Optimal convergence rates of totally asynchronous optimization

Xuyang Wu, Sindri Magnússon, Hamid Reza Feyzmahdavian, and Mikael Johansson

Abstract—Asynchronous optimization algorithms are at the core of modern machine learning and resource allocation systems. However, most convergence results consider bounded information delays and several important algorithms lack guarantees when they operate under total asynchrony. In this paper, we derive explicit convergence rates for the proximal incremental aggregated gradient (PIAG) and the asynchronous block-coordinate descent (Async-BCD) methods under a specific model of total asynchrony, and show that the derived rates are order-optimal. The convergence bounds provide an insightful understanding of how the growth rate of the delays deteriorates the convergence times of the algorithms. Our theoretical findings are demonstrated by a numerical example.

I. INTRODUCTION

Distributed and parallel algorithms are powerful tools for solving large-scale problems. These algorithms coordinate multiple computing nodes to solve the overall problem. The coordination can be synchronous, meaning that each node needs to wait for all other nodes to conclude their computations and communications before proceeding to the next iteration [1], [2]. This is clearly inefficient: the slowest node dictates the convergence speed, systems become sensitive to single node failures, and the implementation overhead for synchronization can be large. Therefore, asynchronous algorithms that need no synchronization are often preferred [3]–[5]. However, compared to synchronous algorithms, asynchronous algorithms are more difficult to analyze, and their convergence properties are not as well understood.

Early efforts on convergence analysis of asynchronous algorithms were made in the 1980s by Bertsekas and Tsitsiklis, e.g., [6]–[8]. They considered two models for asynchrony: partial asynchrony (“bounded delays”) and total asynchrony (“unbounded delays”) and analyzed convergence for several classes of algorithms under these two models.

Recent works on asynchronous optimization include AsySPA [4], PIAG [9]–[11], Async-BCD [12]–[14], ARock [15], [16], and Asynchronous SGD [17]. However, only a few algorithms are shown to work under total asynchrony [18]–[20]. In particular, [18], [19] study Asynchronous SGD and [20] focuses on a delay-tolerant averaged proximal gradient method. Moreover, different from us, references [18]–[20] do not characterize how unbounded delays affect convergence rates and do not explore the existence of an optimal rate.

This paper studies two popular asynchronous optimization algorithms, PIAG and Async-BCD, under a particular model of total asynchrony where the rate at which the delays can grow unbounded is limited. None of these algorithms has been proven to converge under unbounded delays before. We make the following specific contributions:

• We derive explicit convergence rates for PIAG and Async-BCD under a model of total asynchrony.
• We prove that the derived convergence rates for the two methods are optimal in terms of order.
• We use the convergence bounds to provide insight and understanding of how the growth rate of delays slows down the convergence of PIAG and Async-BCD.

Notation and Preliminaries

We use \( \mathbb{N} \) to denote the set of natural numbers and \( \mathbb{N}_0 = \mathbb{N} \cup \{0\} \). Let \( |m| = \{1, \ldots, m\} \) for any \( m \in \mathbb{N} \) and represent \( x \in \mathbb{R}^d \) as \( x = (x^{(1)}, \ldots, x^{(m)}) \), where each \( x^{(i)} \in \mathbb{R}^{d^{(i)}} \) and \( \sum_{i=1}^{m} d^{(i)} = d \). We define the proximal operator of a function \( r : \mathbb{R}^d \to \mathbb{R} \) as

\[
\text{prox}_{\mu r}(x) = \arg \min_{y \in \mathbb{R}^d} r(y) + \frac{1}{2\mu} \|y - x\|^2.
\]

We say that a differentiable function \( f : \mathbb{R}^d \to \mathbb{R} \) is \( \mu \)-strongly convex if

\[
f(x + h) - f(x) \geq \nabla f(x)^T h + \frac{\mu}{2} \|h\|^2, \forall x, h \in \mathbb{R}^d;
\]

it is \( L \)-smooth if

\[
\|\nabla f(x) - \nabla f(x + h)\| \leq L\|h\|, \forall x, h \in \mathbb{R}^d;
\]

and it is \( \hat{L} \)-block-wise smooth with respect to a partition \( x = (x^{(1)}, \ldots, x^{(m)}) \) if for all \( i, j \in |m| \), and \( h^{(j)} \in \mathbb{R}^{d^{(j)}} \),

\[
\|\nabla_i f(x + U^{(j)}h^{(j)}) - \nabla_i f(x)\| \leq \hat{L}\|h^{(j)}\|.
\]

