Lock-Free Incremental Coordinate Descent

Vien V. Mai and Mikael Johansson

***Abstract*— We study a flexible algorithm for minimizing a sum of component functions, each of which depends on a large number of decision variables. The algorithm combines aspects of incremental gradient method with that of coordinate descent. In contrast to earlier algorithms of this kind, our algorithm is lock-free and does not require synchronization of access to the shared memory. We prove convergence of the algorithm under asynchronous operation and provide explicit bounds on how the solution times depend on the degree of asynchrony. Numerical experiments confirm our theoretical results.**

I. Introduction

The recent emergence of inexpensive multi-core processors and large-scale data sets has motivated researchers to develop novel algorithms that are able to split the data and distribute the computations across multiple processors or computer clusters (see, e.g., [1]–[3] and references therein). To achieve substantial speed-ups with parallel computations, it is essential to minimize the overhead associated with synchronization and communication, so as to maximize the time when worker nodes are busy doing actual work.

In this paper, we propose a simple parallel *lock-free* algorithm that combines aspects of the incremental gradient method with that of coordinate descent. Our algorithm allows multiple worker threads to independently access a shared memory to gather the information necessary to compute partial gradients, while a master thread aggregates the mini-batch and updates the decision variable using *stale* (delayed) information. The proposed algorithm is endowed with performance guarantees in a inconsistent read model, where the coordinates within each mini-batch may have different levels of delay. Our contributions are summarized as follows:

- We analyze the convergence of the proposed algorithm under inconsistent read, accounting for data sparsity and information delay. The result shows that the iterates converge in expectation to a ball around the optimum.
- We explicitly characterize the convergence factors and show that the effects of asynchrony such as delay and inconsistent read are asymptotically negligible.

A. Notation and Preliminaries

We use $\mathbb{R}$ and $\mathbb{N}$ to denote the set of real and nonnegative natural numbers, respectively. For a given set, we use $|\cdot|$ to denote its cardinality. The expectation operator is denoted by $\mathbb{E}\{\cdot\}$; e_i denotes the ith natural basis vector in $\mathbb{R}^n$; $\mathrm{R}_j = e_j e_j^\top$ denotes the projection matrix associated with the jth coordinate; and $\|\cdot\|$ denotes the Euclidean norm.

A function f is called μ-strongly convex on $\mathbb{R}^n$ if there exists a positive constant μ such that

$$f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle + \frac{\mu}{2}\|y-x\|^2$$

holds for all $x, y \in \mathbb{R}^n$. We say that a continuously differentiable function f is L-smooth on $\mathbb{R}^n$ if its gradient is L-Lipschitz, that is

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x-y\|, \ \forall x, y \in \mathbb{R}^n.$$

II. Problem Setup

We consider the unconstrained minimization of the form

$$\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{m}\sum_{i=1}^{m} f_i(x), \qquad (1)$$

where $f_1, f_2, \ldots, f_m$ are L-smooth and convex functions, and f is μ-strongly convex. In the context of machine learning, many cost functions are sparse in the sense that m is very large but each individual $f_i(x)$ depends only on a very small number of components of x (c.f. [1], [3]). In other words, we can express f_i as $f_i(x_{\mathcal{E}_i})$, where $x_{\mathcal{E}_i}$ is the vector x restricted to the coordinates in $\mathcal{E}_i \in \{1, \ldots, n\}$.

We consider an *inconsistent read* model of memory access. Such a model allows multiple independent processes to operate simultaneously on different components of the decision vector. This reflects state-of-the art implementations of shared memory algorithms, where the full decision vector is not protected during an update, only the individual elements, and only while they are written (accomplished using low-level atomic read operations). From a mathematical analysis perspective, the inconsistent model implies that the value of the decision vector $\hat{x}$ read from the shared memory by some process may be different from any value of x that has ever existed in the shared memory; see Figure 1. In effect, the iterations describing the status of the shared memory have heterogeneous and time-varying information delays, which can potentially lead to poor convergence or even divergence. One can enforce consistent read (which would result in homogeneous delays in every component of the decision vector) by locking the shared memory while the master thread is updating the complete vector. However, such a mechanism degrades parallel performance significantly since worker threads will be idle while waiting for the master thread to complete. Therefore, while most analysis results for asynchronous optimization algorithms assume consistent read, the actual implementations are lock-free (e.g., [1], [2],

This work was not supported by any organization

V. V. Mai and M. Johansson are with the Department of Automatic Control, School of Electrical Engineering and ACCESS Linnaeus Center, Royal Institute of Technology (KTH), SE-100 44 Stockholm, Sweden. Emails: {maivv, mikaelj}@kth.se.

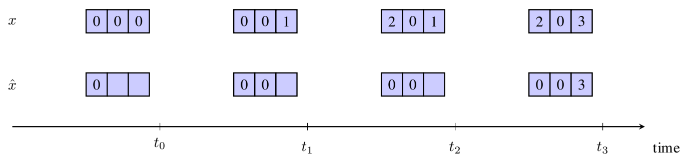

Fig. 1. A demonstration of inconsistent read model. Suppose that at time t_0, the worker thread T_1 is running the reading step and first reads the first component of $x(t_0)$; at time t_1, the master thread updates the third component of x to 1, and T_1 reads the second component of x; at time t_2, the master thread updates the first component of x to 2; and finally, at time t_3, the master thread updates the third component of x to 3, and T_1 reads the last component of x. Eventually, T_1 obtain the read value $\hat{x} = [0\ 0\ 3]^\top$, which is not a real state of x at any time point.

