This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2016.2525015, IEEE Transactions on Automatic Control

An Asynchronous Mini–batch Algorithm for Regularized Stochastic Optimization

Hamid Reza Feyzmahdavian, Arda Aytekin, and Mikael Johansson

***Abstract*—Mini-batch optimization has proven to be a powerful paradigm for large-scale learning. However, the state of the art parallel mini–batch algorithms assume synchronous operation or cyclic update orders. When worker nodes are heterogeneous (due to different computational capabilities or different communication delays), synchronous and cyclic operations are inefficient since they will leave workers idle waiting for the slower nodes to complete their computations. In this paper, we propose an asynchronous mini-batch algorithm for regularized stochastic optimization problems with smooth loss functions that eliminates idle waiting and allows workers to run at their maximal update rates. We show that by suitably choosing the step-size values, the algorithm achieves a rate of the order $\mathcal{O}(1/\sqrt{T})$ for general convex regularization functions, and the rate $\mathcal{O}(1/T)$ for strongly convex regularization functions, where T is the number of iterations. In both cases, the impact of asynchrony on the convergence rate of our algorithm is asymptotically negligible, and a near-linear speedup in the number of workers can be expected. Theoretical results are confirmed in real implementations on a distributed computing infrastructure.**

I. Introduction

Many optimization problems that arise in machine learning, signal processing, and statistical estimation can be formulated as *regularized stochastic optimization* (also referred to as *stochastic composite optimization*) problems in which one jointly minimizes the expectation of a stochastic loss function plus a possibly nonsmooth regularization term. Examples include Tikhonov and elastic net regularization, Lasso, sparse logistic regression, and support vector machines [1]–[3].

Stochastic gradient methods were among the first and the most commonly used algorithms developed for solving stochastic optimization problems [4]–[10]. Their popularity comes mainly from the fact that they are easy to implement and have low computational cost per iteration. Stochastic gradient methods are inherently *serial* in the sense that the gradient computations take place on a single processor which has access to the whole dataset. However, it happens more and more often that one single computer is unable to store and handle the amounts of data that we encounter in practical problems. This has caused a strong interest in developing *parallel* optimization algorithms which are able to split the data and distribute the computation across multiple processors or multiple computer clusters (see, *e.g.*, [11]–[16] and references therein). The performance of Google's DistBelief model [17] and Microsoft's Project Adam [18] have proven that parallel stochastic gradient methods are remarkably effective in real-world machine learning problems such as training deep learning systems. For example, while training a neural network for the ImageNet task with 16 million images may take about two weeks on a modern GPU, Google's DistBelief model can successfully utilize 16,000 cores in parallel and train the network for three days [17].

A common practical solution for parallelizing stochastic gradient methods is *mini-batching*, where iterates are updated based on the average gradient with respect to multiple data points rather than based on gradients evaluated at a single data at a time. Recently, Dekel *et. al.* [19] proposed a parallel mini-batch algorithm for regularized stochastic optimization problems, in which multiple processors compute gradients in parallel using their own local data, and then aggregate the gradients up a spanning tree to obtain the averaged gradient. While this algorithm can achieve linear speedup in the number of processors, it has the drawback that the processors need to synchronize at each round and, hence, if one of them is slower than the rest, then the entire algorithm runs at the pace of the slowest processor. Furthermore, the need for global synchronization and requiring massive communication overhead make this method fragile to many types of failures that are common in distributed computing environments. For example, if one processor fails throughout the execution of the algorithm or is disconnected from the network connecting the processors, the algorithm will come to an immediate halt.

Asynchronous computation has a long history in optimization. Many early results were unified and significantly extended in the influential book by Bertsekas and Tsitsklis [20]. In contrast to synchronous algorithms, asynchrony allows the processors to compute gradients at different rates without synchronization, and lets each processor perform its update independently of other processors by using out-of-date gradients. Some advantages that we can gain from asynchronous implementations of optimization algorithms:

- Reduced idle time of processors;
- More iterates executed by fast processors;
- Alleviated congestion in inter-process communication;
- Robustness to individual processor failures.

However, on the negative side, asynchrony runs the risk of rendering an otherwise convergent algorithm divergent. Note that convergence analysis of asynchronous algorithms tends to be more challenging since their dynamics are much richer [20], and asynchronous optimization algorithms often converge under more restrictive conditions than their synchronous counterparts. Thus, tuning an algorithm to withstand large amounts

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2016.2525015, IEEE Transactions on Automatic Control

of asynchrony will typically result in unnecessarily slow convergence if the actual implementation is synchronous.

There have been extensive studies on asynchronous stochastic optimization, but mostly under the assumption that the loss function is *nonsmooth* with *bounded* subgradients, see, *e.g.*, [21]–[23]. The literature on asynchronous algorithms for *smooth* stochastic optimization is relatively sparse. The main objective of this paper is to make a contribution in this direction by proposing an asynchronous mini-batch algorithm for regularized stochastic optimization problems with smooth loss functions that eliminates the overhead associated with global synchronization. Our algorithm allows multiple processors to work at *different rates*, perform computations *independently* of each other, and update global decision variables using *out-of-date* gradients. A similar model of parallel asynchronous computation was applied to coordinate descent methods for deterministic optimization in [24]–[26] and mirror descent and dual averaging methods for stochastic optimization in [27]. In particular, Agarwal and Duchi [27] have analyzed the convergence of asynchronous mini-batch algorithms for smooth stochastic convex problems, and interestingly shown that bounded delays do not degrade the asymptotic convergence. However, they only considered the case where the regularization term is the indicator function of a compact convex set. Moreover, convergence rates for strongly convex stochastic problems was not discussed in [27].

We extend the results of [27] to general regularization functions (like the l_1 norm, often used to promote sparsity), and establish a sharper expected-value type of convergence rate than the one given in [27]. Specifically, we make the following contributions:

(*i*) For general convex regularization functions, we show that when the feasible set is closed and convex (but not necessarily bounded), the running average of the iterates generated by our algorithm with constant step-sizes converges at rate $\mathcal{O}(1/T)$ to a ball around the optimum. We derive an explicit expression that quantifies how the convergence rate and the residual error depend on loss function properties and algorithm parameters such as the step-size and the maximum delay bound $\tau_{\max}$.

(*ii*) For general convex regularization functions and compact feasible sets, we prove that the running average of the iterates produced by our algorithm with a time-varying step-size converges to the true optimum (without residual error) at rate

$$\mathcal{O}\left(\frac{(\tau_{\max}+1)^2}{T}+\frac{1}{\sqrt{T}}\right).$$

As long as the number of processors is $\mathcal{O}(T^{1/4})$, our algorithm enjoys near-linear speedup and converges asymptotically at a rate $\mathcal{O}(1/\sqrt{T})$. This rate is known to be optimal for convex stochastic problems even in the absence of delays [6], [9].

(*iii*) When the regularization function is strongly convex and the feasible set is closed and convex, we establish that the iterates converge at rate

$$\mathcal{O}\left(\frac{(\tau_{\max}+1)^4}{T^2}+\frac{1}{T}\right).$$

If the number of processors is of the order of $\mathcal{O}(T^{1/4})$, this rate is $\mathcal{O}(1/T)$ asymptotically in T, which is the best known rate for strongly convex stochastic optimization problems in a serial setting [28]–[30].

