Fifty-second Annual Allerton Conference
Allerton House, UIUC, Illinois, USA
October 1 - 3, 2014

Asynchronous Incremental Block-Coordinate Descent

Arda Aytekin Hamid Reza Feyzmahdavian Mikael Johansson

Abstract— **This paper studies a flexible algorithm for minimizing a sum of component functions, each of which depends on a large number of decision variables. Such formulations appear naturally in "big data" applications, where each function describes the loss estimated using the data available at a specific machine, and the number of features under consideration is huge. In our algorithm, a coordinator updates a global iterate based on delayed partial gradients of the individual objective functions with respect to blocks of coordinates. Delayed incremental gradient and delayed coordinate descent algorithms are obtained as special cases. Under the assumption of strong convexity and block coordinate-wise Lipschitz continuous partial gradients, we show that the algorithm converges linearly to a ball around the optimal value. Contrary to related proposals in the literature, our algorithm is delay-insensitive: it converges for any bounded information delay, and its step-size parameter can be chosen independently of the maximum delay bound.**

I. Introduction

The increased availability of abundant data has created a strong interest in machine learning in hope of inferring useful information. To deal with the large feature vectors and immense data sets typically involved, it becomes instrumental to develop algorithms that parallelize well.

While parallel optimization has a rich history [1], [2], it is important to tailor algorithms to make the best use of current computer architectures. Today, the architectural trends are desktops equipped with powerful multi-core processors and general purpose graphics processing units (GPGPUs), on the one hand, and commoditized distributed computing services such as Amazon's Elastic Compute Cloud [3], on the other. These architectures certainly differ in scale, but they also tend to differ in their use of shared and distributed memory.

In a shared memory architecture, all the computational nodes have access to the same memory, and they work simultaneously to achieve a desired task. However, if the task involves coupling among different parts of the memory, the computational nodes need to synchronize to update the coupled parts of the memory and to access the most up-to-date data. Such synchronization tends to degrade the parallel performance of a given algorithm dramatically. Several authors have proposed solutions that avoid this synchronization. In [4], an asynchronous incremental sub-gradient method has been studied in which gradient steps are taken using out-of-date gradients. Niu et al. have studied a lock-free approach to parallelizing the stochastic gradient descent method [5]. Their code, called HOGWILD!, uses atomic operations to avoid locking of loosely coupled memory locations in the minimization problem of sparse separable cost functions, and have achieved linear speedup in the number of processors. Following a similar sparsity and separability assumption, Fercoq and Richtárik have proposed an accelerated, parallel and proximal coordinate descent method to better utilize the available processors to achieve even further speedups [6].

In a distributed memory architecture, however, each computational node has immediate access to its local memory, whereas some global information needs to be coordinated among nodes. Such an architecture is inherent in cloud computing systems, but also in desktop computers utilizing both CPUs and GPGPUs. Moreover, in cloud computing systems, nodes may be placed in different geographical locations, which may lead to significant delays and failures in nodes. These issues must be taken into account in designing optimization algorithms. Li et al. have proposed the Parameter Server framework to overcome these problems [7]. In Parameter Server, optimization problems involving huge parameter vectors and big training data sets are distributed among different general purpose computers in a server-client setting. Servers can communicate among each other and have access to globally shared data, and they utilize different client computers to solve parts of the overall problem. Since different servers can utilize the same client for different purposes, each client is assigned a set of training data (set of functions) and a copy of the parameter vector. To avoid communication overhead, servers push the most up-to-date parameter vector to clients only when needed. Hence, clients have delayed copies of the most recent parameter vector, and they do their calculations based on these outdated values. Whenever a client finishes the assigned work, it sends the result to the calling server which updates the parameter vector and synchronizes with the other servers.

In this paper, we focus on this latter architecture, and propose and analyze an algorithm to minimize a sum of loss functions over a (possibly huge and not necessarily sparse) parameter vector using an incremental block-coordinate descent method with delayed parameter information.

The paper is organized as follows. We first introduce the notation and some necessary background used throughout the paper. In Section II, we describe and analyze our asynchronous incremental block-coordinate algorithm. Section III presents a comparison between this work and our previous work on asynchronous incremental gradient descent algorithms, while numerical evaluations are reported in Section IV. Finally, conclusions are stated in Section V.

A. Aytekin, H. R. Feyzmahdavian 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: {aytekin, hamidrez, mikaelj}@kth.se.