[4]) and do not enjoy the same theoretical guarantees (and sometimes no guarantees at all).

To parallelize the computations more efficiently, we will exploit the fact that data sparsity allows workers to run largely independently since they only have a small probability of interfering with each other. This allows to use larger step-sizes, and better convergence guarantees. To this end, we define the sparsity measures: $\bar{\nu} = \max_{1\le i\le m} |\mathcal{E}_i|$ and $\underline{\nu} = \min_{1\le i\le m} |\mathcal{E}_i|$. Similarly to [1], we define by Δ the maximum fraction of rows that intersect any variable as

$$\Delta = \frac{\max_{1\le j\le d} |\{i \in \{1,2,\ldots,m\} : j \in \mathcal{E}_i\}|}{m}.$$

These quantities will play a central role in characterizing the convergence of our algorithm.

III. Algorithm and Convergence Properties

In this section, we introduce a lock-free incremental coordinate descent algorithm for solving (1). We then characterize the convergence rate of the proposed algorithm.

A. Description of the algorithm

Our algorithm is designed to run fully asynchronously on a shared memory multicore architecture. The computational burden is divided amongst T worker threads, each of which has access to a shared memory for the decision variable x. Workers coordinate with the master thread independently of each other. The detailed steps are described in Algorithm 1.

In order to establish convergence of Algorithm 1, we impose the following assumptions.

Assumption 1 (Independence): The sequence $\{i_b(k)\}$ used to form the sum in Step 4 of Algorithm 1 is uniformly sampled from $\{1,2,\ldots,m\}$. In addition, we assume that the random variables $i_b(k)$ and $j_b(k)$ in Algorithm 1 are independent of $\hat{x}_b(k)$, $\forall b \in \{1,2,\ldots,B\}$ and $\forall k \in \mathbb{N}$.

Assumption 2 (Bounded Delay): The time from which a thread starts reading the iterate vector from shared memory until its computed gradient is used in an update of the shared memory is bounded. That is, for each iteration $k \in \mathbb{N}$, the delay $\tau(k)$ is finite and there exists an integer $\bar{\tau}$ such that

$$0 \le \tau(k) \le \bar{\tau}.$$

Algorithm 1 Lock-free incremental coordinate descent

```
// Master thread
Input: Initial $x_0$, mini-batch size $B$, and thread count $T$.
1: Write $x_0$ into the shared memory and initialize a container for collecting data from the $T$ worker threads.
2: for $k = 0, 1, \ldots$ do
3:     Keep popping the container until we have $B$ disjoint coordinates. Denote the index set by $\mathcal{S}(k) = \{j_1(k), j_2(k), \ldots, j_B(k)\}$.
4:     Update the components of $x(k)$ corresponding to $\mathcal{S}(k)$
           $$x(k+1) = x(k) - \frac{\gamma}{B}\sum_{b=1}^{B} |\mathcal{E}_{i_b(k)}| \mathrm{R}_{j_b(k)} \nabla f_{i_b(k)}(\hat{x}_b(k)).$$
5: end for
Output: $x(k)$

// Worker threads
1: while true do
2:     Randomly select $i \in \{1,2,\ldots,m\}$ and $j \in \mathcal{E}_i$.
3:     Read from the shared memory the components of $x$ that correspond to $\mathcal{E}_i$. Denote the read value by $\hat{x}$.
4:     Compute $\nabla_j f_i(\hat{x})$.
5:     Push $(j, \nabla_j f_i(\hat{x}))$ into the container.
6: end while
```

B. Convergence guarantees

The following theorem establishes convergence properties of Algorithm 1.

Theorem 1: Let Assumptions 1–2 hold and let ξ be any positive constant. If the step-size γ satisfies

$$\gamma \le \left(2L\left(2c\bar{\tau} + \frac{\bar{\nu}}{B}\right) + \frac{\xi\mu\bar{\tau}}{\xi+1} + \sqrt{\left(c\bar{\tau} + \frac{\bar{\nu}}{B}\right)\xi\mu\bar{\tau}}\right)^{-1} \quad (2)$$

where $c = \sqrt{\Delta}\bar{\nu}/\sqrt{\underline{\nu}}$, then for every $k \in \mathbb{N}$, we have

$$\mathbb{E}\left\{\|x(k) - x^\star\|^2\right\} \le \left(1 - \frac{\xi\mu\gamma}{\xi+1}\right)^k \|x(0) - x^\star\|^2 + e, \quad (3)$$

where

$$e = \left(\frac{2\sqrt{\Delta}\bar{\nu}\bar{\tau}}{\sqrt{\underline{\nu}}} + \frac{\bar{\nu}}{B}\right)\left(\frac{\gamma}{\mu} + \xi\bar{\tau}\gamma^2\right)\frac{2(\xi+1)}{\xi m}\sum_{i=1}^{m} \|\nabla f_i(x^\star)\|^2,$$

and $x^\star$ is the unique minimizer of (1).