The remainder of the paper is organized as follows. In Section II, we introduce the notation and review some preliminaries that are essential for the development of the main results in this paper. In Section III, we formulate the problem and discuss our assumptions. The proposed asynchronous mini-batch algorithm and its main theoretical results are presented in Section IV. Section V reports on promising computational results. Finally, conclusions and potential extensions are given in Section VI.

II. Notation and Preliminaries

A. Notation

We let $\mathbb{N}$ and $\mathbb{N}_0$ denote the set of natural numbers and the set of natural numbers including zero, respectively. The inner product of two vectors $x, y \in \mathbb{R}^n$ is denoted by $\langle x, y\rangle$. We assume that $\mathbb{R}^n$ is endowed with a norm $\|\cdot\|$, and use $\|\cdot\|_*$ to represent the corresponding dual norm, defined by

$$\|y\|_* = \sup_{\|x\|\leq 1} \langle x, y\rangle.$$

B. Preliminaries

We start with the definition of a *Bregman distance function*, also referred to as a *prox-function*.

Definition 1: *A function $\omega : C \to \mathbb{R}$ is called a distance generating function with modulus $\mu_\omega > 0$ with respect to norm $\|\cdot\|$, if ω is continuously differentiable and μ_ω-strongly convex with respect to $\|\cdot\|$ over the convex set $C \subseteq \mathbb{R}^n$. That is, for all $x, y \in C$,*

$$\omega(y) \geq \omega(x) + \langle \nabla\omega(x), y - x\rangle + \frac{\mu_\omega}{2}\|y - x\|^2.$$

Every distance generating function introduces a corresponding Bregman distance function given by

$$D_\omega(x, y) := \omega(y) - \omega(x) - \langle \nabla\omega(x), y - x\rangle.$$

For example, choosing $\omega(x) = \frac{1}{2}\|x\|_2^2$, which is 1-strongly convex with respect to the l_2-norm over any convex set C, would result in $D_\omega(x, y) = \frac{1}{2}\|y - x\|_2^2$. Another common example of distance generating functions is the entropy function

$$\omega(x) = \sum_{i=1}^{n} x_i \log x_i,$$

which is 1-strongly convex with respect to the l_1-norm over the standard simplex

$$\Delta := \left\{ x \in \mathbb{R}^n \;\middle|\; \sum_{i=1}^{n} x_i = 1,\ x \geq 0 \right\},$$

and its associated Bregman distance function is

$$D_\omega(x, y) = \sum_{i=1}^{n} y_i \log \frac{y_i}{x_i}.$$

The main motivation to use a generalized distance generating function, instead of the usual Euclidean distance function, is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2016.2525015, IEEE Transactions on Automatic Control

to design optimization algorithms that can take advantage of the geometry of the feasible set (see, *e.g.*, [5], [31]–[33]).

Remark 1: The strong convexity of the distance generating function ω always ensures that

$$D_\omega(x,y) \geq \frac{\mu_\omega}{2}\|y-x\|^2, \quad \forall x,y \in C,$$

and $D_\omega(x,y) = 0$ if and only if $x = y$.

Remark 2: Throughout the paper, there is no loss of generality to assume that $\mu_\omega = 1$. Indeed, if $\mu_\omega \neq 1$, we can choose the scaled function $\overline{\omega}(x) = \frac{1}{\mu_\omega}\omega(x)$, which has modulus $\overline{\mu}_\omega = 1$, to generate the Bregman distance function.

The following definition introduces *subgradients* of proper convex functions.

Definition 2: *For a convex function $\Psi : \mathbb{R}^n \to \mathbb{R}\cup\{+\infty\}$, a vector $s \in \mathbb{R}^n$ is called a subgradient of Ψ at $x \in \mathbb{R}^n$ if*

$$\Psi(y) \geq \Psi(x) + \langle s, y-x\rangle, \quad \forall y \in \mathbb{R}^n.$$

The set of all subgradients of Ψ at x is called the subdifferential of Ψ at x, and is denoted by $\partial\Psi(x)$.

III. Problem Setup

We consider the stochastic convex optimization problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ \phi(x) := \mathbb{E}_\xi\big[F(x,\xi)\big] + \Psi(x). \tag{1}$$

Here, x is the decision variable, ξ is a random vector whose probability distribution $\mathcal{P}$ is supported on a set $\Xi \subseteq \mathbb{R}^m$, $F(\cdot,\xi)$ is convex and differentiable for each $\xi \in \Xi$, and $\Psi(x)$ is a proper convex function that may be nonsmooth and extended real-valued. Let us define

$$f(x) := \mathbb{E}_\xi\big[F(x,\xi)\big] = \int_\Xi F(x,\xi)d\mathcal{P}(\xi). \tag{2}$$

Note that the expectation function f is convex, differentiable, and $\nabla f(x) = \mathbb{E}_\xi[\nabla_x F(x,\xi)]$ [34]. Thus, $\nabla_x F(x,\xi)$ can be viewed as an unbiased estimate of $\nabla f(x)$. We use $X^\star$ to denote the set of optimal solutions of Problem (1) and $\phi^\star$ to denote the corresponding optimal value.

A difficulty when solving Problem (1) is that the distribution $\mathcal{P}$ is often unknown, so the expectation (2) cannot be computed. This situation occurs frequently in data-driven applications such as machine learning. To support these applications, we do not assume knowledge of f (or of $\mathcal{P}$), only access to a stochastic oracle. Each time the oracle is queried with an $x \in \mathbb{R}^n$, it generates an independent and identically distributed (i.i.d.) sample ξ from $\mathcal{P}$ and returns $\nabla_x F(x,\xi)$, which is a noise-corrupted version of $\nabla f(x)$. The erroneous gradient $\nabla_x F(x,\xi)$ will be used in the update rule of our optimization algorithm instead of $\nabla f(x)$.

We also impose the following assumptions on Problem (1).

Assumption 1 (Existence of a minimum): The optimal set $X^\star$ is nonempty.

Assumption 2 (Lipschitz continuity of F): For each $\xi \in \Xi$ the function $F(\cdot,\xi)$ has Lipschitz continuous gradient with constant L. That is, for all $y, z \in \mathbb{R}^n$,

$$\|\nabla_x F(y,\xi) - \nabla_x F(z,\xi)\|_* \leq L\|y-z\|. \tag{3}$$

Note that under Assumption 2, $\nabla f(x)$ is also Lipschitz continuous with the same constant L [9].

Assumption 3 (Bounded gradient variance): There exists a constant $\sigma \geq 0$ such that

$$\mathbb{E}_\xi\big[\|\nabla_x F(x,\xi) - \nabla f(x)\|_*^2\big] \leq \sigma^2, \quad \forall x \in \mathbb{R}^n.$$

In the case when the gradients are evaluated without any errors, *i.e*, $\nabla f(x) = \nabla_x F(x,\xi)$, we can set $\sigma = 0$.

Assumption 4 (Closed effective domain of Ψ): The function Ψ is simple and lower semi-continuous, and its effective domain, $\operatorname{dom}\Psi = \{x \in \mathbb{R}^n \mid \Psi(x) < +\infty\}$, is closed.