978-1-4799-8009-3/14/$31.00 ©2014 IEEE

A. Notation and Preliminaries

Here, we introduce the notation and review the key definitions that will be used throughout the paper. We let $\mathbb{R}$, $\mathbb{N}$, and $\mathbb{N}_0$ denote the set of real numbers, the set of natural numbers, and the set of natural numbers including zero, respectively. The Euclidean norm is denoted by $\|\cdot\|$.

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 $x, y \in \mathbb{R}^n$. The gradient of f is *L-Lipschitz continuous* on $\mathbb{R}^n$ if there exists a positive constant L such that

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

For any partition of $x \in \mathbb{R}^n$ into B non-overlapping blocks, $(x_1, \ldots, x_B)$, with $x_b \in \mathbb{R}^{n_b}$, $b = 1, \ldots, B$, and

$$\sum_{b=1}^{B} n_b = n,$$

we define $U_b \in \mathbb{R}^{n \times n_b}$ as the corresponding partition of the identity matrix:

$$I_n = (U_1, U_2, \ldots, U_B) \in \mathbb{R}^{n \times n}.$$

Then, the *partial gradient* of $f : \mathbb{R}^n \to \mathbb{R}$ with respect to $x_b \in \mathbb{R}^{n_b}$ is defined as

$$\nabla_b f(x) = U_b^\top \nabla f(x).$$

The gradient of f is called *block coordinate-wise Lipschitz continuous* on $\mathbb{R}^n$ if there exist constants $l_1, \ldots, l_B$ such that

$$\|\nabla_b f(x + U_b h_b) - \nabla_b f(x)\| \leq l_b \|h_b\|$$

holds for all $x \in \mathbb{R}^n$, $b = 1, \ldots, B$, and $h_b \in \mathbb{R}^{n_b}$. An important consequence of this assumption is that

$$f(x + U_b h_b) \leq f(x) + \langle \nabla_b f(x), h_b \rangle + \frac{l_b}{2}\|h_b\|^2 \quad (1)$$

holds for all $x \in \mathbb{R}^n$ and all $h_b \in \mathbb{R}^{n_b}$ [8].

II. An Asynchronous Incremental Block-Coordinate Descent Algorithm

We consider the optimization problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \quad (2)$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is of the form

$$f(x) = \sum_{m=1}^{M} f_m(x),$$

and satisfies the following assumptions:

A1) Each f_m has L_m-Lipschitz continuous gradient,

A2) The gradient of f is block coordinate-wise Lipschitz continuous with constants $l_1, \ldots, l_B$,

A3) The function f is μ-strongly convex.

Algorithm 1 Asynchronous Incremental Coordinate Descent

Inputs: $L_1, \ldots, L_m$; $l_1, \ldots, l_B$; $\theta \in (0, 1]$; α.
Initialization: $x(0) \in \mathbb{R}^n$.
for $t \in \mathbb{N}_0$ **do**
  Pick $i(t) \in \{1, \ldots, M\}$ with probability

$$p\{i(t) = m\} = \frac{L_m}{\sum_{m=1}^{M} L_m}$$

  Choose $j(t) \in \{1, \ldots, B\}$ with probability

$$p\{j(t) = b\} = \frac{l_b}{\sum_{b=1}^{B} l_b}$$

  $s(t) = x(t - \tau(t)) - \frac{\alpha}{L_{i(t)} l_{j(t)}} U_{j(t)} \nabla_{j(t)} f_{i(t)}(x(t - \tau(t)))$
  $x(t+1) = (1 - \theta)x(t) + \theta s(t)$
end for

Note that Assumption **A1)** guarantees that the gradient of f is also Lipschitz continuous with constant $L \leq L_{\text{sum}}$, where

$$L_{\text{sum}} = \sum_{m=1}^{M} L_m.$$

We consider a setup similar to Parameter Server: a coordinator which maintains and updates the current iterate of the decision vector x, and M worker nodes, each of which can compute the gradient of one of the component functions f_m. Coordinators and workers follow the procedure in Algorithm 1. At every iteration t_c, the coordinator picks a worker $m \in \{1, \ldots, M\}$ at random, and informs the worker about the current iterate $x(t_c)$. Then, the worker chooses a block $b \in \{1, \ldots, B\}$ at random and evaluates a partial gradient mapping