Proof: See Appendix A. ■

Theorem 1 demonstrates that for a constant step-size satisfying (2), the iterates generated by Algorithm 1 will converge linearly in expectation to a ball around the optimum. The choice of the step-size will affect both the convergence factor ρ (which determines per-step decay) and the residual error e: increasing γ gives faster decay, but larger residual error. The use of mini-batching enables larger step-sizes, thus better convergence guarantees. The upper bound on the admissible

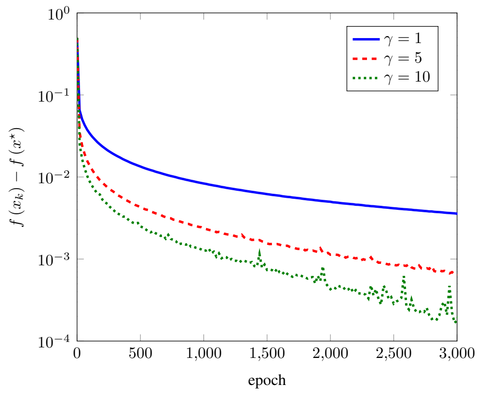

Fig. 2. The distance to the optimal values versus the number of epochs when $B = 5000$, $\bar{\tau} = 5$, $T = 4$, and $\gamma = 1, 5, 10$.

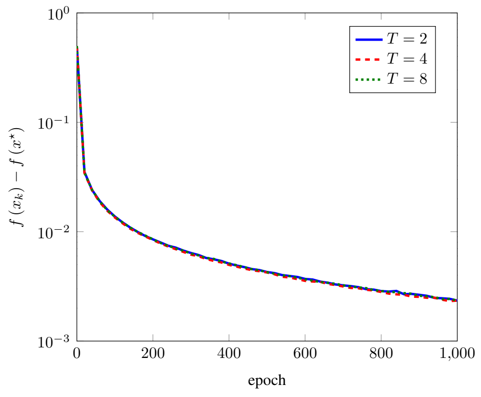

Fig. 3. The distance to the optimal values versus the number of epochs when $B = 5000$, $\bar{\tau} = 5$, and $\gamma = 5$.

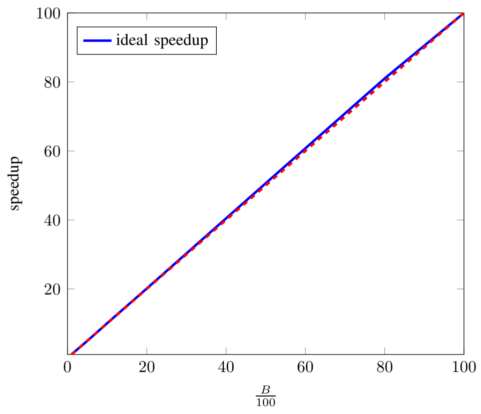

Fig. 4. Speedup of mini-batching for the lock-free algorithm when $T = 4$ and $\bar{\tau} = 5$.

step-size is of $\mathcal{O}(1/\bar{\tau})$ and which indicates that both the decay and the residual error deteriorates as $\bar{\tau}$ increases.

It is instructive to compare our results with the state-of-the art for related algorithms. If we ignore delays (i.e. $\bar{\tau} = 0$) and data sparsity (i.e. $\bar{\nu} = d$), we recover the results derived in [5, Theorem 5.5.3] for the synchronous incremental coordinate descent. We can also compare our result with [6], which makes a restrictive assumption of bounded gradient. Specifically, if we let $\bar{\tau} = 0$ and $B = \bar{\nu}$, then we obtain the convergence factors $\rho = 1 - \mu/2L$ and $\rho = 1 - u/L$, and the error terms of $\frac{C_1^2}{L\mu}$ and $\frac{C_2^2}{4L\mu}$ for Theorem 1 and [6], respectively. Here, $C_1^2 = \frac{1}{m}\sum_{i=1}^{m} \|\nabla f_i(x^\star)\|^2$, and C_2^2 is a constant such that $\|\nabla f_i(x)\| \leq C_2$, $\forall i$ and $x \in \mathbb{R}^d$. Note that, C_1 can be much smaller than C_2.

In machine learning applications, the strong convex modulus μ often arises due to the use of a strongly convex regularizer, such as the squared ℓ_2-norm, and is typically of the order $1/m$, and hence the denominator in (2) is dominated by the first term. Therefore, as long as the maximum delay is bounded by $O\left(\sqrt{\bar{\nu}}/\sqrt{\Delta}\right)$, which may be as large as $O(\sqrt{m})$, the effects of asynchrony are asymptotically negligible in both convergence factor and the error term. Theorem 1 also implies that when γ is chosen according to (2), the remaining error will never increase as $\bar{\tau}$ increases.

Finally, we remark that there exist algorithms that have *delay insensitive* convergence (see, e.g., [7]), in the sense that both the error term and step-size are irregardless of $\bar{\tau}$. However, delay does affect the convergence factor and can result in an arbitrary slow convergence as $\bar{\tau}$ grows large.

IV. Experimental Results

In this section, we perform numerical experiments to validate the efficacy of the proposed algorithm. Specifically, we consider an ℓ_2-regularized least-squares problem:

$$\min_x \frac{1}{2m}\|Ax - b\|^2 + \frac{\lambda}{2}\|x\|^2,$$

where A and b are constructed from the real data set RCV1 containing 20242 samples and 47236 features with nearly 1.5×10^6 non-zero entries and $\Delta \approx 0.42$.[1] The value of λ is set to $1/m$. In order to preserve sparsity of the updates, we re-write

$$\frac{\lambda}{2}\|x\|^2 = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}x^T M_i x$$

where $M_i x$ has the same sparsity pattern as the ith row of A. Our experiments are implemented in C++ and sparse vector and matrix operations are handled by the library Eigen[2]. All experiments are conducted on a 32-core Intel Xenon machine with 128 Gigabytes of RAM.