Possible choices of Ψ include:

- *Unconstrained smooth minimization*: $\Psi(x) = 0$.
- *Constrained smooth minimization*: Ψ is the indicator function of a non-empty closed convex set $C \subseteq \mathbb{R}^n$, *i.e.*,
$$\Psi(x) = I_C(x) := \begin{cases} 0, & \text{if } x \in C, \\ +\infty, & \text{otherwise.} \end{cases}$$
- *l_1-regularized minimization*: $\Psi(x) = \lambda\|x\|_1$ with $\lambda > 0$.
- *Constrained l_1-regularized minimization*: In this case, $\Psi(x) = \lambda\|x\|_1 + I_C(x)$ with $\lambda > 0$.

Several practical problems in machine learning, statistical applications, and signal processing satisfy Assumptions 1–4 (see, *e.g.*, [1]–[3]). One such example is *l_1-regularized logistic regression* for sparse binary classification. We are then given a large number of observations

$$\Big\{\xi_j = \big(\xi_j^{(1)}, \xi_j^{(2)}\big) \mid \xi_j^{(1)} \in \mathbb{R}^n,\ \xi_j^{(2)} \in \{-1,+1\},\ j = 1,\ldots,J\Big\},$$

drawn i.i.d. from an unknown distribution $\mathcal{P}$, and want to solve Problem (1) with

$$F(x,\xi) = \log\Big(1 + \exp\big(-\xi_j^{(2)}\langle\xi_j^{(1)}, x\rangle\big)\Big),$$

and $\Psi(x) = \lambda\|x\|_1$. The role of l_1 regularization is to produce sparse solutions.

In many emerging applications, such as large-scale machine learning and statistics, the size of dataset is so huge that it cannot fit on one computer. Hence, we need optimization algorithms that can be conveniently and efficiently executed in parallel on multiple processors. Our goal is (i) to develop an algorithm for solving regularized stochastic optimization problems which combines the strong performance guarantees of serial stochastic gradient methods, the parallelization benefits of mini-batching algorithms, and the speed-ups enabled by asynchronous implementations; (ii) to extend the analysis in [27] to solve Problem (1) with *general* regularization functions (not necessarily $\Psi(x) = I_C(x)$) without any additional assumption on boundedness of either the gradients or the feasible sets; and (iii) to determine whether an asynchronous mini-batch algorithm achieves the optimal rate $\mathcal{O}\big(1/T\big)$ under the strong convexity assumption.

IV. An Asynchronous Mini-Batch Algorithm

In this section, we present an *asynchronous* mini-batch algorithm that exploits multiple processors to solve Problem (1). We characterize the iteration complexity and the convergence rate of the proposed algorithm, and show that these compare

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2016.2525015, IEEE Transactions on Automatic Control

TABLE I: Comparison of our algorithm with selected recent algorithms in the literature for stochastic convex optimization.

Reference	Regularized stochastic optimization	Parallel	Asynchronous	Convergence rate for		Notable feature
				convex ϕ	strongly convex ϕ	
[6]	Yes	×	×	Yes	×	Mirror descent method with optimal rate
[8]	Yes	×	×	Yes	Yes	Mirror descent method with optimal rate
[9]	Yes	×	×	Yes	Yes	Dual-averaging method with optimal rate
[19]	Yes	Yes	×	Yes	Yes	Mini-batch method with linear speedup
[27]	$\Psi(x) = I_C(x)$	Yes	Yes	Yes	×	Asynchronous method with linear speedup
Our work	Yes	Yes	Yes	Yes	Yes	

favourably with the state of the art. Our approach is distinguished from recent work on stochastic optimization [6]–[9], [19], [27] in that it can deal with *asynchrony* and *smooth* objective functions as well as *general* regularization functions at the same time, cf. Table I. To the best of our knowledge, our asynchronous algorithm is the first to attain the optimal convergence rates for convex and strongly convex stochastic composite optimization in spite of time-varying delays.

A. Description of Algorithm

We assume p processors have access to a shared memory for the decision variable x. The processors may have different capabilities (in terms of processing power and access to data) and are able to update x without the need for coordination or synchronization. Conceptually, the algorithm lets each processor run its own stochastic composite mirror descent process, repeating the following steps:

1) Read x from the shared memory and load it into the local storage location $\widehat{x}$;
2) Sample b i.i.d random variables $\xi_1, \dots, \xi_b$ from the distribution $\mathcal{P}$;
3) Compute the averaged stochastic gradient vector
$$\widehat{g}_{\text{ave}} = \frac{1}{b}\sum_{i=1}^{b} \nabla_x F(\widehat{x}, \xi_i);$$
4) Update current x in the shared memory via
$$x \leftarrow \operatorname*{argmin}_z \left\{ \langle \widehat{g}_{\text{ave}}, z\rangle + \Psi(z) + \frac{1}{\gamma} D_\omega(x, z) \right\}.$$

The algorithm can be implemented in many ways as depicted in Figure 1. One way is to consider the p processors as peers that each execute the four-step algorithm independently of each other and only share the global memory for storing x. In this case, each processor reads the decision vector twice in each round: once in the first step (before evaluating the averaged gradient), and once in the last step (before carrying out the minimization). To ensure correctness, Step 4 must be an atomic operation, where the executing processor puts a write lock on the global memory until it has written back the result of the minimization (cf. Figure 1, left). The algorithm can also be executed in a master-worker setting. In this case, each of the worker nodes retrieves x from the master in Step 1 and returns the averaged gradient to the master in Step 3; the fourth step (carrying out the minimization) is executed by the master (cf. Figure 1, right).

Independently of how we choose to implement the algorithm, processors may work at different rates: while one processor updates the decision vector (in the shared memory setting) or sends its averaged gradient to the master (in the master-worker setting), the others are generally busy computing averaged gradient vectors. The processors that perform gradient evaluations do not need to be aware of updates to the decision vector, but can continue to operate on stale information about x. Therefore, unlike *synchronous* parallel mini-batch algorithms [19], there is no need for processors to wait for each other to finish the gradient computations. Moreover, the value $\widehat{x}$ at which the average of gradients is evaluated by a processor may differ from the value of x to which the update is applied.

Algorithm 1 Asynchronous Algorithm (running on each processor)

1: **Inputs**: positive step-sizes $\{\gamma(k)\}_{k\in\mathbb{N}_0}$; batch size $b \in \mathbb{N}$.
2: **Initialization:** $x(0) \in \operatorname{dom}\Psi$; $k = 0$.
3: **repeat**
4: receive b inputs $\xi_1, \dots, \xi_b$ sampled i.i.d. from distribution $\mathcal{P}$;
$$g_{\text{ave}}\big(d(k)\big) \leftarrow \frac{1}{b}\sum_{i=1}^{b} \nabla_x F\big(x(d(k)), \xi_i\big);$$
$$x(k+1) \leftarrow \operatorname*{argmin}_z \left\{ \big\langle g_{\text{ave}}\big(d(k)\big), z\big\rangle + \Psi(z) + \frac{1}{\gamma(k)} D_\omega(x(k), z) \right\} \quad (4)$$
$$k \leftarrow k+1;$$
5: **until** termination test satisfied

Algorithm 1 describes the p asynchronous processes that run in parallel. To describe the progress of the overall optimization process, we introduce a counter k that is incremented each time x is updated. We let $d(k)$ denote the time at which $\widehat{x}$ used to compute the averaged gradient involved in the update of $x(k)$ was read from the shared memory. It is clear that $0 \le d(k) \le k$ for all $k \in \mathbb{N}_0$. The value
$$\tau(k) := k - d(k)$$
can be viewed as the delay between reading and updating for processors. Moreover, $\tau(k)$ captures the staleness of the information used to compute the average of gradients for the k^{th} update. We assume that the time-varying delay $\tau(k)$ is bounded; this is stated in the following assumption.

