{s}(X) + \mathbf{r}(X) + \frac{1}{\alpha}\langle Y, VX\rangle, \quad (12)$$

where $\alpha > 0$ is a parameter and Y is the variable that stacks all local dual variables $\{y^e\}_{e=1}^m$. Notice that a similar formulation using the incidence and Laplacian matrices was also employed in [13] to derive primal-dual distributed optimization strategies over networks. Problem (12) can be solved iteratively by the primal-dual algorithm that is adapted from [31] [32]:

$$\begin{cases} Y^{k+1} = Y^k + VX^k,\\ X^{k+1} = \mathbf{prox}_{\alpha\mathbf{r}}[X^k - \alpha\nabla\mathbf{s}(X^k) - V^\top(2Y^{k+1} - Y^k)]. \end{cases} \quad (13)$$

where the proximal operator is defined as

$$\mathbf{prox}_{\alpha\mathbf{r}}(U) := \underset{X\in\mathbb{R}^{n\times p}}{\arg\min}\left\{\mathbf{r}(X) + \frac{1}{2\alpha}\|X - U\|_{\mathrm{F}}^2\right\}. \quad (14)$$

Next, in the X-update, eliminating Y^{k+1} by plugging in the Y-update and, with $I - 2V^\top V = W$, we arrive at:

$$\begin{cases} Y^{k+1} = Y^k + VX^k,\\ X^{k+1} = \mathbf{prox}_{\alpha\mathbf{r}}[WX^k - \alpha\nabla\mathbf{s}(X^k) - V^\top Y^k], \end{cases} \quad (15)$$

which computes (Y^{k+1}, X^{k+1}) from (Y^k, X^k)[2]. Applying W, V and $V^\top$ requires communication. Other operations are local.

B. Synchronous algorithm

Recursion (15) can run in a decentralized manner. To do so, we associate each row of the dual variable Y with an edge $e = (i,j) \in \mathcal{E}$, and for simplicity we let agent i store and update the variable y^e (the choice is arbitrary). We also define

$$\mathcal{L}_i := \{e = (i,j) \in \mathcal{E},\ \forall\, j > i\}, \quad (16)$$

as the index set of dual variables that agent i needs to update.

Algorithm 1 implements the iteration (15) in the synchronous setting, which requires two *synchronization barriers* in each iteration k. The first one holds computing until an agent receives all necessary input; after the agent finishes updating its variables, the second barrier prevents it from sending out information until all of its neighbors finish their computation (otherwise, an update intended for iteration $k+1$ may arrive at a neighbor too early, entering its computing still at iteration k). Note that the second barrier can be replaced by a buffer.

Algorithm 1: Synchronous algorithm based on (15)

```
Input: Starting point $\{x^{i,0}\}_{i=1}^n$, $\{y^{e,0}\}_{e=1}^m$. Set $k = 0$;
while all agents $i \in \mathcal{V}$ in parallel do
    Wait until $\{x^{j,k}\}_{j\in\mathcal{N}_i}$ and $\{y^{e,k}\}_{e\in\mathcal{E}_i}$ are received;
    Compute:
    $x^{i,k+1} = \mathbf{prox}_{\alpha r_i}\Big(\sum_{j\in\mathcal{N}_i} w_{ij}x^{j,k} - \alpha\nabla s_i(x^{i,k}) - \sum_{e\in\mathcal{E}_i} v_{ei}y^{e,k}\Big)$;
    $y^{e,k+1} = y^{e,k} + \big(v_{ei}x^{i,k} - v_{ej}x^{j,k}\big),\ \forall\, e \in \mathcal{L}_i$.
Wait until all neighbors finish computing;
Set $k \leftarrow k + 1$;
Send out $x^{i,k+1}$ and $\{y^{e,k+1}\}_{e\in\mathcal{L}_i}$ to neighbors;
```

[2]Algorithm (15) is essentially equivalent to PG-EXTRA developed in [1]

C. Asynchronous algorithm

In the asynchronous setting, each agent computes and communicates independently without any coordinator. Whenever an arbitrary agent finishes a round of its variables' updates, we let the iteration index k increase by 1 (see Fig. 1). As discussed in Sec. I, communication latency and uncoordinated computation result in delays. Therefore, in the asynchronous algorithm the agent will compute with delayed information from its neighbors. Suppose agent i is activated at iteration k, and let $\tau_j^k > 0$ and $\delta_e^k > 0$ denote the delays for agent j and edge e at iteration k. Agent i will compute

$$\begin{cases} \widetilde{x}^{i,k+1} = \mathbf{prox}_{\alpha r_i}\Big(\sum_{j\in\mathcal{N}_i} w_{ij}x^{j,k-\tau_j^k} - \alpha\nabla s_i(x^{i,k-\tau_i^k}) - \sum_{e\in\mathcal{E}_i} v_{ei}y^{e,k-\delta_e^k}\Big), \\ \widetilde{y}^{e,k+1} = y^{e,k-\delta_e^k} + \big(v_{ei}x^{i,k-\tau_i^k} + v_{ej}x^{j,k-\tau_j^k}\big),\ \forall e\in\mathcal{L}_i. \end{cases} \tag{17}$$