Figure 2 shows the iterate convergence of the proposed algorithm versus the number of epochs for different values of γ. One epoch is the expected number of iterations for the algorithm to use all data once, and is in our case equivalent to the number of non-zero components of A divided by the mini-batch size B. We can see that the iterates with larger γ converge faster to its error ball, however, they also fluctuate sooner with larger magnitude as predicted in Theorem 1. Fig. 3 shows the iterate convergence as a function of T. As can be seen, regardless of the number of worker threads, the

[1] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

[2] `http://eigen.tuxfamily.org/`

convergence behavior is nearly identical. This is consistent with the analytical result where we have shown that as long as the delay is below a certain value, the asynchrony does not affect the convergence rate of the algorithm. Finally, the effect of mini-batching is assessed in Fig. 4, where the gain in terms of the number of iterations required to reach a suboptimality of 10^{-4} relative to the case $B = 100$ is plotted. We can see that mini-batching achieves a near-linear reduction in the number of iterations.

V. Conclusion

We proposed and analyzed a lock-free incremental coordinate algorithm for minimizing a finite-sum of convex functions. By exploiting data sparsity and using mini-batching, we shown a significant improvement both in theory and practice. The results suggest that under mild conditions, the noise due to asynchronous parallelism is small compared to the noise arising from stochastic approximation, thus allows faster parallel and asynchronous solution methods.

Appendix

In this section, we prove the main result of the paper. Before proceeding to the proof, we first introduce two key lemmas which are very useful in our argument. The first Lemma is a slight extension of [8] which allows to account also for the residual error.

Lemma 1: Let $\{V(t)\}$ and $\{w(t)\}$ be sequences of non-negative real numbers satisfying

$$V(t+1) \leq \rho V(t) - \alpha w(t) + \beta \sum_{\ell=[t-\tau(t)]_+}^{t-1} w(\ell) + e,\ t \in \mathbb{N},$$

for some non-negative constants α, β, and $\rho \in (0,1)$. If $\tau(t)$ for $t \in \mathbb{N}$ satisfies

$$0 \leq \tau(t) \leq \bar{\tau} \quad \text{and} \quad \frac{\beta}{1-\rho}\frac{1-\rho^{\bar{\tau}}}{\rho^{\bar{\tau}}} \leq \alpha,$$

then

$$V(t) \leq \rho^t V(0) + \frac{1}{1-\rho} e, \forall t \in \mathbb{N}.$$

To simplify notation, we introduce

$$g(x,j,i) = |\mathcal{E}_i| \, \mathrm{R}_j \nabla f_i(x).$$

and re-write the iterations of Algorithm 1 as

$$x(k+1) = x(k) - \frac{\gamma}{B}\sum_{b=1}^{B} g(\hat{x}_b(k), j_b(k), i_b(k)). \tag{4}$$

Similarly as in [3], we note that

$$\hat{x}_b(k) - x(k) = \frac{\gamma}{B}\sum_{\ell=[k-\tau(k)]_+}^{k-1} \mathsf{P}_\ell \sum_{b=1}^{B} g(\hat{x}_b(\ell), j_b(\ell), i_b(\ell)), \tag{5}$$

where P_ℓ is a diagonal matrix whose entries are in $\{0,1\}$. These matrices account for any possible pattern of (partial) updates that can occur while $\hat{x}_b(k)$ is being processed.

Lemma 2: For any $k, \ell \in \mathbb{N}$ and $k \neq \ell$, it holds that

$$\begin{aligned} &\mathbb{E}\{\langle g(\hat{x}(\ell), j(\ell), i(\ell)), g(\hat{x}(k), j(k), i(k))\rangle\} \\ &\leq \frac{\sqrt{\Delta}}{2\sqrt{\nu}}\Big(\mathbb{E}\Big\{\|g(\hat{x}(\ell), j(\ell), i(\ell))\|^2\Big\} \\ &\quad + \mathbb{E}\Big\{\|g(\hat{x}(k), j(k), i(k))\|^2\Big\}\Big). \end{aligned}$$

The proof of Lemma 2 is omitted for brevity, but can be found in [9]. Note that the sparsity of the vectors involved allows to decrase the bound by a factor of roughly $1/\sqrt{\nu}$ compared to similar bounds derived in, *e.g.*, [10].

A. Proof of Theorem 1

Proof: We follow a similar flow as in [9], from (4), it follows that

$$\begin{aligned} \mathbb{E}\Big\{\|x(k+1) - x^\star\|^2\Big\} &= \mathbb{E}\Big\{\|x(k) - x^\star\|^2\Big\} \\ &- \frac{2\gamma}{B}\sum_{b=1}^{B}\overbrace{\mathbb{E}\{\langle \hat{x}_b(k) - x^\star, g(\hat{x}_b(k), j_b(k), i_b(k))\rangle\}}^{T_1} \\ &+ \frac{2\gamma}{B}\sum_{b=1}^{B}\underbrace{\mathbb{E}\{\langle \hat{x}_b(k) - x(k), g(\hat{x}_b(k), j_b(k), i_b(k))\rangle\}}_{T_3} \\ &+ \underbrace{\frac{\gamma^2}{B^2}\mathbb{E}\left\{\left\|\sum_{b=1}^{B} g(\hat{x}_b(k), j_b(k), i_b(k))\right\|^2\right\}}_{T_2}. \end{aligned} \tag{6}$$