Assumption 5 (Bounded Delay): There is a nonnegative integer $\tau_{\max}$ such that
$$0 \le \tau(k) \le \tau_{\max}$$
for all $k \in \mathbb{N}_0$.

The value of $\tau_{\max}$ is an indicator of the asynchronism in the algorithm and in the execution platform. In practice, $\tau_{\max}$ will depend on the number of parallel processors used

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2016.2525015, IEEE Transactions on Automatic Control

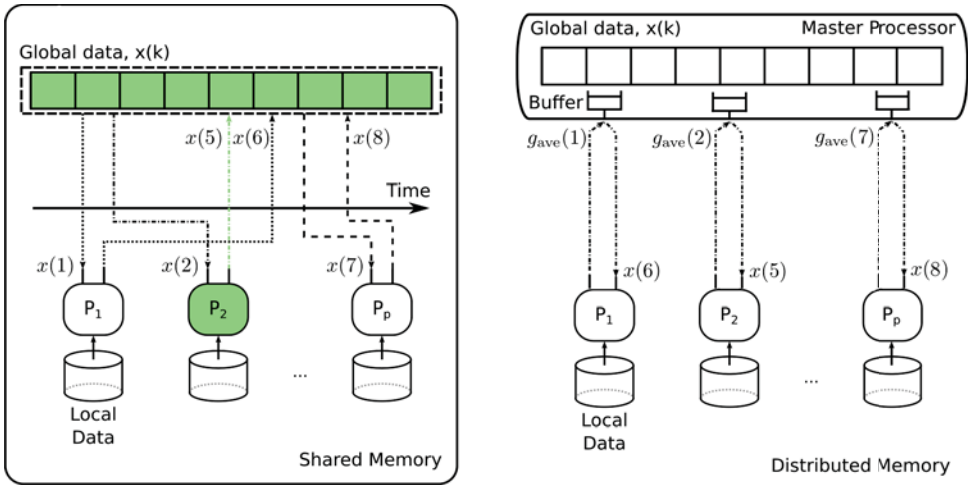

Fig. 1: Illustration of two conceptually different realizations of Algorithm 1: (1) a shared memory implementation (left); (2) a master-worker implementation (right). In the shared memory setting shown to the left, processor P_2 reads $x(2)$ from the shared memory and computes the averaged gradient vector $g_{\text{ave}}(2) = \frac{1}{b}\sum_{i=1}^{b} \nabla_x F\big(x(2), \xi_i\big)$. As the processors are being run without synchronization, $x(3)$ and $x(4)$ are written to the shared memory by other processors while P_2 is evaluating $g_{\text{ave}}(2)$. The figure shows a snapshot of the algorithm at time instance $k = 5$, at which the shared memory is locked by P_2 to read the current x, *i.e.* $x(4)$, to update it using the out-of-date averaged gradient $g_{\text{ave}}(2)$, and write $x(5)$ to the memory. In the master-worker setting illustrated to the right, workers evaluate averaged gradient vectors in parallel and send their computations to buffers on the master processor, which is the sole entity with access to the global memory. The master performs an update using (possibly) out-of-date averaged gradients and passes the updated x back to the workers.

in the algorithm [24]–[26]. Note that the cyclic-delay mini-batch algorithm [27], in which the processors are ordered and each updates the decision variable under a fixed schedule, is a special case of Algorithm 1 where $d(k) = k - p + 1$, or, equivalently, $\tau(k) = p - 1$ for all k.

B. Convergence Rate for General Convex Regularization

The following theorem establishes convergence properties of Algorithm 1 when a constant step-size is used.

Theorem 1: *Let Assumptions 1–5 hold. Assume also that*

$$\gamma(k) = \gamma \in \left(0,\ \frac{1}{L(\tau_{\max}+1)^2}\right). \tag{5}$$

Then, for every $T \in \mathbb{N}$ and any optimizer $x^\star$ of (1)*, we have*

$$\mathbb{E}\big[\phi\big(x_{\text{ave}}(T)\big)\big] - \phi^\star \leq \frac{D_\omega\big(x(0), x^\star\big)}{\gamma T} + \frac{\gamma c \sigma^2}{2b\big(1 - \gamma L(\tau_{\max}+1)^2\big)},$$

where $x_{\text{ave}}(T)$ is the Cesáro average of the iterates, i.e.,

$$x_{\text{ave}}(T) := \frac{1}{T}\sum_{k=1}^{T} x(k).$$

Furthermore, the expectation is taken with respect to all random variables $\{\xi_i(k) \mid i = 1, \ldots, b,\ k = 0, \ldots, T-1\}$, b is the batch size, and $c \in [1, b]$ is given by

$$c = \begin{cases} 1, & \text{if } \|\cdot\| = \|\cdot\|_2, \\ 2\max_{\|x\|\leq 1} \omega(x), & \text{otherwise.} \end{cases}$$

Proof: See Appendix A. ∎

Theorem 1 demonstrates that for any constant step-size γ satisfying (5), the running average of iterates generated by Algorithm 1 will converge in expectation to a ball around the optimum at a rate of $\mathcal{O}(1/T)$. The convergence rate and the residual error depend on the choice of γ: decreasing γ reduces the residual error, but it also results in a slower convergence. We now describe a possible strategy for selecting the constant step-size. Let T_ϵ be the total number of iterations necessary to achieve ϵ-optimal solution to Problem (1), that is, $\mathbb{E}\big[\phi\big(x_{\text{ave}}(T)\big)\big] - \phi^\star \leq \epsilon$ when $T \geq T_\epsilon$. If we pick

$$\gamma = \frac{\epsilon}{L\epsilon(\tau_{\max}+1)^2 + c\sigma^2/b}, \tag{6}$$

then, using Theorem 1, the corresponding $x_{\text{ave}}(T)$ satisfies

$$\mathbb{E}\big[\phi\big(x_{\text{ave}}(T)\big)\big] - \phi^\star \leq \frac{\epsilon_0}{T}\left(L(\tau_{\max}+1)^2 + \frac{c\sigma^2}{b\epsilon}\right) + \frac{\epsilon}{2},$$

where $\epsilon_0 = D_\omega\big(x(0), x^\star\big)$. This inequality tells us that if the first term on the right-hand side is less than $\epsilon/2$, *i.e.*, if

$$T \geq T_\epsilon := 2\epsilon_0\left(\frac{L(\tau_{\max}+1)^2}{\epsilon} + \frac{c\sigma^2}{b\epsilon^2}\right),$$

then $\mathbb{E}\big[\phi\big(x_{\text{ave}}(T)\big)\big] - \phi^\star \leq \epsilon$. Hence, the iteration complexity of Algorithm 1 with the step-size choice (6) is given by

$$\mathcal{O}\left(\frac{L(\tau_{\max}+1)^2}{\epsilon} + \frac{c\sigma^2}{b\epsilon^2}\right). \tag{7}$$