$$x(t_c) - \frac{\alpha}{L_m l_b} U_b \nabla_b f_m(x(t_c)).$$

In other words, worker m only updates block b of $x(t_c)$ by taking step of length $\alpha/(L_m l_b)$ in the direction $-\nabla_b f_m(x(t_c))$. When the worker completes the computation at some later time t, it returns the partial gradient mapping to the coordinator, which averages the update according to

$$x\big(t+1\big) = \big(1 - \theta\big)x(t) + \theta\big(x(t_c) - \frac{\alpha}{L_m l_b} U_b \nabla_b f_m(x(t_c))\big)$$

and passes the updated $x(t+1)$ back to the worker. We assume infinite delays cannot occur between reading and updating: there is a bound $\tau_{\max}$ on the time interval between the time at which one worker reads the vector x and the time at which it sends its update back to the coordinator.

After t iterations of Algorithm 1, a random vector $x(t)$ is generated, which depends on the observed implementation of random variables

$$\big\{i(0), j(0), i(1), j(1), \ldots, i(t-1), j(t-1)\big\}. \quad (3)$$

The next theorem shows that under assumptions **A1)**–**A3)**, the expected value of $f(x(t))$, over all random variables (3), converges linearly to a ball around the optimum.

Theorem 1: Assume that $\alpha \in (0, \mu)$ and that

$$0 \leq \tau(t) \leq \tau_{\max}, \quad t \in \mathbb{N}_0.$$

Then, the sequence $\{x(t)\}$ generated by Algorithm 1 satisfies

$$\mathbb{E}_{t-1}[f(x(t))] - f^\star \leq \rho^t\big(f(x(0)\big) - f^\star) + e, \qquad (4)$$

where $\mathbb{E}_{t-1}$ is the expectation over all random variables (3), $f^\star$ is the optimal value for problem (2),

$$\rho = \left(1 - 2\alpha\theta \, \frac{\mu - \alpha}{\sum_{m=1}^{M} L_m \sum_{b=1}^{B} l_b}\right)^{\frac{1}{1+\tau_{\max}}}, \qquad (5)$$

and

$$e = \frac{\alpha(2+\theta)}{2\big(2\mu - \alpha(2+\theta)\big)} \sum_{m=1}^{M} \frac{1}{L_m} \|\nabla f_m(x^\star)\|^2. \qquad (6)$$

Proof: See Appendix A. ∎

Theorem 1 establishes that the proposed algorithm has *delay insensitive* convergence, in the sense that as long as $\alpha \in (0, \mu)$, the iterates will converge in expectation to a ball around the optimum regardless of how large $\tau_{\max}$ is. However, $\tau_{\max}$ does affect the convergence factor ρ, and hence the time it takes for the iterates to converge. Specifically, ρ is monotonically increasing in $\tau_{\max}$, and approaches one as $\tau_{\max}$ tends to infinity. Therefore, while the algorithm converges linearly to a ball around the optimal value for arbitrary bounded time-varying delays, the convergence speed deteriorates with increasing delays.

We also note choosing α involves a trade-off between accuracy and convergence speed: to decrease the residual error e, one needs to decrease α; but this increases ρ, and yields slower convergence. The averaging parameter θ is subject to a similar trade-off.

III. Efficiency Comparison with Asynchronous Incremental Gradient Descent

Randomized coordinate descent has been shown to be competitive with the classical gradient descent method, in the sense that it requires less work per iteration, but a comparable number of iterations to converge [8]. In this section, we demonstrate that a similar property holds for asynchronous incremental block-coordinate descent: if the amount of work required to evaluate a partial gradient is proportional to its block size, then incremental block-coordinate descent can always be expected to be more efficient than a corresponding incremental gradient descent algorithm. We establish this property by analyzing the incremental gradient method in Algorithm 2, obtained by restricting Algorithm 1 to use $B = 1$, and comparing the total work that each method requires to guarantee a given target error. Our comparison is based on the following result.

Theorem 2: If $\alpha \in (0, \mu)$, then the sequence $\{x(t)\}$ generated by Algorithm 2 satisfies

$$\mathbb{E}_{t-1}[f(x(t))] - f^\star \leq \rho^t\big(f(x(0)\big) - f^\star) + e, \qquad (7)$$