Here, \( \nabla_if(\cdot) \) is the partial gradient of \( f \) with respect to the \( i \)th block and \( U^{(j)} : \mathbb{R}^{d^{(j)}} \to \mathbb{R}^d \) maps \( h^{(j)} \in \mathbb{R}^{d^{(j)}} \) into a \( d \)-dimensional vector where the \( j \)th block is \( h^{(j)} \) and other blocks are zero. Clearly, any \( L \)-smooth function is also \( L \)-block-wise smooth, but \( \hat{L} \) may be much smaller than \( L \). For example, if \( f(x) = 0.5x^TAx \) for some \( A = (A_{ij})_{m \times m} \) where \( A_{ij} \in \mathbb{R}^{d^{(i)} \times d^{(j)}} \) is the \( ij \)-block, then \( \hat{L} = \max_{i,j \in |m|} \|A_{ij}\|_2 \) and \( L = \|A\|_2 \). For an \( L \)-smooth function \( f \) and a convex function \( r \), we say \( P(x) = f(x) + r(x) \) satisfies the proximal PL condition [21] for \( \sigma > 0 \) if

\[
s(P(x) - P^*) \leq -L\hat{P}(x), \forall x \in \text{dom}(P),
\]
where \( P^* = \min_{x \in \mathbb{R}^d} P(x) \) and \( \hat{P}(x) = \min_{y \in \mathbb{R}^d} \{ \langle \nabla f(x), y - x \rangle + \frac{L}{2} \| y - x \|^2 + r(y) - r(x) \} \).

## II. PROBLEM STATEMENT

We focus on optimization problems of the form

\[
\min_{x \in \mathbb{R}^d} P(x) = f(x) + r(x),
\]

where \( f : \mathbb{R}^d \to \mathbb{R} \) is smooth and possibly non-convex, and \( r : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\} \) is convex but possibly non-differentiable. Such a composite structure is common in, for example, machine learning where \( f \) is a loss and \( r \) is a regularizer or the indicator function of a convex set.

We consider the proximal incremental aggregated gradient (PIAG) algorithm [9] and the asynchronous block-coordinate descent (Async-BCD) method [12] to solve (3).

### A. PIAG

The algorithm solves problem (3) for \( f \) on the form

\[
f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)
\]

using the following update

\[
g_k = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x_{k-1}^{(i)}),
\]

\[
x_{k+1} = \text{prox}_{\gamma_k r}(x_k - \gamma_k g_k),
\]

where \( k \) is the iteration index, \( \tau_k^{(i)} \in [0, k] \) is the delay of the gradient \( \nabla f_i \) at iteration \( k \), and \( \gamma_k \geq 0 \) is the step-size. The update (4)–(5) is often implemented in the parameter server architecture, where each worker \( i \) computes \( \nabla f_i(x_{k-1}^{(i)}) \) and the master aggregates all the most recent local gradients to form (4) and updates the iterate using (5). The implementation of PIAG is detailed in Algorithm 1.

### B. Async-BCD

Suppose that the non-smooth function \( r \) in problem (3) is separable, i.e., for a partition \( x = (x^{(1)}, \ldots, x^{(m)}) \) with \( x^{(i)} \in \mathbb{R}^{d^{(i)}} \) and \( \sum_{i=1}^{m} d^{(i)} = d \), it holds that \( r(x) = \sum_{i=1}^{m} r^{(i)}(x^{(i)}) \forall x \in \mathbb{R}^d \). When the dimension \( d \) of \( x \) is large, one attractive method for solving (3) is the block-coordinate descent (BCD) method. At each \( k \in \mathbb{N}_0 \), this method randomly chooses \( j \in [m] \) and executes the update

\[
x_{k+1}^{(j)} = \text{prox}_{\gamma_k r^{(j)}}(x_k^{(j)} - \gamma_k \nabla_j f(x_k)),
\]

where \( \nabla_j f(\cdot) \) is the partial gradient of \( f \) with respect to the \( j \)th block \( x^{(j)} \) and \( \gamma_k \geq 0 \) is the step-size. Async-BCD implements BCD using multiple processors in a shared memory setting [15]. The decision vector is stored in shared memory and at each iteration \( k \), one worker \( i_k \in [n] \) updates

\[
x_{k+1}^{(j)} = \text{prox}_{\gamma_k r^{(j)}}(x_k^{(j)} - \gamma_k \nabla_j f(x_k)).
\]