As long as the maximum delay bound $\tau_{\max}$ is of the order $1/\sqrt{\epsilon}$, the first term in (7) is asymptotically negligible. In this case, the iteration complexity of Algorithm 1 is asymptotically $\mathcal{O}(c\sigma^2/b\epsilon^2)$, which is exactly the iteration complexity achieved by a serial mini-batch algorithm [19]. As discussed before, $\tau_{\max}$ is related to the number of processors. Therefore, if the number of processors is of the order of $\mathcal{O}(1/\sqrt{\epsilon})$, parallelization does not appreciably degrade asymptotic convergence of Algorithm 1. Furthermore, as p processors are being run asynchronously and in parallel, updates may occur

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2016.2525015, IEEE Transactions on Automatic Control

roughly p times as quickly, which means that the near-linear speedup in the number of processors can be expected.

Remark 3: Another strategy for the selection of the constant step-size in Algorithm 1 is to use γ that depends on the prior knowledge of the number of iterations to be performed. More precisely, assume that the number of iterations is fixed in advance, say equal to T_F. By choosing γ as

$$\gamma = \frac{1}{L(\tau_{\max}+1)^2 + \alpha\sqrt{T_F}},$$

for some $\alpha > 0$, it follows from Theorem 1 that the running average of the iterates after T_F iterations satisfies

$$\mathbb{E}\big[\phi\big(x_{\text{ave}}(T_F)\big)\big] - \phi^\star \leq \frac{L(\tau_{\max}+1)^2 D_\omega\big(x(0),x^\star\big)}{T_F} + \frac{1}{\sqrt{T_F}}\left(\alpha D_\omega\big(x(0),x^\star\big) + \frac{c\sigma^2}{2\alpha b}\right).$$

The optimal choice of α, which minimizes the second term on the right-hand-side of the above inequality, is

$$\alpha^\star = \frac{\sigma\sqrt{c}}{\sqrt{2bD_\omega\big(x(0),x^\star\big)}}.$$

With this choice of α, we then have

$$\mathbb{E}\big[\phi\big(x_{\text{ave}}(T_F)\big)\big] - \phi^\star \leq \frac{L(\tau_{\max}+1)^2 D_\omega\big(x(0),x^\star\big)}{T_F} + \frac{\sigma\sqrt{2cD_\omega\big(x(0),x^\star\big)}}{\sqrt{bT_F}}.$$

In the case that $\tau_{\max} = 0$, the preceding guaranteed bound reduces to the one obtained in [6, Theorem 1] for the serial stochastic mirror descent algorithm with constant step-sizes. Note that in order to implement Algorithm 1 with the optimal constant step-size policy, we need to estimate an upper bound on $D_\omega\big(x(0),x^\star\big)$, since $D_\omega\big(x(0),x^\star\big)$ is usually unknown.

The following theorem characterizes the convergence of Algorithm 1 with a time-varying step-size sequence when dom Ψ is *bounded* in addition to being closed and convex.

Theorem 2: *Suppose that Assumptions 1–5 hold. In addition, suppose that* dom Ψ *is compact and that* $D_\omega(\cdot,\cdot)$ *is bounded on* dom Ψ. *Let*

$$R^2 = \max_{x,y\in\text{dom }\Psi} D_\omega(x,y).$$

If $\{\gamma(k)\}_{k\in\mathbb{N}_0}$ *is set to* $\gamma(k)^{-1} = L(\tau_{\max}+1)^2 + \alpha(k)$ *with*

$$\alpha(k) = \frac{\sigma\sqrt{c}\sqrt{k+1}}{R\sqrt{b}},$$

then for all $T \in \mathbb{N}$, *the Cesáro average of the iterates generated by Algorithm 1 satisfies*

$$\mathbb{E}\big[\phi\big(x_{\text{ave}}(T)\big)\big] - \phi^\star \leq \frac{LR^2(\tau_{\max}+1)^2}{T} + \frac{2\sigma R\sqrt{c}}{\sqrt{bT}}.$$

Proof: See Appendix B. ∎

The time-varying step-size $\gamma(k)$, which ensures the convergence of the algorithm, consists of two terms: the time-varying term $\alpha(k)$ should control the errors from stochastic gradient information while the role of the constant term $L(\tau_{\max}+1)^2$ is to decrease the effects of asynchrony (bounded delays) on the convergence of the algorithm. According to Theorem 2, in the case that $\tau_{\max} = \mathcal{O}(T^{1/4})$, the delay becomes increasingly harmless as the algorithm progresses and the expected function value evaluated at $x_{\text{ave}}(T)$ converges asymptotically at a rate $\mathcal{O}(1/\sqrt{T})$, which is known to be the best achievable rate of the mirror descent method for nonsmooth stochastic convex optimization problems [5].

For the special case of Problem (1) where Ψ is restricted to be the indicator function of a compact convex set, Agarwal and Duchi [27, Theorem 2] showed that the convergence rate of the delayed stochastic mirror descent method with a time-varying step-size is

$$\mathcal{O}\left(\frac{LR^2 + RG\tau_{\max}}{T} + \frac{\sigma R\sqrt{c}}{\sqrt{bT}} + \frac{LR^2G^2\tau_{\max}^2 b\log T}{c\sigma^2 T}\right),$$

where G is the maximum bound on $\sqrt{\mathbb{E}[\|\nabla_x F(x,\xi)\|_*^2]}$. Comparing with this result, instead of a asymptotic penalty of the form $\mathcal{O}(\tau_{\max}^2\log T/T)$ due to the delays, we have the penalty $\mathcal{O}(\tau_{\max}^2/T)$, which is much smaller for large T. Therefore, not only do we extend the result of [27] to general regularization functions, but we also obtain a sharper guaranteed convergence rate than the one presented in [27].

C. Convergence Rate for Strongly Convex Regularization

In this subsection, we restrict our attention to stochastic composite optimization problems with strongly convex regularization terms. Specifically, we assume that Ψ is μ_Ψ-strongly convex with respect to $\|\cdot\|$, that is, for any $x, y \in$ dom Ψ,

$$\Psi(y) \geq \Psi(x) + \langle s, y-x\rangle + \frac{\mu_\Psi}{2}\|y-x\|^2, \quad \forall s \in \partial\Psi(x).$$

The strong convexity of Ψ implies that Problem (1) has a unique minimizer $x^\star$ [35, Corollary 11.16]. Examples of the strongly convex function Ψ include:

- l_2*-regularization*: $\Psi(x) = (\lambda/2)\|x\|_2^2$ with $\lambda > 0$.
- *Elastic net regularization*: $\Psi(x) = \lambda_1\|x\|_1 + (\lambda_2/2)\|x\|_2^2$ with $\lambda_1 > 0$ and $\lambda_2 > 0$.