In recursion (17), $x^{j,k-\tau_j^k}$ is τ_j^k-iteration out of date and $y^{e,k-\delta_e^k}$ is δ_e^k-iteration out of date. To guarantee convergence, instead of letting $x^{i,k+1} = \widetilde{x}^{i,k+1}$ and $y^{e,k+1} = \widetilde{y}^{e,k+1}$ directly, we propose a relaxation step

$$\begin{cases} x^{i,k+1} = x^{i,k} + \eta_i\big(\widetilde{x}^{i,k+1} - x^{i,k-\tau_i^k}\big), \\ y^{e,k+1} = y^{e,k} + \eta_i\big(\widetilde{y}^{e,k+1} - y^{e,k-\delta_e^k}\big),\ \forall e\in\mathcal{L}_i. \end{cases} \tag{18}$$

where $\widetilde{x}^{i,k+1} - x^{i,k-\tau_i^k}$ and $\widetilde{y}^{e,k+1} - y^{e,k-\delta_e^k}$ behave as updating directions, and $\eta_i \in (0,1)$ behaves as the relaxation parameter which depends on how out of date an agent knows about the inputs from its neighbors. For agent j not activated at iteration k, its local variables remain the same, i.e. $x^{j,k+1} = x^{j,k}$ and $y^{e,k+1} = y^{e,k}, \forall e \in \mathcal{L}_j$. Algorithm 2 implements the asynchronous updates.

```
Algorithm 2: Asynchronous algorithm
Input: Starting point $\{x^{i,0}\}_{i=1}^n, \{y^{e,0}\}_{e=1}^m$. Set $k = 0$;
while each agent $i$ asynchronously do
    Compute per (17)–(18) using the information it has;
    Send out $x^{i,k+1}$ and $\{y^{e,k+1}\}_{e\in\mathcal{L}_i}$ to neighbors;
```

III. Convergence Analysis

We present our main assumptions and convergence results. As space is limited, all proofs are left to the longer report. We first introduce a new symmetric matrix $G = [I_n\ V^\top; V\ I_m]$, which can be further verified to be positive definite. Let $\rho_{\min} = \lambda_{\min}(G)$, then $\rho_{\min} > 0$. We also Let κ be the condition number of G.

Assumption 1.

1) *The functions s_i and r_i are closed, proper and convex.*
2) *The functions s_i are differentiable and satisfy:*
$$\|\nabla s_i(x) - \nabla s_i(\widetilde{x})\| \le L_i\|x - \widetilde{x}\|, \quad \forall x, \widetilde{x} \in \mathbb{R}^p,$$
where $L_i > 0$ is the Lipschitz constant.
3) *The parameter α in synchronous algorithm* (15) *and asynchronous algorithm* (17) *satisfies $0 < \alpha < 2\rho_{\min}/L$, where $L := \max_i L_i$.*

Assumption 2. *The delays $\tau_j^k, j = 1, 2, \ldots, n$ and $\delta_e^k, e = 1, 2, \ldots, m$, $\forall k$, defined in* (17) *have an upper bound $\tau > 0$.*

As we nearly finish this paper, we notice that this assumption can be relaxed using the results in the recent paper [33].

Assumption 3. *For any $k > 0$, the index i_k of the agent responsible for the kth update is random and has probability*

$$q_i := P(i_k = i) > 0.$$

The random variables $i_1, i_2, \cdots$ are independent.

Assumption 3 is satisfied under either of the following scenarios: (i) every agent i is activated following an independent Poisson process with parameter λ_i and its computation is instant, leading to $q_i = \lambda_i/(\sum_{i=1}^n \lambda_i)$; (ii) every agent i runs continuously, and the duration of each round follows the exponential distribution $\exp(\beta_i)$, leading to $q_i = \beta_i^{-1}/(\sum_{i=1}^n \beta_i^{-1})$.