Here, \( \hat{x}_k \) is the decision vector that worker \( i_k \) has read from shared memory and based its partial gradient computation on. A specific aspect of Async-BCD is that while \( i_k \) reads from the shared memory, other workers may be in the process of writing. Hence, \( \hat{x}_k \) itself may never have existed in the shared memory. This phenomenon is known as inconsistent read [22]. However, if we assume that each (block) write is atomic, then we can express \( x_k \) as

\[
x_k = \hat{x}_k + \sum_{j \in J_k} (x_{j+1} - x_j).
\]

where \( J_k \subseteq \{0, 1, \ldots, k\} \). The sum represents all updates that have occurred since \( i_k \) began reading \( \hat{x}_k \) until the block update is written back to memory. We call \( \tau_k = k - \min \{ j : j \in J_k \} \) the delay of \( \hat{x}_k \) at iteration \( k \). The block index \( j \) is drawn by \( i_k \) uniformly at random at time \( k - \tau_k \). Algorithm 2 details the implementation of Async-BCD.
III. MAIN RESULT

In this section, we will derive convergence results for PIAG and Async-BCD under a totally asynchronous delay model. In our setting, the totally asynchronous model of Bertsekas and Tsitsiklis [23] would allow the delays to grow unbounded, as long as no processor ceases to update and

$$\lim_{k \to +\infty} k - \tau_k = +\infty,$$  \hspace{1cm} (8)

where $\tau_k = \max_{i \in [n]} \tau_k^{(i)}$ for PIAG. Convergence rate results under this model are unlikely since it does not impose any strict bound on how quickly the delays can grow. We thus focus on a particular model of asynchrony that satisfies (8).

Assumption 1: For some $a \in (0, 1)$, $b \in [0, 1]$, and $c \geq 0$,

$$\tau_k \leq \min(k, ak^b + c), \forall k \in \mathbb{N}_0.$$  

Assumption 1 is a specialization of the delay model in [16] and it guarantees that (8) holds. It allows the delays to grow unbounded when $b \in (0, 1]$ as both $k$ and $ak^b + c$ grow unbounded when $k \to +\infty$. By varying $b \in [0, 1]$ we can move seamlessly between several interesting and important models of asynchrony. In particular,

- $b = 0$ yields bounded delays: $\tau_k \leq \min(k, a + c)$.
- $b = 1/2$ is sublinear growth: $\tau_k \leq \min(k, a\sqrt{k} + c)$.
- $b = 1$ is a linear delay bound: $\tau_k \leq \min(k, ak + c)$.

The linearly growing delay bound is the largest polynomial growth we can have, because when $b > 1$ the total asynchrony condition (8) no longer holds. However, since we require $a \in (0, 1)$ and $b \in [0, 1]$ there are delay sequences that satisfy (8) and $\tau_k \leq k$ but not Assumption 1. One such example is $\tau_k = [k - \ln(k + 1)]$. In our step-size rules, over-estimation of $a, b, c$ within their theoretical ranges still guarantees Assumption 1 but degrades the convergence rates of the two algorithms (see Theorems 1–2 in Section III), while under-estimation may break Assumption 1 and could lead to divergence. Our analysis can be extended to other delay models, e.g., those in [16] and [24].

A. PIAG

Let us first present the convergence rate guarantees for PIAG under the delays characterized by Assumption 1. For convenient notation, we introduce

$$\phi(k) = \begin{cases} k^{1-b}, & b \in (0, 1), \\ \ln k, & b = 1, \end{cases} \forall k \in \mathbb{N}_0,$$

where $b$ is the delay bound parameter in Assumption 1.

Theorem 1: Suppose that each $f^{(i)}$ is $\hat{L}_i$-smooth with respect to the partition \( x = (x^{(1)}, \ldots, x^{(m)}) \), \( r(x) = \sum_{i=1}^{m} f^{(i)}(x^{(i)}) \) with each $r^{(i)}$ being convex and closed, \( P^* := \min_x P(x) > -\infty \), and that Assumption 1 holds. Let \( \{x_k\} \) be generated by the PIAG algorithm with

$$\gamma_k = \frac{h}{\hat{L}_i(a^{k^{1-b}} + c + 1)}, \forall k \in \mathbb{N}_0,$$  \hspace{1cm} (9)

where $h \in (0, 1)$. Then,

(i) There exist $\xi_k \in \partial r(x_k) \forall k \in \mathbb{N}_0$ such that

$$\min_{t \leq k} \| \nabla f(x_t) + \xi_t \| = O(1/\phi(k)).$$

(ii) If each $f^{(i)}$ is convex and the optimal solution set of problem (3) is non-empty, then

$$P(x_k) - P^* = O(1/\phi(k)).$$

(iii) If $P$ satisfies the proximal PL-condition (2), then

$$P(x_k) - P^* = O(\lambda^k)$$

for some $\lambda \in (0, 1)$.