First, for the term T_1, we have

$$\begin{aligned} T_1 &= \mathbb{E}\left\{\left\langle \hat{x}_b(k) - x^\star, \sum_{\ell \in \mathcal{E}_{i_b(k)}} \mathrm{R}_\ell \nabla f_{i_b(k)}(\hat{x}_b(k))\right\rangle\right\} \\ &\overset{\text{(a)}}{=} \mathbb{E}\left\{\left\langle \hat{x}_b(k) - x^\star, \nabla f_{i_b(k)}(\hat{x}_b(k))\right\rangle\right\} \\ &\overset{\text{(b)}}{=} \mathbb{E}\{\langle \hat{x}_b(k) - x^\star, \nabla f(\hat{x}_b(k))\rangle\} \\ &\overset{\text{(c)}}{\geq} \mathbb{E}\{f(\hat{x}_b(k)) - f(x^\star)\} + \frac{\mu}{2}\mathbb{E}\Big\{\|(\hat{x}_b(k) - x^\star)\|^2\Big\} \\ &\overset{\text{(d)}}{\geq} \mathbb{E}\{f(\hat{x}_b(k)) - f(x^\star)\} + \frac{\mu}{2}\frac{\xi}{\xi+1}\mathbb{E}\Big\{\|x(k) - x^\star\|^2\Big\} \\ &\quad - \frac{\xi\mu}{2}\mathbb{E}\Big\{\|\hat{x}_b(k) - x(k)\|^2\Big\}, \end{aligned} \tag{7}$$

where (a) follows by taking the expectation over the coordinates given all other variables; (b) follows due to the fact that $\hat{x}_b(k)$ is independent of $i_b(k)$; (c) follows from the strong convexity of f; and (d) follows from the inequality $\|a+b\|^2 \leq \left(1+\frac{1}{\xi}\right)\|a\|^2 + (1+\xi)\|b\|^2$ with $a = \hat{x}_b(k) - x^\star$, $b = \hat{x}_b(k) - x(k)$, and $\xi > 0$.

We next bound the term T_2. Note that $g(\cdot,\cdot,\cdot)$ is a vector that has only one non-zero element. Thus,

$$\begin{aligned}
T2 &= \sum_{b=1}^{B} \mathbb{E}\left\{\left\| g\left(\hat{x}_b(k), j_b(k), i_b(k)\right)\right\|^2\right\} \\
&= \sum_{b=1}^{B} \mathbb{E}\Bigg\{ \left|\mathcal{E}_{i_b(k)}\right| \sum_{\ell \in \mathcal{E}_{i_b(k)}} \left\langle \nabla f_{i_b(k)}\left(\hat{x}_b(k)\right), \right. \\
&\qquad\qquad \left. \mathrm{R}_\ell \nabla f_{i_b(k)}\left(\hat{x}_b(k)\right)\right\rangle \Bigg\} \\
&\le \bar{\nu} \sum_{b=1}^{B} \mathbb{E}\left\{\left\|\nabla f_{i_b(k)}\left(\hat{x}_b(k)\right)\right\|^2\right\}, \qquad (8)
\end{aligned}$$

where the third equality follows by taking the conditional expectation over the coordinates given other variables and the fact that $\mathrm{R}_\ell^2 = \mathrm{R}_\ell$.

We are now ready to bound the term T_3. Due to the limited space, from now on, we slightly abuse the notation $g(\cdot,\cdot,\cdot)$ by writing it in the first argument only. From (5), we have

$$\begin{aligned}
T_3 &= \frac{\gamma}{B} \mathbb{E}\left\{ \sum_{\ell=[k-\tau(k)]_+}^{k-1} \sum_{b'=1}^{B} \left\langle \mathsf{P}_\ell\, g\left(\hat{x}_{b'}(\ell)\right), g\left(\hat{x}_b(k)\right)\right\rangle \right\} \\
&\overset{(a)}{\le} \frac{\gamma}{B} \sum_{\ell=[k-\tau(k)]_+}^{k-1} \sum_{b'=1}^{B} \mathbb{E}\left\{\left|\left\langle g\left(\hat{x}_{b'}(\ell)\right), g\left(\hat{x}_b(k)\right)\right\rangle\right|\right\} \\
&\overset{(b)}{\le} \frac{\sqrt{\Delta}\gamma}{2\sqrt{\underline{\nu}}B} \sum_{\ell=[k-\tau(k)]_+}^{k-1} \sum_{b'=1}^{B} \Big(\mathbb{E}\left\{\left\|g\left(\hat{x}_{b'}(\ell)\right)\right\|^2\right\} \\
&\qquad\qquad + \mathbb{E}\left\{\left\|g\left(\hat{x}_b(k)\right)\right\|^2\right\}\Big) \\
&\le \frac{\sqrt{\Delta}\bar{\tau}\gamma}{2\sqrt{\underline{\nu}}} \mathbb{E}\left\{\left\|g\left(\hat{x}_b(k)\right)\right\|^2\right\} \\
&\qquad + \frac{\sqrt{\Delta}\gamma}{2\sqrt{\underline{\nu}}B} \sum_{\ell=[k-\tau(k)]_+}^{k-1} \sum_{b'=1}^{B} \mathbb{E}\left\{\left\|g\left(\hat{x}_{b'}(\ell)\right)\right\|^2\right\} \\
&\overset{(c)}{\le} \frac{\sqrt{\Delta}\bar{\nu}\bar{\tau}\gamma}{2\sqrt{\underline{\nu}}} \mathbb{E}\left\{\left\|\nabla f_{i_b(k)}\left(\hat{x}_b(k)\right)\right\|^2\right\} \\
&+ \frac{\sqrt{\Delta}\bar{\nu}\gamma}{2\sqrt{\underline{\nu}}B} \sum_{\ell=[k-\tau(k)]_+}^{k-1} \sum_{b'=1}^{B} \mathbb{E}\left\{\left\|\nabla f_{i_{b'}(\ell)}\left(\hat{x}_{b'}(\ell)\right)\right\|^2\right\}, \qquad (9)
\end{aligned}$$