Algorithm 2 Asynchronous Incremental Descent

1: **Inputs**: $L_1, \ldots, L_m$; $\theta \in (0, 1]$; α.
2: **Initialization:** $x(0) \in \mathbb{R}^n$.
3: **for** $t \in \mathbb{N}_0$ **do**
4: Pick $i(t) \in \{1, \ldots, M\}$ with probability

$$p\{i(t) = m\} = \frac{L_m}{\sum_{m=1}^{M} L_m}$$

5: $s(t) = x(t - \tau(t)) - \frac{\alpha}{L_{i(t)} L} \nabla f_{i(t)}(x(t - \tau(t)))$
6: $x(t+1) = (1-\theta)x(t) + \theta s(t)$
7: **end for**

where

$$\rho = \left(1 - 2\alpha\theta \, \frac{\mu - \alpha}{L \sum_{m=1}^{M} L_m}\right)^{\frac{1}{1+\tau_{\max}}}, \qquad (8)$$

and

$$e = \frac{\alpha(2+\theta)}{2\big(2\mu - \alpha(2+\theta)\big)} \sum_{m=1}^{M} \frac{1}{L_m} \|\nabla f_m(x^\star)\|^2. \qquad (9)$$

Proof: Letting $B = 1$, $l_b = L$, and $p\{j(t) = b\} = 1$, the proof is similar to that of Theorem 1, and thus, omitted. ∎

Comparing Theorems 1 and 2, we see that for any fixed $\alpha \in (0, \mu)$ and $\theta \in (0, 1]$, the algorithms have the same bound on e. Thus, partitioning the decision variables into several blocks does not affect the guaranteed residual error.

However, the two methods have different convergence factors, and will therefore require a different number of iterations to reach a guaranteed target error. To compare the two methods, we assume that the work required to evaluate a partial gradient is proportional to its block-size, and we count every B iterations of Algorithm 1 as *one* iteration of Algorithm 2. After B iterations of Algorithm 1, by (5)

$$\mathbb{E}_{t-1}[f(x(t))] - f^\star \leq \left(1 - 2\frac{\alpha\theta B}{1+\tau_{\max}} \, \frac{\mu - \alpha}{\sum_{m=1}^{M} L_m \sum_{b=1}^{B} l_b}\right)\big(f(x(0)\big) - f^\star) + e,$$

and after one iteration of Algorithm 2, from (8), we obtain

$$\mathbb{E}_{t-1}[f(x(t))] - f^\star \leq \left(1 - 2\frac{\alpha\theta}{1+\tau_{\max}} \, \frac{\mu - \alpha}{L \sum_{m=1}^{M} L_m}\right)\big(f(x(0)\big) - f^\star) + e.$$

We can see that if

$$\frac{\sum_{b=1}^{B} l_b}{L} \leq B, \qquad (10)$$

then the difference of $f^\star$ and the expectation of the function values $f(x(t))$ generated by Algorithm 1 is smaller than that by Algorithm 2. Since $l_b \leq L$ for each $b \in \{1, \ldots, B\}$, (10) always holds for all $B \in \mathbb{N}$, which implies that for any fixed

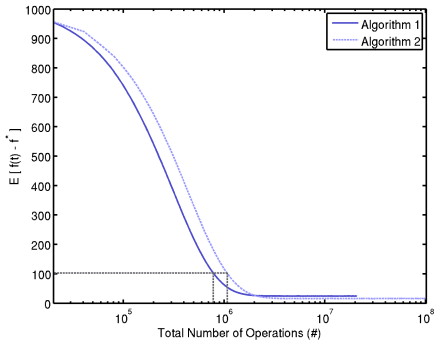

Fig. 1. Convergence of Algorithms 1 and 2 with respect to the total operations required. In the example, the targeted error to achieve is 10% of the initial error. Clearly, Algorithm 2 is computationally more intensive in achieving the same targeted error.

$\alpha \in (0, \mu)$ and any fixed $\theta \in (0, 1]$, Algorithm 1 is always more efficient than Algorithm 2 (see Figure 1).

Remark 1: Algorithm 2 and its analysis is slightly different than the delayed incremental gradient method that we presented in [9]. In the notation of this paper, our earlier algorithm would have a convergence factor on the same form as (8) but with $\sum_{m=1}^{M} L_m$ replaced by $ML_{\max}$, where

$$L_{\max} = \max_{1 \le m \le M} L_m.$$

Since $\sum_{m=1}^{M} L_m \le ML_{\max}$ [10], it follows that our guaranteed upper bound improves upon the one in [9], especially for applications where the component functions vary substantially in smoothness.