Proof: See Appendix A.

Table I extracts the relationship between the delay bound, admissible step-size, and convergence rates in Theorem 1.

| delay bound          | step-size (non-convex, convex) | rate (proximal PL) |
|----------------------|--------------------------------|-------------------|
| $O(k^b)$, $b < 1$    | $O(k^{-b})$                    | $O(1/k^{1-b})$    |
| $O(k)$               | $O(1/k)$                      | $O(1/\ln k)$      |

TABLE I: asynchrony, step-size, and convergence rate for PIAG (all three cases) and Async-BCD (non-convex).

When $b = 0$, i.e., delays are bounded, the rates in Table I match those of PIAG under partial asynchrony [9]–[11], [25] and those of gradient descent [21]. The table quantifies how large delays limit the admissible step-sizes and deteriorate the convergence rates, which agrees with intuition.

B. Async-BCD

Based on the block-wise smoothness assumption (1), the following theorem establishes convergence rates for Async-BCD in solving problem (3).

Theorem 2: Suppose that $f$ is $\hat{L}$-block-wise smooth with respect to the partition $x = (x^{(1)}, \ldots, x^{(m)})$, $r(x) = \sum_{i=1}^{m} r^{(i)}(x^{(i)})$ with each $r^{(i)}$ being convex and closed, $P^* := \min_x P(x) > -\infty$, and that Assumption 1 holds. Let \( \{x_k\} \) be generated by the Async-BCD algorithm with

$$\gamma_k = \frac{h}{\hat{L}(a^{k^{1-b}} + c + 1)}, \forall k \in \mathbb{N}_0,$$  \hspace{1cm} (10)

where $h \in (0, 1)$. Then,

$$\min_{t \leq k} \mathbb{E}[\| \nabla P(x_t) \|^2] = O(1/\phi(k)),$$

where $\nabla P(x_t) = \hat{L}(\text{prox}_{\frac{1}{\hat{L}}} \nabla f(x_t)) - x_t$.

Proof: See Appendix B.

In Theorem 2, $\nabla P(x) = 0$ if and only if $0 \in \partial P(x)$, i.e., $x$ is a stationary point of problem (3). When $b = 0$, our convergence rate is of the same order compared to Async-BCD under partial asynchrony [13], [14]. The relationship between delay bound, step-size, and convergence rate of Async-BCD is summarized in Table I for non-convex objective functions. Once again, a larger delay requires smaller step-sizes and leads to a slower convergence.
C. Optimal convergence rate

This subsection establishes that the convergence rates in the preceding theorems are order-optimal under Assumption 1, and not a consequence of the particular step-size policies. To show this, we first derive the optimal rates of the gradient descent method with time-varying step-sizes:

$$x_{k+1} = x_k - \gamma_k \nabla f(x_k),$$  \hspace{1cm} (11)

which is equivalent to PIAG with \(n = 1\) and Async-BCD with \(m = n = 1\), without information delay and with \(r \equiv 0\).

**Lemma 1:** Let \(L > 0\), \(N \in \mathbb{N}_0\) and \(d \in \mathbb{N}\). Also let \(\{x_k\}\) be generated by the gradient method (11) with \(\gamma_k \leq h/L\) \(\forall k \in \mathbb{N}_0\) for some \(h > 0\). Then,

(a) there exists a point \(x_0 \in \mathbb{R}^d\) and a convex and \(L\)-smooth function \(f : \mathbb{R}^d \to \mathbb{R}\) such that

$$f(x_N) - f^* \geq \frac{L\|x_0 - x^*\|^2}{2} - \frac{1}{2Nh + 1},$$

where \(x^* = \arg\min_x f(x)\) and \(f^* = f(x^*)\).