where (a) follows since P_ℓ is a diagonal matrix with diagonal elements taking values in $\{0,1\}$; (b) follows by Lemma 2; and (c) follows from (8). Now, combining (6)—(9) yields

$$\begin{aligned}
&\mathbb{E}\left\{\left\|x(k+1) - x^\star\right\|^2\right\} \\
&\le \left(1 - \frac{\xi\mu\gamma}{\xi+1}\right)\mathbb{E}\left\{\left\|x(k) - x^\star\right\|^2\right\} \\
&+ \frac{\xi\mu\gamma}{B} \sum_{b=1}^{B} \mathbb{E}\left\{\left\|\hat{x}_b(k) - x(k)\right\|^2\right\} \\
&+ \frac{\sqrt{\Delta}\bar{\nu}\gamma^2}{\sqrt{\underline{\nu}}B} \sum_{\ell=[k-\tau(k)]_+}^{k-1} \sum_{b=1}^{B} \mathbb{E}\left\{\left\|\nabla f_{i_b(\ell)}\left(\hat{x}_b(\ell)\right)\right\|^2\right\} \\
&+ \left(\frac{\sqrt{\Delta}\bar{\nu}\bar{\tau}\gamma^2}{\sqrt{\underline{\nu}}B} + \frac{\bar{\nu}\gamma^2}{B^2}\right) \sum_{b=1}^{B} \mathbb{E}\left\{\left\|\nabla f_{i_b(k)}\left(\hat{x}_b(k)\right)\right\|^2\right\} \\
&- \frac{2\gamma}{B} \sum_{b=1}^{B} \mathbb{E}\left\{f\left(\hat{x}_b(k)\right) - f\left(x^\star\right)\right\}. \qquad (10)
\end{aligned}$$

We now pay attention to the term $\mathbb{E}\left\{\left\|\hat{x}_b(k) - x(k)\right\|^2\right\}$ in (10), which from (5), can be written as

$$\begin{aligned}
&\mathbb{E}\left\{\left\|\hat{x}_b(k) - x(k)\right\|^2\right\} \\
&= \frac{\gamma^2}{B^2} \mathbb{E}\left\{\left\| \sum_{\ell=[k-\tau(k)]_+}^{k-1} \mathsf{P}_\ell \sum_{b=1}^{B} g\left(\hat{x}_b(\ell)\right)\right\|^2\right\} \\
&\le \frac{\gamma^2}{B^2} \sum_{\ell=[k-\tau(k)]_+}^{k-1} \mathbb{E}\left\{\left\|\mathsf{P}_\ell \sum_{b=1}^{B} g\left(\hat{x}_b(\ell)\right)\right\|^2\right\} \\
&+ \frac{\gamma^2}{B^2} \sum_{\substack{\ell=[k-\tau(k)]_+ \\ \ell' \neq \ell}}^{k-1} \mathbb{E}\Bigg\{\Bigg|\Bigg\langle \mathsf{P}_\ell \sum_{b=1}^{B} g\left(\hat{x}_b(\ell)\right), \\
&\qquad\qquad \mathsf{P}_{\ell'} \sum_{b'=1}^{B} g\left(\hat{x}_{b'}(\ell')\right)\Bigg\rangle\Bigg|\Bigg\}. \qquad (11)
\end{aligned}$$

Let S_1 and S_2 be the first and second terms in (11), respectively. Then, S_1 can be bounded as

$$\begin{aligned}
S_1 &\le \frac{\gamma^2}{B^2} \sum_{\ell=[k-\tau(k)]_+}^{k-1} \mathbb{E}\left\{\left\|\sum_{b=1}^{B} g\left(\hat{x}_b(\ell)\right)\right\|^2\right\} \\
&= \frac{\gamma^2}{B^2} \sum_{\ell=[k-\tau(k)]_+}^{k-1} \sum_{b=1}^{B} \mathbb{E}\left\{\left\|g\left(\hat{x}_b(\ell)\right)\right\|^2\right\}.
\end{aligned}$$