IV. Experimental Results

We have constructed a randomly generated, unconstrained quadratic programming (QP) test problem to evaluate the performance of our asynchronous incremental block-coordinate descent algorithm. The problem instance is as follows:

$$\underset{x \in \mathbb{R}^{100}}{\text{minimize}} \qquad f(x) = \sum_{m=1}^{20} \left(\frac{1}{2} x^\top Q_m x + r_m^\top x \right), \qquad (11)$$

where the matrices Q_m are positive definite with their condition numbers taking values in $[1, 5]$, and the vectors r_m are standard normal. Since we have constructed this numerical example on our local computer, we have artificially introduced random time delays, $\tau(t) \in [0, \tau_{\max}]$, into our simulation code. We have simulated our algorithm 1000 times for $\theta = \{0.2, 1\}$, $\tau_{\max} = \{1, 10\}$ and $\alpha = \{0.03\mu, 0.1\mu\}$, and present our averaged convergence results with confidence intervals of one standard deviation.

Figures 2 and 3 show how different choices of the algorithm parameters θ and α affect the convergence rate and the remaining error, whereas the effect of the magnitude of the time-varying delay is shown in Figure 4. As can be seen in Figure 2, an increase in θ results in a faster convergence, but larger residual error. A similar trade-off is also valid for α as shown in Figure 3. Finally, as the magnitude of the time-varying delay increases, the algorithm converges slower while the residual error remains unaffected (see Figure 4).

Although not so obvious due to the logarithmic y-axes in Figures 2–4, it is also worth noting that the variance of our sample iterates is sensitive to the averaging parameter, θ. This is not surprising as increasing (respectively, decreasing) θ results in an amplification (respectively, suppression) of the random, outdated information obtained from the workers.

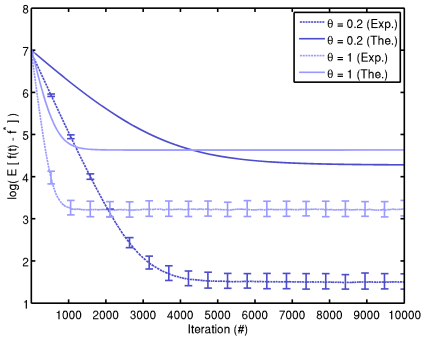

Fig. 2. Convergence for different choices of the averaging parameter, θ, when $\tau_{\max} = 1$ and $\alpha = 0.1\mu$. Solid curves represent the theoretical upper bounds on the expected error, whereas dashed curves represent the averaged experimental results for averaging (dark blue color), *e.g.* $\theta = 0.2$, and non-averaging (light blue color), *e.g.* $\theta = 1$, cases. Vertical error bars represent confidence intervals of one standard deviation.

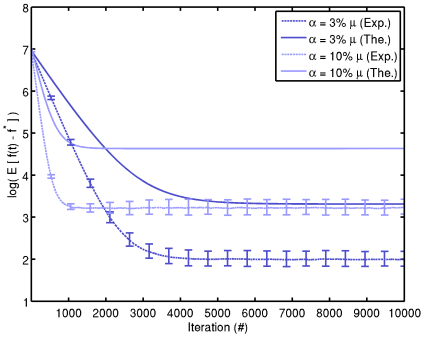

Fig. 3. Convergence for different choices of α, when $\theta = 1$ and $\tau_{\max} = 1$. Solid curves represent the theoretical upper bounds on the expected error, whereas dashed curves represent the averaged experimental results for $\alpha = 0.03\mu$ (dark blue color) and $\alpha = 0.1\mu$ (light blue color). Vertical error bars represent confidence intervals of one standard deviation.

V. Conclusions

We have proposed a new, flexible method for minimizing a sum of objective functions, each of which depends on a large number of shared decision variables. Contrary to similar work in literature, our method is delay-insensitive, in the sense that it converges linearly for any bounded, time-varying information delay, and its parameters can be chosen independently of the delay bound. Moreover, our method covers delayed incremental gradient and delayed coordinate descent algorithms as special cases. Through extensive simulations, we have also verified that our theoretical bounds hold reasonably well.

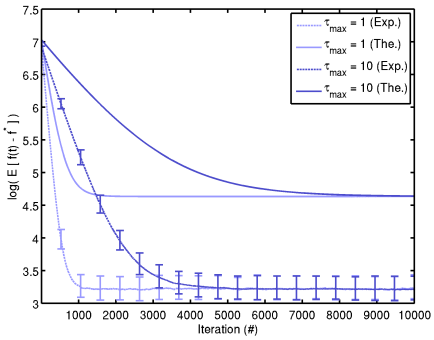

Fig. 4. Convergence for different values of maximum time delay, $\tau_{\max}$, when $\theta = 1$ and $\alpha = 0.1\mu$. Solid curves show the theoretical upper bounds on the expected error, while dashed curves represent averaged experimental results for $\tau_{\max} = 1$ (dark blue color) and $\tau_{\max} = 10$ (light blue color). Error bars represent confidence intervals of one standard deviation.