(b) if \(h < 1\), for any \(x_0 \in \mathbb{R}^d\), there exists a \(\mu\)-strongly convex and \(L\)-smooth function \(f : \mathbb{R}^d \to \mathbb{R}\) such that

$$f(x_N) - f^* \geq \frac{\mu\|x_0 - x^*\|^2}{2} \cdot (1 - h)^2N,$$

Lemma 1 indicates that the worst-case convergence rate of the gradient descent method (11) with time-varying step-sizes cannot outperform \(O(1/k)\) and linear for convex and strongly convex objective functions, respectively. These optimal rates are established under a general step-size condition: \(\gamma_k \leq h/L\) where \(h > 0\) for convex case and \(h \in (0, 1)\) for strongly convex case. For gradient descent on non-convex but smooth objective functions, the typical rate is

$$\min_{t \leq k} \|\nabla f(x_t)\|^2 \leq O(1/k),$$ \hspace{1cm} (12)

which is not likely to be significantly improved without additional problem assumptions. A \(O(1/k^2)\) rate of the same optimality criteria is provided in [26] for gradient method (11) with fixed \(\gamma_k\), but is only proved for convex \(f\).

**Theorem 3:** Under the same assumptions,

(a) the worst-case convergence rate in Theorem 1 for PIAG on convex and proximal-PL functions are order-optimal.

(b) if the typical rate (12) of gradient descent is order-optimal, so are the convergence rates of PIAG and Async-BCD in Theorems 1–2 for non-convex functions.

**Proof:** We follow [27, Section 2.2] and prove rate optimality by (a) first establishing that the algorithms converge at the given rate for all problems under consideration, and (b) then find one example for which the algorithms cannot converge faster. Part (a) is proven in Theorems 1 and 2. In the following we prove (b) by constructing an objective function and a delay sequence that satisfy the assumptions in Theorems 1–2 and showing the two algorithms cannot achieve a better convergence for the constructed example.

Let \(r \equiv 0\) and let \(f\) be \(L\)-smooth for some \(L > 0\). Then, both PIAG and Async-BCD reduce to

$$x_{k+1} = x_k - \gamma_k \nabla f(x_{k-\tau_k}).$$ \hspace{1cm} (13)

Now, consider the delay sequence

$$\tau_k = k - T_t, \quad \text{if } k \in [T_t, T_{t+1}),$$ \hspace{1cm} (14)

where \(\{T_t\}\) is defined by \(T_0 = 0\) and

$$T_{t+1} = \max\{\kappa \in \mathbb{N} : \kappa - (ak^b + c) \leq T_t\} + 1$$ \hspace{1cm} (15)

for some \(a \in (0, 1)\), \(b \in [0, 1]\), and \(c \geq 0\). In this way, for any \(k \in [T_t, T_{t+1})\), it holds that \(k - (ak^b + c) \leq T_t\). By (14), \(\tau_k \leq ak^b + c\) and \(\tau_k \leq k\), so \(\{\tau_k\}\) satisfies Assumption 1.

By substituting (14) into (13), we obtain

$$x_{k+1} = x_T - \left(\sum_{l=1}^{k} \gamma_l \nabla f(x_{T_l})\right), \quad \forall k \in [T_t, T_{t+1}).$$

This implies that \(x_k, k \in \mathbb{N}\) is obtained by performing \(\max\{t \in \mathbb{N}_0 : T_t \leq k - 1\} + 1\) steps of gradient descent starting from \(x_0\). Moreover, we prove in Appendix D that

$$\max\{t \in \mathbb{N}_0 : T_t \leq k - 1\} + 1 = O(\phi(k)).$$ \hspace{1cm} (16)

The result now follows by the optimal rates of gradient descent in Lemma 1 and the assumption: after \(O(\phi(k))\) iterations the worst-case iteration complexities cannot outperform those in Theorems 1–2. For examples of \(f\) such that the optimal rates of gradient descent (11) are exactly attained (\(O(1/k)\) for convex \(f\), linear for strongly convex \(f\), and (12) for non-convex \(f\)), the rates in Theorems 1–2 are attained exactly for the problem constructed above.

**IV. NUMERICAL EXPERIMENTS**

We demonstrate the theoretical results in Theorems 1–2 and evaluate the practical performance of the two methods under the specific total asynchrony model in simulations. We consider a binary classification problem on the training data set of RCV1 [28] using the regularized logistic regression where \(f(x) = \frac{1}{N} \sum_{i=1}^{N} \left[\log(1 + e^{-y_i(q^T x)}) + \frac{\lambda}{2} \|x\|^2\right]\) and \(r(x) = \lambda_1 \|x\|_1\). Here, \(p_i\) is the feature of the \(i\)th sample, \(q_i\) is the corresponding label, and \(N\) is the number of samples. We set \(\lambda_1 = 10^{-5}\) and consider both strongly convex (\(\lambda_2 = 10^{-1}\)) and convex (\(\lambda_2 = 0\)) problems for both methods.