For S_2, we have

$$S_2 \leq \frac{\gamma^2}{B^2} \sum_{\substack{\ell=[k-\tau(k)]_+ \\ \ell' \neq \ell}}^{k-1} \sum_{b=1}^{B} \sum_{b'=1}^{B} \mathbb{E}\left\{\left|\left\langle g\left(\hat{x}_b(\ell)\right), g\left(\hat{x}_{b'}(\ell')\right)\right\rangle\right|\right\}$$
$$\leq \frac{\sqrt{\Delta}\gamma^2}{2\sqrt{\underline{\nu}}B^2} \sum_{\substack{\ell=[k-\tau(k)]_+ \\ \ell' \neq \ell}}^{k-1} \sum_{b=1}^{B} \sum_{b'=1}^{B} \Big(\mathbb{E}\left\{\left\|g\left(\hat{x}_b(\ell)\right)\right\|^2\right\} + \mathbb{E}\left\{\left\|g\left(\hat{x}_{b'}(\ell')\right)\right\|^2\right\}\Big)$$
$$\leq \frac{\sqrt{\Delta}\bar{\tau}\gamma^2}{\sqrt{\underline{\nu}}B} \sum_{\ell=[k-\tau(k)]_+}^{k-1} \sum_{b=1}^{B} \Big(\mathbb{E}\left\{\left\|g\left(\hat{x}_b(\ell)\right)\right\|^2\right\}\Big),$$

where the third inequality follows from Lemma 2, and the last inequality follows from a simple counting argument. Thus, we obtain

$$\mathbb{E}\left\{\left\|\hat{x}_b(k) - x(k)\right\|^2\right\} \leq \left(\frac{\bar{\nu}\gamma^2}{B^2} + \frac{\sqrt{\Delta}\bar{\nu}\bar{\tau}\gamma^2}{\sqrt{\underline{\nu}}B}\right) \times \sum_{\ell=[k-\tau(k)]_+}^{k-1} \sum_{b=1}^{B} \mathbb{E}\left\{\left\|\nabla f_{i_b(\ell)}\left(\hat{x}_b(\ell)\right)\right\|^2\right\}. \quad (12)$$

Furthermore, the last term in (10) can be bounded by noting that [11, Lemma 6.4]

$$f\left(\hat{x}_b(k)\right) - f\left(x^\star\right) \geq \frac{1}{2L} \mathbb{E}\left\{\left\|\nabla f_{i_b(k)}\left(\hat{x}_b(k)\right) - \nabla f_{i_b(k)}\left(x^\star\right)\right\|^2\right\} \quad (13)$$

By substituting (12) and (13) into (10), we arrive at

$$\mathbb{E}\left\{\left\|x(k+1) - x^\star\right\|^2\right\} \leq \left(1 - \frac{\xi\mu\gamma}{\xi+1}\right) \mathbb{E}\left\{\left\|x(k) - x^\star\right\|^2\right\}$$
$$+ \left(\frac{\sqrt{\Delta}\bar{\nu}\bar{\tau}}{\sqrt{\underline{\nu}}B} + \frac{\bar{\nu}}{B^2}\right)\gamma^2 \sum_{b=1}^{B} \mathbb{E}\left\{\left\|\nabla f_{i_b(k)}\left(\hat{x}_b(k)\right)\right\|^2\right\}$$
$$+ \left(\frac{\sqrt{\Delta}\bar{\nu}}{\sqrt{\underline{\nu}}B}\gamma^2 + \mu\xi\left(\frac{\sqrt{\Delta}\bar{\nu}\bar{\tau}}{\sqrt{\underline{\nu}}B} + \frac{\bar{\nu}}{B^2}\right)\gamma^3\right)$$
$$\times \sum_{\ell=[k-\tau(k)]_+}^{k-1} \sum_{b=1}^{B} \mathbb{E}\left\{\left\|\nabla f_{i_b(\ell)}\left(\hat{x}_b(\ell)\right)\right\|^2\right\}$$
$$- \frac{\gamma}{LB} \sum_{b=1}^{B} \mathbb{E}\left\{\left\|\nabla f_{i_b(k)}\left(\hat{x}_b(k)\right) - \nabla f_{i_b(k)}\left(x^\star\right)\right\|^2\right\}.$$

Using the fact that for $i \in \{1, \ldots, m\}$

$$\left\|\nabla f_i(x)\right\|^2 \leq 2\left\|\nabla f_i(x) - \nabla f_i\left(x^\star\right)\right\|^2 + 2\left\|\nabla f_i\left(x^\star\right)\right\|^2,$$

we can further bound the expression above as

$$\mathbb{E}\left\{\left\|x(k+1) - x^\star\right\|^2\right\} \leq \left(1 - y_1\gamma\right) \mathbb{E}\left\{\left\|x(k) - x^\star\right\|^2\right\}$$
$$- \left(\gamma - y_2\gamma^2\right) \frac{1}{LB} \sum_{b=1}^{B} \mathbb{E}\left\{\left\|\nabla f_{i_b(k)}\left(\hat{x}_b(k)\right) - \nabla f_{i_b(k)}\left(x^\star\right)\right\|^2\right\}$$
$$+ \left(y_4\gamma^2 + y_3\gamma^3\right)$$
$$\times \frac{1}{LB} \sum_{\ell=[k-\tau(k)]_+}^{k-1} \sum_{b=1}^{B} \mathbb{E}\left\{\left\|\nabla f_{i_b(\ell)}\left(\hat{x}_b(\ell)\right) - \nabla f_{i_b(\ell)}\left(x^\star\right)\right\|^2\right\}$$
$$+ \frac{\left(y_2 + \bar{\tau}y_4\right)\gamma^2 + \bar{\tau}y_3\gamma^3}{L} \mathbb{E}\left\{\left\|\nabla f_i\left(x^\star\right)\right\|^2\right\},$$