APPENDIX

Before proving Theorem 1, we state a key lemma that is instrumental in our argument.

Lemma 1 ([9]): Let $\{V(t)\}$ be a sequence of non-negative real numbers satisfying

$$V(t+1) \le pV(t) + q \max_{t-\tau(t)\le s\le t} V(s) + r, \quad t \in \mathbb{N}_0,$$

for some non-negative constants p, q, and r. If $p + q < 1$ and

$$0 \le \tau(t) \le \tau_{\max}, \quad t \in \mathbb{N}_0,$$

then

$$V(t) \le \rho^t V(0) + e, \quad t \in \mathbb{N}_0,$$

where $\rho = (p+q)^{\frac{1}{1+\tau_{\max}}}$ and $e = r/(1-p-q)$.

A. Proof of Theorem 1

Since f is convex and $\theta \in (0, 1]$, we have

$$\begin{aligned} f\big(x(t+1)\big) - f^\star &= f\big((1-\theta)x(t) + \theta s(t)\big) - f^\star \\ &\le (1-\theta) f(x(t)) + \theta f(s(t)) - f^\star \\ &= \big(1-\theta\big)\big(f(x(t)) - f^\star\big) \\ &\quad + \theta\big(f(s(t)) - f^\star\big). \end{aligned} \tag{12}$$

As the gradient of f is block coordinate-wise Lipschitz, it follows from (1) that

$$\begin{aligned} & f(s(t)) \\ &\le f(x(t-\tau(t))) \\ &\quad - \frac{\alpha}{L_{i(t)} l_{j(t)}} \langle \nabla_{j(t)} f(x(t-\tau(t))), \nabla_{j(t)} f_{i(t)}(x(t-\tau(t))) \rangle \\ &\quad + \frac{\alpha^2}{2L_{i(t)}^2 l_{j(t)}} \|\nabla_{j(t)} f_{i(t)}(x(t-\tau(t)))\|^2 \\ &\le f(x(t-\tau(t))) \\ &\quad - \frac{\alpha}{L_{i(t)} l_{j(t)}} \langle \nabla_{j(t)} f(x(t-\tau(t))), \nabla_{j(t)} f_{i(t)}(x(t-\tau(t))) \rangle \\ &\quad + \frac{(1+\xi)\alpha^2}{2L_{i(t)}^2 l_{j(t)}} \|\nabla_{j(t)} f_{i(t)}(x(t-\tau(t))) - \nabla_{j(t)} f_{i(t)}(x^\star)\|^2 \\ &\quad + \left(1 + \frac{1}{\xi}\right) \frac{\alpha^2}{2L_{i(t)}^2 l_{j(t)}} \|\nabla_{j(t)} f_{i(t)}(x^\star)\|^2, \end{aligned}$$

where the second inequality uses the fact that for any $\xi > 0$ and for any vectors x and y in $\mathbb{R}^n$, we have

$$\|x+y\|^2 \le (1+\xi)\|x\|^2 + \left(1 + \frac{1}{\xi}\right)\|y\|^2.$$

We use p_m and q_b to denote $p\{i(t) = m\}$ and $p\{j(t) = b\}$, respectively. We also use $\mathbb{E}_{t|t-1}$ to denote the conditional expectation in term of $\big(i(t), j(t)\big)$ given all random variables (3). Note that $x(t)$ depends on $i(0), j(0), \ldots, i(t-1)$, $j(t-1)$ but not on $i(t'), j(t')$ for any $t' \ge t$. Thus,