**A. PIAG**

We split the \(N\) samples into \(n = 10\) batches and assign each batch to a single worker. We consider the following delay model: \(\tau^{(i)}_k = 0\) for all \(i \in [n]\). For all \(k \in \mathbb{N}\) and \(i \in [n]\), if \(\tau^{(i)}_{k-1} \leq \min\{k, ak^b + c\} - 1\), then \(\tau^{(i)}_k = \tau^{(i)}_{k-1} + 1\); Otherwise, \(\tau^{(i)}_k\) is randomly drawn from \([0, \min\{k, \lfloor ak^b + c\rfloor\}]\). We use \(a = 0.1\) and \(c = 0\), and consider \(b = 0.2, 0.6\) and 1 to evaluate the effect of delays. Note that the constructed delay sequence satisfies Assumption 1.

We plot the objective error \(P(x_k) - P^*\) generated by PIAG and its theoretical upper bounds in Theorem 1 \(O(1/\phi(k))\) for the convex case and \(O(\lambda \phi(k))\) for the strongly convex case) in Fig 1, where the bounds are obtained by substituting (20) into (18) and (19), respectively. We plot in log/log for the convex case and log/linear for the strongly convex case to distinguish theoretical and practical error.
Observe from Fig 1 that PIAG converges slowly in the convex case and quickly in the strongly convex case, deteriorates as \(b\) increases, and is below the theoretical bound, which validates Theorem 1. Although theoretically, the objective errors converge to 0 as the iteration index goes to infinity, such convergence is hard to observe in practice for large \(b\) because of the corresponding large delay and rapidly diminishing step-sizes. Moreover, the theoretical bound (dashed) is much larger than the practical error (solid) in all cases, but this is common for optimization methods since theoretical bounds reflect the worst-case convergence.

**A. Proof of Theorem 1**

The proof uses [25, Theorem 2] stated below.

**Theorem 4:** Under the conditions in Theorem 1, if

\[
\sum_{t=k-\tau_k}^k \gamma_t \leq \frac{h}{\tau} \quad \forall k \in \mathbb{N}_0
\]  

(17)

for some \(h \in (0, 1)\), then

1. There exist \(\xi_k \in \partial r(x_k) \forall k \in \mathbb{N}_0\) such that

\[
\sum_{k=1}^\infty \gamma_{k-1} \|\nabla f(x_k) + \xi_k\|^2 \leq \frac{2(h^2 - h + 1)(P(x_0) - P^*)}{1 - h}
\]

**V. CONCLUSION**

We have derived explicit convergence rates of PIAG and Async-BCD under a model of computation that allows for a broad range of totally asynchronous behaviours. The convergence rates are optimal in terms of the order of iteration index \(k\) and reflect how asynchrony affects the convergence times of the algorithms. The theoretical results were validated in simulations. We believe that the proposed techniques apply also to other asynchronous optimization algorithms, but leave such studies for future work.

**APPENDIX**

**B. Async-BCD**

We use \(n = 4\) processors and split the decision vector \(x\) evenly into \(m = 7\) blocks. We set \(\tau_0 = 0\). For all \(k \in \mathbb{N}\), \(\tau_k = \tau_{k-1} + 1 + \min\{k, \lfloor a k^b + c \rfloor\}\) and is randomly drawn from \([0, \min\{k, \lfloor a k^b + c \rfloor\}]\) otherwise. Like above, we set \(a = 0.1\) and \(c = 0\), and consider \(b = 0.2, 0.6, 1\). The resulting delay sequence satisfies Assumption 1.

Fig 2 plots the objective error \(P(x_k) - P^*\) generated by Async-BCD. We use the same scale as in PIAG, i.e., log/log in the convex case and log/log in the strongly convex case.

From Fig 2 we can see that Async-BCD tends to converge for all three values of \(b\) and the two objective functions, which demonstrates Theorem 2. As for PIAG, the objective error is theoretically guaranteed to vanish for all \(b \in [0, 1]\) as the iteration index goes to infinity, but this is hard to see in practice for large \(b\) (\(b = 0.6, 1\)) because of the corresponding large delay and quickly decaying step-sizes. In addition, by comparing Fig 2(a), (b), we note that Async-BCD converges faster for the strongly convex objective function.
(2) If each $f^{(i)}$ is convex, then by letting $a_0 = \frac{h(h+1)}{L(1-h)}$, we have
\[
P(x_k) - P^* \leq P(x_0) - P^* + \frac{1}{2a_0} \|x_0 - x^*\|^2 + \frac{1}{1 + \frac{1}{a_0} \sum_{t=0}^{k-1} \gamma_t} \geq (1-h)\|x'\|.
\]
Then by (24),
\[
f(x_N) - f^* = f(x_N) \geq \frac{\mu}{2} (1-h)^2 N \|x_0 - x^*\|^2,
\]
which completes the proof of claim (b).