In order to derive the convergence rate of Algorithm 1 for solving Problem (1) with a strongly convex regularization term, we need to assume that the Bregman distance function $D(x,y)$ used in the algorithm satisfies the next assumption.

Assumption 6 (Quadratic growth condition): For all $x, y \in$ dom Ψ, we have

$$D_\omega(x,y) \leq \frac{Q}{2}\|x-y\|^2$$

with $Q \geq \mu_\omega$.

For example, if $\omega(x) = \frac{1}{2}\|x\|_2^2$, then $D_\omega(x,y) = \frac{1}{2}\|x-y\|_2^2$ and $Q = 1$. Note that Assumption 6 will automatically hold when the distance generating function ω has Lipschitz continuous gradient with a constant Q [30].

The associated convergence result now reads as follows.

Theorem 3: *Suppose that the regularization function* Ψ *is* μ_Ψ*-strongly convex and that Assumptions 2–6 hold. If* $\{\gamma(k)\}_{k\in\mathbb{N}_0}$ *is set to* $\gamma(k)^{-1} = 2L(\tau_{\max}+1)^2 + \beta(k)$ *with*

$$\beta(k) = \frac{\mu_\Psi}{3Q}\big(k+\tau_{\max}+1\big),$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2016.2525015, IEEE Transactions on Automatic Control

then for $T \in \mathbb{N}$, the iterates produced by Algorithm 1 satisfies

$$\mathbb{E}\big[\|x(T) - x^\star\|^2\big] \leq \frac{2\left(\frac{6LQ}{\mu_\Psi} + 1\right)^2 (\tau_{\max}+1)^4}{(T+1)^2} D_\omega\big(x(0), x^\star\big) + \frac{18c\sigma^2 Q^2}{b\mu_\Psi^2 (T+1)}.$$

Proof: See Appendix C. ∎

An interesting point regarding Theorem 3 is that the maximum delay bound $\tau_{\max}$ can be as large as $\mathcal{O}(T^{1/4})$ without affecting the asymptotic convergence rate of Algorithm 1. In this case, our asynchronous mini-batch algorithm converges asymptotically at a rate of $\mathcal{O}(1/T)$, which matches the best known rate achievable for strongly convex stochastic problems in a serial setting [28]–[30].

D. Running-time Comparisons

Having derived the convergence rates for convex and strongly convex composite stochastic problems, we now explicitly compare the running times of the serial mini-batch algorithm ($p = 1$ and $\tau_{\max} = 0$) and the asynchronous mini-batch algorithm ($p > 1$ and $\tau_{\max} > 0$). We define a *time-unit* to be the time it takes a single processor to sample ξ from $\mathcal{P}$ and evaluate $\nabla_x F(x, \xi)$; a worker thus needs b time-units to process a batch of b samples. We ignore the time required to update current x in step (4) since this step usually can be done very efficiently and requires negligible time compared to computing the averaged gradient, especially when b is large [27]. Let N_t be the number of time-units allocated to each algorithm. Since the serial algorithm uses b samples to compute an averaged gradient and execute an update, it will be able to complete N_t/b iterations in N_t time-units. In the asynchronous algorithm, p processors concurrently and simultaneously compute stochastic averaged gradients, so the master will receive one averaged gradient vector every b/p time-units. It follows that in N_t time-units, the serial and asynchronous mini-batch algorithms perform N_t/b and pN_t/b iterations, respectively. Substituting these iteration counts into the guaranteed bounds provided by Theorems 2 and 3 together with assuming that $\tau_{\max}$ is roughly proportional to p [24]–[26], we can derive upper bounds on the expected optimization accuracy of each algorithm after N_t time-units, cf. Table II. We can see that the if the number of processors is suitably chosen, then the asynchronous mini-bach algorithm enjoys asymptotically faster convergence times for regularized stochastic optimization problems.

TABLE II: Upper bounds on $\mathbb{E}[\phi(x_{\text{ave}})] - \phi^\star$ for convex problems and $\mathbb{E}[\|x - x^\star\|^2]$ for strongly convex problems after N_t time-units.

Convergence rate for	Serial Algorithm	Asynchronous algorithm
convex ϕ	$\mathcal{O}\left(\frac{b}{N_t} + \frac{\sigma}{\sqrt{N_t}}\right)$	$\mathcal{O}\left(\frac{bp}{N_t} + \frac{\sigma}{\sqrt{pN_t}}\right)$
strongly convex ϕ	$\mathcal{O}\left(\frac{b^2}{N_t^2} + \frac{\sigma^2}{N_t}\right)$	$\mathcal{O}\left(\frac{b^2p^2}{N_t^2} + \frac{\sigma^2}{pN_t}\right)$

V. Experimental Results

We have developed a complete master-worker implementation of Algorithm 1 in C++ using the Massage Passing Interface (MPI) libraries OpenMPI [36]. Although we argued in Section IV that the Algorithm can be implemented using atomic operations on shared-memory computing architectures, we have chosen the MPI implementation due to its flexibility in scaling the problem to distributed-memory environments.

We provide extensive empirical results to show how the algorithm parameters can be selected based on our theoretical convergence analyses (Section V-A), how Algorithm 1 performs on convex (Section V-B) and strongly convex (Section V-C) stochastic optimization problems, how the performance of our asynchronous algorithm compares to that of the synchronous version (Section V-D), and how regularization parameters affect the convergence rate (Section V-E). To this end, we use two different datasets: `rcv1-v2` [37] and `Epsilon` [38]. The first one, `rcv1-v2`, is the corrected version of Reuters' Text Categorization Test Collection, which consists of $J \approx 800000$ documents with $n \approx 50000$ sparse unique stemmed tokens spanning 103 topics. Out of these topics, we decide to sort out all sports, government and disaster related documents. The second one, `Epsilon`, is a dense dataset consisting of $J = 500000$ samples with $n = 2000$ features. This dataset is already divided into two classes. We apply our code for Algorithm 1 to the following regularized logistic regression problem on the datasets:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \mathbb{E}_{\left(\xi_j^{(1)}, \xi_j^{(2)}\right)}\left[\log\left(1 + \exp\left(-\xi_j^{(2)}\langle \xi_j^{(1)}, x\rangle\right)\right)\right] + \lambda_1\|x\|_1 + \frac{\lambda_2}{2}\|x\|_2^2 + I_C(x)\,. \tag{8}$$

Here, $\xi_j^{(1)} \in \mathbb{R}^n$, $j = 1, \ldots, J$, represents a feature vector and $\xi_j^{(2)} \in \{-1, 1\}$ denotes its associated label, λ_1 and λ_2 are two regularization parameters, and $C = \{x \in \mathbb{R}^n : \|x\|_2 \leq R\}$. The label $\xi_j^{(2)}$ indicates whether a selected sample falls into the desired category, or not. For example, in `rcv1-v2`, $\xi_j^{(2)} = 1$ if the sampled document is about sports, government or disasters, and $\xi_j^{(2)} = -1$ otherwise. We use the distance generating function $\omega(x) = \frac{1}{2}\|x\|_2^2$ in all experiments.