Assumption 4. *The delays $\tau_j^k, j = 1, 2, \ldots, n$ and $\delta_e^k, e = 1, 2, \ldots, m$, at iteration k, are independent of the index i_k of the agent responsible for the update.*

We admit that Assumption 4 is not always practical, but it is a key assumption for our proof to go through. One of the cases for this assumption to hold is when each agent and edge always have the maximum delay τ. In reality, what happens is between this worst case and the no-delay case. Besides, Assumption 4 is also a common assumption in the recent literature of stochastic asynchronous algorithms; see [29] and the references therein.

Based on the above assumptions, we have the following theorem for $Z^k := [X^k; Y^k]$.

Theorem 1. *Let Z^* be the set of primal-dual solutions to* (12), *$\{Z^k\}_{k\ge0}$ be the sequence generated by Alg. 2, and $\eta_i = \frac{\eta}{nq_i}$ with $\eta \in (0, \eta_{\max}]$ where $\eta_{\max} < \frac{nq_{\min}}{2\tau\sqrt{\kappa q_{\min}}+\kappa}$ and $q_{\min} := \min_i q_i$. Then $\{Z^k\}_{k\ge0}$ converges to a point in Z^* with probability 1.*

This theorem guarantees that, if we run the asynchronous algorithm 2 from an arbitrary starting point X^0, then with probability 1, the sequence $\{X^k\}_{k\ge0}$ produced will converge to one of the solutions to problem (12).

IV. Numerical Experiments

In this part we simulate the performance of the proposed asynchronous algorithm (Alg. 2). We will compare it with its synchronous counterpart (Alg. 1).

The tested problem is *decentralized compressed sensing*. Each agent $i \in \{1, \cdots, n\}$ holds some measurements: $b_i = A_ix + e_i \in \mathbb{R}^{m_i}$, where $A_i \in \mathbb{R}^{m_i\times p}$ is a sensing matrix, $x \in \mathbb{R}^p$ is the common unknown *sparse* signal, and e_i is i.i.d. Gaussian noise. The goal is to recover x. The number of measurements $\sum_{i=1}^n m_i$ may be less than the number of unknowns p, so we solve the ℓ_1-regularized least squares:

$$\underset{x}{\text{minimize}}\ \frac{1}{n}\sum_{i=1}^n s_i(x) + r_i(x), \tag{19}$$

where $s_i(x) = \frac{1}{2}\|A_ix - b_i\|_2^2, \quad r_i(x) = \theta_i\|x\|_1$, and θ_i is the regularization parameter with agent i and we set $\theta_i = 0.01$.

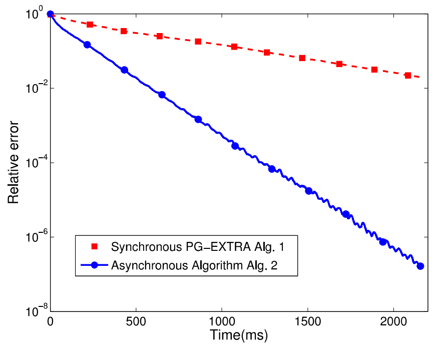

Fig. 2: asynchronous and synchronous algorithms

The tested network has 10 nodes and 14 edges. We set $m_i = 3$ for $i = 1, \cdots, 10$ and $p = 50$. The entries of A_i, e_i are independently sampled from the standard normal distribution $N(0, 1)$, and A_i is normalized so that $\|A_{(i)}\|_2 = 1$. The signal x is generated randomly with 20% nonzero elements.

We also simulate the computation and communication times. The computation time of agent i is sampled from $\exp(\mu_i)$. For agent i, μ_i is set as $2 + |\bar{\mu}|$ where $\bar{\mu} \sim N(0, 1)$. The communication time between agents are independently sampled from $\exp(1/0.6)$. After the computation time is generated, the probability q_i can be computed accordingly.

We run the synchronous Alg. 1 and the asynchronous Alg. 2 and plot the relative error $\frac{\|X^k - X^*\|_F}{\|X^0 - X^*\|_F}$ against time, as depicted in Fig. 2. X^* is the exact solution. The step-sizes for both algorithms are tuned to be 1, and for Alg. 2 we set $\eta_i = 0.0288/q_i$. From Fig. 2 we can see that both algorithms exhibit linear convergence and that Alg. 2 converges significantly faster. Within the same period (roughly 2760ms), the asynchronous algorithm finishes 21 times as many rounds of computation and communication as the synchronous counterparts, due to the elimination of waiting time.

References

[1] W. Shi, Q. Ling, G. Wu, and W. Yin, "A proximal gradient algorithm for decentralized composite optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 22, pp. 6013–6023, 2015.