C. Proof of Lemma 1

We prove claims (a) and (b) below.

**Proof of (a):** For simplicity, we assume $L = 1$ and $\|x_0 - x^*\| = 1$. Generalizing this proof to the case where $L \neq 1$ or $\|x_0 - x^*\| \neq 1$ is straightforward. The result is shown by the construction of the following convex and 1-smooth $f$:
\[
f(x) = \begin{cases} \frac{1}{2N h + 1} \|x\|^2, & \|x\| \geq \frac{1}{2N h + 1}, \\ \frac{1}{2} \|x\|^2, & \text{otherwise.} \end{cases}
\]

For this $f$, we have $x^* = 0$ and $f^* = 0$. This example is also used in [29, Theorem 3.2] to prove the optimal rates of gradient method with fixed step-sizes. Applying gradient method (11) with $x_0 = v$ on the function $f$ in (22) where $v$ is a unit vector in $\mathbb{R}^d$, we have
\[
\nabla f(x_k) = v \frac{1}{2N h + 1}, \quad x_k = (1 - \sum_{t=0}^{k-1} \gamma_t) v,
\]
so that
\[
f(x_k) = \frac{1 - \sum_{t=0}^{k-1} \gamma_t}{2N h + 1} \leq \frac{1}{2(2N h + 1)^2} = \frac{1}{4N h + 2} \left( \frac{4N h + 2 - 2 \sum_{t=0}^{k-1} \gamma_t - 1}{2N h + 1} \right).
\]

In addition, $\sum_{t=0}^{k-1} \gamma_t \leq kh$ due to $\gamma_t \leq \gamma \forall t \in \mathbb{N}_0$. Then, we have
\[
f(x_N) \geq \frac{1}{4N h + 2} = \frac{L\|x_0 - x^*\|^2}{2}. \frac{1}{2N h + 1}.
\]

This completes the proof of (a).

**Proof of (b):** We construct the following $f$:
\[
f(x) = \frac{1}{2} x^T A x,
\]
where $A \in \mathbb{R}^{d \times d}$ is a symmetric matrix whose minimal eigenvalue is $\mu$ and maximal eigenvalue is $L$. Clearly, $f$ is $\mu$-strongly convex and $L$-smooth. Applying gradient method (11) with any $x_0$ will give
\[
\nabla f(x_k) = A x_k, \quad x_k = \Pi_{t=0}^{k-1} (I - \gamma_t A) x_0,
\]
so that
\[
f(x_k) = \frac{1}{2} x_0^T (\Pi_{t=0}^{k-1} (I - \gamma_t A))^T A (\Pi_{t=0}^{k-1} (I - \gamma_t A)) x_0 \geq \frac{\mu}{2} \|x_0 - x^*\|^2.
\]

Because each $I - \gamma_t A$ is positive-semidefinite due to $\gamma_k \leq h/L \leq 1/L$, for any $x' \in \mathbb{R}^d$ we have
\[
\|I - \gamma_t A\| x' \geq (1 - \gamma_t L) \|x'\| \geq (1 - h) \|x'\|.
\]

Then by (24),
\[
f(x_N) - f^* = f(x_N) \geq \frac{\mu}{2} (1-h)^2 N \|x_0 - x^*\|^2,
\]
which completes the proof of claim (b).
D. Proof of (16)

By (15), \( \kappa - (a\kappa b + c) > T_t \) for any \( \kappa > T_{t+1} - 1 \). Hence,
\[
T_{t+1} \geq T_t + aT_{t+1} + c \geq T_t + aT_t + c,
\]
where the second step uses \( T_{t+1} \geq T_t \) from the first step.