A. Algorithm Parameter Selection

To demonstrate how the algorithm parameters can be selected based on our theoretical convergence analyses and the problem data, we choose to solve Problem (8) with $\lambda_1 = 0.01$, $\lambda_2 = 0$ and $R = 10$ on the `Epsilon` dataset. Our goal is to achieve an error of $\epsilon = 0.3$ after $T = 2500$ iterations by running the algorithm on $p = 4$ workers, *i.e.*, $\phi\big(x_{\text{ave}}(2500)\big) - \phi^\star \leq 0.3$. Since the feasible set is compact, Algorithm 1 can be implemented with time-varying step-sizes. By Theorem 2, we should find the batch-size b such that

$$\frac{LR^2(\tau_{\max}+1)^2}{T} + \frac{2\sigma R\sqrt{c}}{\sqrt{bT}} \leq \epsilon\,.$$

Using the problem data $L = 0.25$ and $\sigma = 1.0$ together with $c = 1$, one can verify that b should be at least 64. In our master-worker implementation, we observe that choosing a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2016.2525015, IEEE Transactions on Automatic Control

mini-batch size of $b = 100$ balances the communication and computations times, resulting in a good overall performance. In Figure 2, we present the actual function value attained by the iterates of the algorithm together with the theoretical upper bound derived in Theorem 2. As can be observed, the objective function value converges within the desired tolerance of the optimum, and the theoretical upper bound is clearly valid.

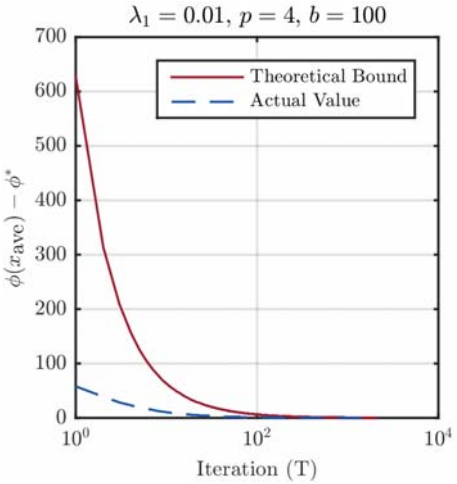

Fig. 2: Convergence of the objective function value in Problem (8) evaluated at the Cesáro average of the iterates to $\epsilon = 0.3$ neighborhood of the optimum function value when $\lambda_1 = 0.01$ and $\lambda_2 = 0$.

B. Asynchronous Algorithm on Convex Problems

To evaluate the performance of our asynchronous algorithm on convex stochastic problems for both sparse and dense datasets, we set $\lambda_1 = 0.01$, $\lambda_2 = 0$ and $R = 100$ in the objective function given in (8). As the feasible set is bounded, we implement Algorithm 1 with the time-varying step-size given in Theorem 2. This time, we use a batch size of $b = 1000$ samples over the total sample size of $J = 500000$. For relative speedup comparison purposes, we run the algorithm for 10 epochs on $p = 1, 2, 4, 6, 8, 10$ workers. We present the wall-clock times (running-time of the algorithm) to achieve the true optimum function value on the `Epsilon` dataset for different number of workers in Figure 3 (left). We see that in terms of wall-clock time, asynchronous updating by multiple workers offers a clear improvement over serial optimization ($p = 1$).

C. Asynchronous Algorithm on Strongly Convex Problems

This time, we set $R = +\infty$ (unconstrained minimization) and the l_2-regularization parameter in Problem (8) to $\lambda_2 = 0.001$ while keeping other variables the same as in Section V-B. We run the same experiments with the step-size given in Theorem 3. We present the wall-clock times to reach the unique optimizer for different number of workers in Figure 3 (right).

In Table III, we summarize the relative speedup of the algorithm achieved during our experiments in Sections V-B and V-C with respect to the number of workers used. The relative speedup of the algorithm on p workers is defined as $S(p) = t_1/t_p$, where t_1 and t_p are the time it takes to run the corresponding algorithm (to ϵ-accuracy) on 1 and p workers, respectively. Using this definition, we observe super-linear speedups in our experiments. Super-linear speedups are common in parallel computations and are often due to caching effects, especially when dataset size of an application is so large that it does not fit in the cache hierarchy of one single processors [39]. One way to eliminate the effect of such caching effects when presenting the results is to scale speedups with respect to that of the smallest number of processing units which result in a super-linear speedup. For this reason, Table III presents speed-up values relative to the two-worker case, *i.e.*, $S(p) = t_2/t_p$. The table confirms a near-linear relative speedup, consistent with our theoretical results, when a relatively small number of workers is used. However, as the number of workers increases, the relative speedup starts saturating due to the communication overhead at the master side. Speedup values are averaged over 10 Monte Carlo simulations.

TABLE III: Relative speedup of the asynchronous algorithm for convex ($\lambda_2 = 0$) and strongly convex ($\lambda_2 = 0.001$) stochastic objective functions, all with $\lambda_1 = 0.01$.

	$p =$	2	4	6	8	10
`Epsilon`	Convex	1	1.88	2.55	2.70	3.23
	Strongly Convex	1	1.82	2.52	3.08	3.69
`rcv1-v2`	Convex	1	1.60	2.32	2.92	3.53
	Strongly Convex	1	1.52	2.29	2.78	3.37
	Ideal	**1**	**2**	**3**	**4**	**5**

D. Comparison to Synchronous Algorithm

We now compare the performance of our asynchronous algorithm to that of the synchronous version. The synchronous version of Algorithm 1 can be implemented in the master-worker setting by forcing the master to wait for all the workers to return their stochastic gradients before updating the decision vector and sending it to each of the workers. We run the synchronous version of our algorithm on the `rcv1-v2` dataset with the same settings as in Section V-B. Figure 4 shows the convergence time of the serial, synchronous and asynchronous implementations of the algorithm. We observe that although the synchronous mini-batch algorithm can benefit from parallelization, asynchronous updates yield significant additional speedups.

E. Effects of Regularization Parameters on Convergence Rate

Finally, we run experiments to demonstrate the effect of the regularization parameter λ_2 on the convergence of our algorithm; see Figure 5. The results are obtained for $p = 4$ workers on the `rcv1-v2` dataset with the same settings as in Section V-C, and averaged over 10 Monte Carlo simulations.