$$\begin{aligned} & \mathbb{E}_{t|t-1}[f(s(t))] - f^\star \\ &\le f(x(t-\tau(t))) - f^\star \\ &\quad - \sum_{m=1}^{M}\sum_{b=1}^{B} \frac{\alpha p_m q_b}{L_m l_b} \langle \nabla_b f(x(t-\tau(t))), \nabla_b f_m(x(t-\tau(t))) \rangle \\ &\quad + \sum_{m=1}^{M}\sum_{b=1}^{B} \frac{(1+\xi)\alpha^2 p_m q_b}{2L_m^2 l_b} \|\nabla_b f_m(x(t-\tau(t))) - \nabla_b f_m(x^\star)\|^2 \\ &\quad + \sum_{m=1}^{M}\sum_{b=1}^{B} \left(1 + \frac{1}{\xi}\right) \frac{\alpha^2 p_m q_b}{2L_m^2 l_b} \|\nabla_b f_m(x^\star)\|^2, \\ &= f(x(t-\tau(t))) - f^\star \\ &\quad - \frac{\alpha}{\sum_{m=1}^{M} L_m \sum_{b=1}^{B} l_b} \big\|\nabla f(x(t-\tau(t)))\big\|^2 \\ &\quad + \frac{(1+\xi)\alpha^2}{\sum_{m=1}^{M} L_m \sum_{b=1}^{B} l_b} \sum_{m=1}^{M} \frac{1}{2L_m} \|\nabla f_m(x(t-\tau(t))) - \nabla f_m(x^\star)\|^2 \\ &\quad + \left(1 + \frac{1}{\xi}\right) \frac{\alpha^2}{\sum_{m=1}^{M} L_m \sum_{b=1}^{B} l_b} \sum_{m=1}^{M} \frac{1}{2L_m} \|\nabla f_m(x^\star)\|^2 \qquad (13) \\ &= f(x(t-\tau(t))) - f^\star \\ &\quad - c\big\|\nabla f(x(t-\tau(t)))\big\|^2 \\ &\quad + (1+\xi)\alpha c \sum_{m=1}^{M} \frac{1}{2L_m} \|\nabla f_m(x(t-\tau(t))) - \nabla f_m(x^\star)\|^2 \\ &\quad + \left(1 + \frac{1}{\xi}\right) \alpha c \sum_{m=1}^{M} \frac{1}{2L_m} \|\nabla f_m(x^\star)\|^2, \qquad (14) \end{aligned}$$

where

$$c = \frac{\alpha}{\sum_{m=1}^{M} L_m \sum_{b=1}^{B} l_b}.$$

For each $m = 1, \dots, M$, f_m is convex and L_m-smooth. Therefore, according to [11, Theorem 2.1.5], it holds that

$$\begin{aligned}&\|\nabla f_m(x(t-\tau(t))) - \nabla f_m(x^\star)\|^2 \\ &\quad \le L_m \langle f_m(x(t-\tau(t))) - \nabla f_m(x^\star), x(t-\tau(t)) - x^\star \rangle.\end{aligned}$$

Substituting this inequality into (14) yields

$$\begin{aligned}&\mathbb{E}_{t|t-1}[f(s(t))] - f^\star \\ &\quad \le f(x(t-\tau(t))) - f^\star \\ &\qquad - c\big\|\nabla f(x(t-\tau(t)))\big\|^2 \\ &\qquad + \frac{(1+\xi)\alpha c}{2}\big\langle \nabla f(x(t-\tau(t))), x(t-\tau(t)) - x^\star \big\rangle \\ &\qquad + \left(1 + \frac{1}{\xi}\right)\frac{\alpha c}{2}\sum_{m=1}\frac{1}{L_m}\|\nabla f_m(x^\star)\|^2.\end{aligned}$$

Since f is μ-strongly convex, it follows from [11, Theorem 2.1.10] that

$$\big\langle \nabla f(x(t-\tau(t))), x(t-\tau(t)) - x^\star \big\rangle \le \frac{1}{\mu}\big\|\nabla f(x(t-\tau(t)))\big\|^2,$$

which implies that

$$\begin{aligned}&\mathbb{E}_{t|t-1}[f(s(t))] - f^\star \\ &\qquad \le f(x(t-\tau(t))) - f^\star \\ &\qquad\quad - c\left(1 - \frac{(1+\xi)\alpha}{2\mu}\right)\big\|\nabla f(x(t-\tau(t)))\big\|^2 \\ &\qquad\quad + \left(1 + \frac{1}{\xi}\right)\frac{\alpha c}{2}\sum_{m=1}^{M}\frac{1}{L_m}\|\nabla f_m(x^\star)\|^2.\end{aligned}$$

Moreover, according to [8, Theorem 2], it holds that

$$2\mu\big(f(x(t-\tau(t))) - f^\star\big) \le \big\|\nabla f(x(t-\tau(t)))\big\|^2.$$