[2] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[3] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in Ad hoc WSNs with noisy links – Part I: Distributed estimation of deterministic signals," *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 350–364, 2008.

[4] T.-H. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus admm," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 482–497, 2015.

[5] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[6] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

[7] J. Chen and A. H. Sayed, "Distributed Pareto optimization via diffusion strategies," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 2, pp. 205–220, 2013.

[8] J. Chen and A. H. Sayed, "On the learning behavior of adaptive networks—Part I: Transient analysis," *IEEE Transactions on Information Theory*, vol. 61, no. 6, pp. 3487–3517, 2015.

[9] J. Chen and A. H. Sayed, "On the learning behavior of adaptive networks—part II: Performance analysis," *IEEE Transactions on Information Theory*, vol. 61, no. 6, pp. 3518–3548, 2015.

[10] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, April 2014.

[11] S. Vlaski and A. H. Sayed, "Proximal diffusion for stochastic costs with non-differentiable regularizers," in *Proc. International Conference on Acoustic, Speech and Signal Processing (ICASSP)*, Brisbane, Australia, April 2015, pp. 3352–3356.

[12] S. Vlaski, L. Vandenberghe, and A. H. Sayed, "Diffusion stochastic optimization with non-smooth regularizers," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4149–4153.

[13] Z. J. Towfic and A. H. Sayed, "Stability and performance limits of adaptive primal-dual networks," *IEEE Transactions on Signal Processing*, vol. 63, no. 11, pp. 2888–2903, 2015.

[14] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2508–2530, 2006.

[15] A. G. Dimakis, S. Kar, J. M. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, 2010.

[16] J. M. Kar, S.and Moura, "Sensor networks with random links: Topology design for distributed consensus," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3315–3326, 2008.

[17] F. Fagnani and S. Zampieri, "Randomized consensus algorithms over large scale networks," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, pp. 634–649, 2008.

[18] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, "Asynchronous distributed optimization using a randomized alternating direction method of multipliers," in *Proc. IEEE Conference on Decision and Control (CDC)*, Florence, Italy, 2013, pp. 3671–3676.

[19] P. D. Lorenzo, S. Barbarossa, and A. H. Sayed, "Decentralized resource assignment in cognitive networks based on swarming mechanisms over random graphs," *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3755–3769, 2012.

[20] E. Wei and A. Ozdaglar, "On the $o(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers," in *Proc. IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Austin, USA, 2013, pp. 551–554.

[21] M. Hong and T. Chang, "Stochastic proximal gradient consensus over random networks," *arXiv preprint arXiv:1511.08905*, 2015.

[22] A. Nedic and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2015.

[23] X. Zhao and A. H. Sayed, "Asynchronous adaptation and learning over networks—Part I: Modeling and stability analysis," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 811–826, 2015.

[24] X. Zhao and A. H. Sayed, "Asynchronous adaptation and learning over networks—Part II: Performance analysis," *IEEE Transactions on Signal Processing,*, vol. 63, no. 4, pp. 827–842, 2015.

[25] X. Zhao and A. H. Sayed, "Asynchronous adaptation and learning over networks—Part III: Comparison analysis," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 843–858, 2015.

[26] K. I. Tsianos and M. G. Rabbat, "Distributed consensus and optimization under communication delays," in *Proc. Allerton Conference on Communication, Control, and Computing (Allerton)*, Monticello, USA, 2011, pp. 974–982.

[27] K. I. Tsianos and M. G. Rabbat, "Distributed dual averaging for convex optimization under communication delays," in *Proc. IEEE American Control Conference (ACC)*, Montréal, Canada, 2012, pp. 1067–1072.

[28] L. Liu, S.and Xie and H. Zhang, "Distributed consensus for multi-agent systems with delays and noises in transmission channels," *Automatica*, vol. 47, no. 5, pp. 920–934, 2011.

[29] Z. Peng, Y. Xu, M. Yan, and W. Yin, "ARock: an algorithmic framework for asynchronous parallel coordinate updates," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, pp. A2851–A2879, 2016.

[30] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014.

[31] L. Condat, "A primal–dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms," *Journal of Optimization Theory and Applications*, vol. 158, no. 2, pp. 460–479, 2013.

[32] B. C. Vũ, "A splitting algorithm for dual monotone inclusions involving cocoercive operators," *Advances in Computational Mathematics*, vol. 38, no. 3, pp. 667–681, 2013.

[33] R. Hannah and W. Yin, "On unbounded delays in asynchronous parallel fixed-point algorithms," *UCLA CAM 16-64*, 2016.