**Case 1:** \( b \in [0, 1) \). The proof uses induction. Let \( \eta = a(1 - b)2^{-\frac{t}{\tau_N}} \). Suppose that \( T_t \geq \eta t \) for some \( t \in \mathbb{N}_0 \), which holds naturally at \( t = 0 \). Then, by (25),
\[
T_{t+1} - (\eta(t + 1))^{\frac{1}{\tau_N}} \geq \eta t^{\frac{1}{\tau_N}} + a(\eta t)^{\frac{1}{\tau_N}} + c - (\eta(t + 1))^{\frac{1}{\tau_N}} = (\eta t^{\frac{1}{\tau_N}} + a(\eta t)^{\frac{1}{\tau_N}} + c)
\]
\[
\geq \quad (\eta t^{\frac{1}{\tau_N}} + a\eta t^{\frac{1}{\tau_N}} - (1 + \frac{1}{t})^{\frac{1}{\tau_N}}) + c. \tag{26}
\]

Note that \( v'(t) = -\frac{1}{\tau_N}(\frac{a}{\eta} - \frac{1}{1 - b}) t^{\frac{1}{\tau_N}} \), which satisfies
\[
v'(t) \leq -\frac{1}{\tau_N}(\frac{a}{\eta} - \frac{1}{1 - b}2^{\frac{-t}{\tau_N}}) = 0 \text{ when } t \geq 1.
\]
Hence, \( v(t) \) is monotonically decreasing on \([1, +\infty)\) and \( v(t) \geq \lim_{t \to +\infty} v(t) = 0 \) for all \( t \geq 1 \), which, together with (26), gives
\[
T_{t+1} \geq \eta(t + 1)^{\frac{1}{\tau_N}}. \quad \text{Hence, } t \leq \frac{T_{t+1} - 1}{\eta}\text{ for all } t \in \mathbb{N}_0 \text{ and max}\{t \in \mathbb{N}_0 : T_t \leq k - 1\} \leq \frac{k - 1}{\eta}, \quad \text{i.e., } \max\{t \in \mathbb{N}_0 : T_t \leq k - 1\} + 1 = O(k^{1-\eta}).
\]

**Case 2:** \( b = 1 \). By (25), \( T_{t+1} \geq (1 + a)T_t \geq (1 + a)\eta t \geq (1 + a)t^{\frac{1}{\tau_N}}, \) so that \( t \leq \ln \frac{T_{t+1}}{1 + a} + 1 \). Thus, \( \max\{t \in \mathbb{N}_0 : T_t \leq k - 1\} \leq \ln \frac{k - 1}{1 + a} - 1 \), i.e., \( \max\{t \in \mathbb{N}_0 : T_t \leq k - 1\} + 1 = O(\ln k) \).

Concluding the two cases, we complete the proof.

**References**

[1] R. Johnson and T. Zhang, “Accelerating stochastic gradient descent using predictive variance reduction,” *Advances in neural information processing systems*, vol. 26, 2013.

[2] X. Qian, Z. Qu, and P. Richtárik, “L-svrg and l-katyusha with arbitrary sampling,” 2021.

[3] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, “Asynchronous distributed optimization using a randomized alternating direction method of multipliers,” in 52nd *IEEE conference on decision and control*, IEEE, 2013, pp. 3671–3676.

[4] J. Zhang and K. You, “AsySPA: An exact asynchronous algorithm for convex optimization over digraphs,” *IEEE Transactions on Automatic Control*, vol. 65, no. 6, pp. 2494–2509, 2019.

[5] M. Assran, A. Aytekin, H. R. Feyzmahdavian, M. Johansson, and M. G. Rabbat, “Advances in asynchronous parallel and distributed optimization,” *Proceedings of the IEEE*, vol. 108, no. 11, pp. 2013–2031, 2020.

[6] D. P. Bertsekas, “Distributed asynchronous computation of fixed points,” *Mathematical Programming*, vol. 27, no. 1, pp. 107–120, 1983.

[7] J. Tsitsiklis, D. Bertsekas, and M. Athans, “Distributed asynchronous deterministic and stochastic gradient optimization algorithms,” *IEEE transactions on automatic control*, vol. 31, no. 9, pp. 803–812, 1986.

[8] D. P. Bertsekas and J. N. Tsitsiklis, “Convergence rate and termination of asynchronous iterative algorithms,” in *Proceedings of the 3rd International Conference on Supercomputing*, 1989, pp. 461–470.

[9] A. Aytekin, H. R. Feyzmahdavian, and M. Johansson, “Analysis and implementation of an asynchronous optimization algorithm for the parameter server,” *arXiv preprint arXiv:1610.05507*, 2016.

[10] N. D. Vanli, M. Gurbuzbalaban, and A. Ozdaglar, “Global convergence rate of proximal incremental aggregated gradient methods,” *SIAM Journal on Optimization*, vol. 28, no. 2, pp. 1282–1300, 2018.

[11] T. Sun, Y. Sun, D. Li, and Q. Liao, “General proximal incremental aggregated gradient algorithms: Better and novel results under general scheme,” *Advances in Neural Information Processing Systems*, vol. 32, pp. 996–1006, 2019.