Thus, if we take

$$\alpha \in \left(0, \frac{2\mu}{1+\xi}\right), \tag{15}$$

we have

$$\begin{aligned}&\mathbb{E}_{t|t-1}[f(s(t))] - f^\star \\ &\quad \le \left(1 - 2\mu c\left(1 - \frac{(1+\xi)\alpha}{2\mu}\right)\right)\Big(f(x(t-\tau(t))) - f^\star\Big) \\ &\qquad + \left(1 + \frac{1}{\xi}\right)\frac{\alpha c}{2}\sum_{m=1}^{M}\frac{1}{L_m}\|\nabla f_m(x^\star)\|^2. \qquad (16)\end{aligned}$$

Define the sequence $\{V(t)\}$ by

$$V(t) = \mathbb{E}_{t-1}\big[f(x(t))\big] - f^\star, \quad t \in \mathbb{N}$$

and note that

$$\begin{aligned}V(t+1) &= \mathbb{E}_t\big[f(x(t+1))\big] - f^\star \\ &= \mathbb{E}_{t-1}\big[\mathbb{E}_{t|t-1}[f(x(t+1))]\big] - f^\star.\end{aligned}$$

By combing this fact with (12) and (16), we obtain

$$\begin{aligned}V(t+1) \le\ & \underbrace{(1-\theta)}_{p} V(t) \\ &+ \underbrace{\theta\left(1 - 2\mu c\left(1 - \frac{(1+\xi)\alpha}{2\mu}\right)\right)}_{q} V(t-\tau(t)) \\ &+ \underbrace{\theta\left(1 + \frac{1}{\xi}\right)\frac{\alpha c}{2}\sum_{m=1}^{M}\frac{1}{L_m}\|\nabla f_m(x^\star)\|^2}_{r}.\end{aligned}$$

One can verify that for any α satisfying (15),

$$p + q = 1 - 2\theta\mu c\left(1 - \frac{(1+\xi)\alpha}{2\mu}\right) \in \big(0, 1\big).$$

It now follows from Lemma 1 that

$$V(t) \le \rho^t V(0) + e, \quad t \in \mathbb{N}_0,$$

where

$$\rho = (p+q)^{\frac{1}{1+\tau_{\max}}} = \left(1 - 2\theta\mu c\left(1 - \frac{(1+\xi)\alpha}{2\mu}\right)\right)^{\frac{1}{1+\tau_{\max}}},$$

and

$$e = \frac{r}{1-p-q} = \frac{(1+\xi)\alpha}{2\xi(2\mu - \alpha(1+\xi))}\sum_{m=1}^{M}\frac{1}{L_m}\|\nabla f_m(x^\star)\|^2.$$

Letting $\xi = 1/(1+\theta)$ completes the proof.

REFERENCES

[1] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*. Prentice-Hall, 1989.

[2] Y. A. Censor and S. A. Zenios, *Parallel Optimization: Theory, Algorithms and Applications*. Oxford University Press, 1997.

[3] Amazon Web Services, Inc., "AWS—Amazon Elastic Compute Cloud (EC2) - Scalable Cloud Hosting," October 2014. [Online]. Available: http://aws.amazon.com/ec2/

[4] A. Nedić, D. Bertsekas, and V. Borkar, "Distributed asynchronous incremental subgradient methods," *Studies in Computational Mathematics*, vol. 8, pp. 381–407, 2001.

[5] F. Niu, B. Recht, C. Ré, and S. J. Wright, "Hogwild!: A lock-free approach to parallelizing stochastic gradient descent," in *NIPS*, 2011, pp. 693–701.

[6] O. Fercoq and P. Richtárik, "Accelerated, parallel and proximal coordinate descent," *arXiv preprint arXiv:1312.5799*, 2013.

[7] M. Li, L. Zhou, Z. Yang, A. Li, F. Xia, D. G. Andersen, and A. Smola, "Parameter server for distributed machine learning," in *Big Learning Workshop, NIPS*, 2013.

[8] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems." *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[9] H. R. Feyzmahdavian, A. Aytekin, and M. Johansson, "A delayed proximal gradient method with linear convergence rate," *IEEE Workshop on Machine Learning for Signal Processing (MLSP)*, 2014.

[10] L. Xiao and T. Zhang, "A proximal stochastic gradient method with progressive variance reduction," *arXiv preprint arXiv:1403.4699*, 2014.

[11] Y. Nesterov, *Introductory lectures on convex optimization : a basic course*. Springer, 2004.