where $y_1 = \frac{\xi\mu}{\xi+1}$, $y_2 = 2L\left(\frac{\sqrt{\Delta}\bar{\nu}\bar{\tau}}{\sqrt{\underline{\nu}}} + \frac{\bar{\nu}}{B}\right)$, $y_3 = \xi\mu y_2$, and $y_4 = \frac{2L\sqrt{\Delta}\bar{\nu}}{\sqrt{\underline{\nu}}}$. Then, by invoking Lemma 1 with $V(k) = \mathbb{E}\left\{\left\|x(k) - x^\star\right\|^2\right\}$; $w(k) = \frac{1}{LB}\sum_{b=1}^{B}\mathbb{E}\left\{\left\|\nabla f_{i_b(k)}\left(\hat{x}_b(k)\right) - \nabla f_{i_b(k)}\left(x^\star\right)\right\|^2\right\}$; $\rho = 1 - y_1\gamma$; $\alpha = -y_2\gamma^2 + \gamma$; $\beta = y_3\gamma^3 + y_4\gamma^2$; and $e = \frac{(y_2+\bar{\tau}y_4)\gamma^2+\bar{\tau}y_3\gamma^3}{L}\mathbb{E}\left\{\left\|\nabla f_i\left(x^\star\right)\right\|^2\right\}$, it follows that the sequence $\{x(k)\}$ generated by Algorithm 1 satisfies (3) if the following conditions are fulfilled:

$$\gamma y_2 \leq 1 \quad (14)$$
$$\frac{1}{\left(1 - y_1\gamma\right)^{\bar{\tau}}} - 1 \leq y_1 \frac{-y_2\gamma + 1}{y_3\gamma + y_4}, \quad (15)$$

where the first condition comes from the positivity of α and the second one is the result of the general convergence condition. By Bernoulli's inequality, we have that $\left(1 - y_1\gamma\right)^{\bar{\tau}} \geq 1 - y_1\gamma\bar{\tau}$. Hence, if $y_1\bar{\tau}\gamma < 1$, the LHS of (15) can be further upper-bounded by

$$\frac{1}{\left(1 - y_1\gamma\right)^{\bar{\tau}}} - 1 \leq \frac{y_1\bar{\tau}\gamma}{1 - y_1\bar{\tau}\gamma}. \quad (16)$$

Therefore, if the step-size γ is chosen such that

$$\frac{\bar{\tau}\gamma}{1 - y_1\bar{\tau}\gamma} \leq \frac{-y_2\gamma + 1}{y_3\gamma + y_4},$$

then the condition (15) will be automatically satisfied. That is, we want to seek the step-size satisfying

$$\left(y_3 - y_2y_1\right)\bar{\tau}\gamma^2 + \left(\bar{\tau}y_4 + y_2 + y_1\bar{\tau}\right)\gamma - 1 \leq 0. \quad (17)$$

We observe that $x = \frac{1}{1+a}$ satisfies the inequality $a^2x^2 + x - 1 \leq 0$ if a is nonegative. Then, by applying the result above for (17) with $x = \left(\bar{\tau}y_4 + y_2 + y_1\bar{\tau}\right)\gamma$, we can choose the step-size γ as follows to guarantee the condition (15):

$$\gamma \leq \frac{1}{\bar{\tau}y_4 + y_2 + y_1\bar{\tau} + \sqrt{\left(y_3 - y_2y_1\right)\bar{\tau}}}.$$

It is obvious that this step-size also satisfies the condition (14) and $y_1\bar{\tau}\gamma < 1$, which completes the proof. ■

References

[1] F. Niu, B. Recht, C. Ré, and S. J. Wright, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Proc. NIPS'16*, Granada, Spains, Dec 2011, pp. 693–701.

[2] J. Liu, S. J. Wright, C. Ré, V. Bittorf, and S. Sridhar, "An asynchronous parallel stochastic coordinate descent algorithm," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 285–322, Feb. 2015.

[3] H. Mania, X. Pan, D. Papailiopoulos, B. Recht, K. Ramchandran, and M. I. Jordan, "Perturbed iterate analysis for asynchronous stochastic optimization," Mar. 2016. [Online]. Available: http://arxiv.org/abs/1507.06970

[4] J. Liu, S. J. Wright, and S. Sridhar, "An asynchronous parallel randomized Kaczmarz algorithm," Jun. 2014. [Online]. Available: https://arxiv.org/abs/1401.4780

[5] S. Khirirat, "Randomized first-order methods for convex optimization," Master's thesis, KTH Royal Institute of Technology, Stockholm, 2016.

[6] M. Schmidt, "Convergence rate of stochastic gradient with constant step size," University of British Columbia, Tech. Rep., May 2014.

[7] A. Aytekin, H. R. Feyzmahdavian, and M. Johansson, "Asynchronous incremental block-coordinate descent," in *Proc. 52th Annual Allerton Conference on Communication, Control, and Computing*, IL, USA, 2014, pp. 19–24.

[8] ——, "Analysis and implementation of an asynchronous optimization algorithm for the parameter server," Oct. 2016. [Online]. Available: https://arxiv.org/pdf/1610.05507.pdf

[9] V. V. Mai and M. Johansson, "An asynchronous mini-batch randomized Kaczmarz algorithm," submitted for possible publication.

[10] R. Leblond, F. Pedregosa, and S. Lacoste-Julien, "Asaga: Asynchronous parallel saga," Jun. 2016. [Online]. Available: https://arxiv.org/abs/1606.04809

[11] S. Bubeck, "Convex optimization: Algorithms and complexity," *Foundations and Trends in Machine Learning*, vol. 8, no. 3-4, pp. 231–357, 2015.