As can be observed from the figure, increasing λ_2 results in a faster convergence time in the algorithm. This observation is consistent with our theoretical analysis, since modulus of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2016.2525015, IEEE Transactions on Automatic Control

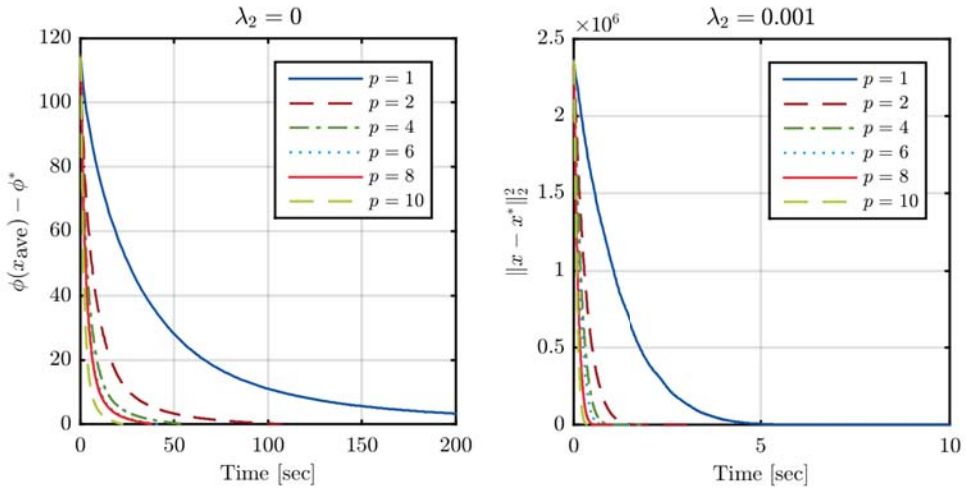

Fig. 3: (Left) Convergence of the objective function value in Problem (8) evaluated at the Cesáro average of the iterates to the optimum function value when $\lambda_2 = 0.0$, and, (right) convergence of the iterates to the optimizer of the same problem when $\lambda_2 = 0.001$, both for $\lambda_1 = 0.01$ and on the `Epsilon` dataset.

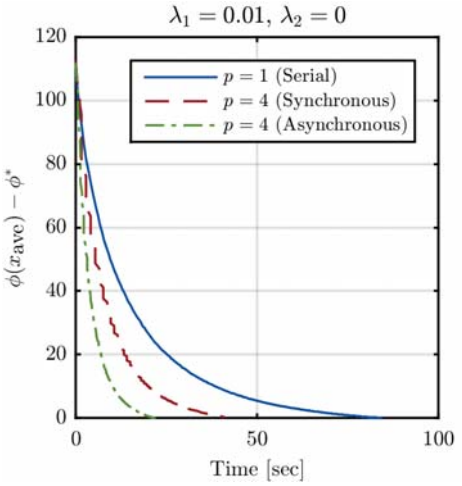

Fig. 4: Comparison of synchronous and asynchronous parallel mini-batch algorithms to the serial version. Relative speedups can be obtained with the synchronous algorithm, and even faster speedups can be obtained with the asynchronous version (two-fold in our experiment with this setting compared to the synchronous version).

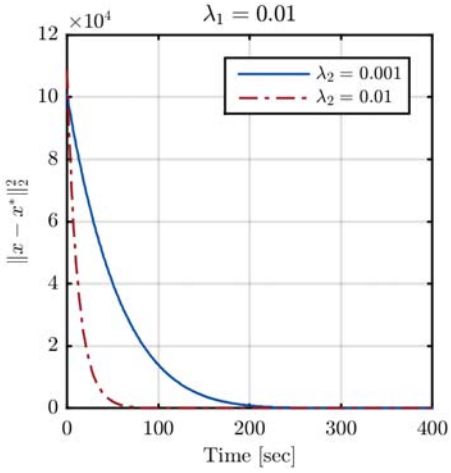

Fig. 5: Effect of the regularization parameter λ_2 on the convergence rate of Algorithm 1.

strong convexity μ_Ψ in Theorem 3 corresponds to λ_2 in Problem (8).

VI. Conclusions and Future Directions

We have proposed an asynchronous mini-batch algorithm that exploits multiple processors to solve regularized stochastic optimization problems with smooth loss functions. We have established that for closed and convex feasible sets, the iteration complexity of the algorithm with constant step-sizes is asymptotically $\mathcal{O}(1/\epsilon^2)$. For compact feasible sets, we have proved that the running average of the iterates generated by our algorithm with time-varying step-sizes converges to the optimum at a rate $\mathcal{O}(1/\sqrt{T})$. When the regularization function is strongly convex and the feasible set is closed and convex, the algorithm achieves the rate of the order $\mathcal{O}(1/T)$. We have shown that the penalty in convergence rate of the algorithm due to asynchrony is asymptotically negligible and a near-linear speedup in the number of processors can be expected. Our computational experience confirmed the theory.

We conclude with some open issues for future work:

1. *Accelerated asynchronous methods*: For general convex regularization functions, the convergence rate of our asynchronous algorithm is

$$\mathcal{O}\left(\frac{L(\tau_{\max}+1)^2}{T} + \frac{\sigma}{\sqrt{T}}\right).$$

The first term is related to the smooth component in the objective function and the existence of time-delays in gradient computations while the second term is related to the variance in stochastic gradients. As mentioned in Section III, the accelerated stochastic approximation method proposed in [6] reduces the impact of the smooth component significantly in the absence of asynchrony and achieves the rate

$$\mathcal{O}\left(\frac{L}{T^2} + \frac{\sigma}{\sqrt{T}}\right).$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2016.2525015, IEEE Transactions on Automatic Control

Hence, an interesting question is whether an asynchronous version of this method decreases or increases the effects of time-delays on the convergence rate. Answering this question is, however, challenging and nontrivial, since the convergence analysis of accelerated first-order methods even in a deterministic and serial setting is much more involved than that of non-accelerated methods [40].

2. *Non-i.i.d. sampling*: In order to establish our results, we assume that the stochastic oracle can generate i.i.d. samples from the distribution over which we optimize. This assumption is commonly used in the analysis for stochastic optimization algorithms, for example, [5]–[10], [19], [27]–[29], [41]–[43]. Recently, stochastic gradient methods for nonsmooth stochastic optimization was developed for situations in which there is no access to i.i.d. samples from the desired distribution [44]. Under reasonable assumptions on the ergodicity of the stochastic process that generates the samples, [44] obtained the convergence rate for serial mirror descent methods. It would be very interesting to extend this result to regularized stochastic optimization with smooth objective functions and investigate the convergence of asynchronous mini-batch algorithms when the random samples are dependent.

APPENDIX

In this section, we prove the main results of the paper, namely, Theorems 1–3. We first state three key lemmas which are instrumental in our argument.

Lemma 1: Suppose Assumptions 1–5 hold. Then, the iterates $\{x(k)\}_{k\in\mathbb{N}_0}$ generated by Algorithm 1 satisfy

$$\begin{aligned}\phi\big(x(k+1)\big)-\phi^\star+\frac{1}{\gamma(k)}D_\omega\big(x(k+1),x^\star\big)&\\ \leq\frac{1}{2\eta(k)}\big\|e\big(d(k)\big)\big\|_*^2&\\ +\big\langle e\big(d(k)\big),x(k)-x^\star\big\rangle+\frac{1}{\gamma(k)}D_\omega\big(x(k),x^\star\big)&\\ +\frac{L(\tau_{\max}+1)}{2}\sum_{j=0}^{\tau_{\max}}\big\|x(k-j)-x(k-j+1)\big\|^2&\\ -\frac{1}{2}\left(\frac{1}{\gamma(k)}-\eta(k)\right)\|x(k+1)-x(k)\|^2&\\ -\frac{\mu_\Psi}{2}\big\|x(k+1)-x^\star)\big\|^2,&\end{aligned}\tag{9}$$

where $x^\star\in X^\star$, $\{\eta(k)\}$ is a sequence of strictly positive numbers, and $e(k):=\nabla f(x(k))-g_{\text{ave}}(k)$ is the error in the gradient estimate.

Proof: We start with the first-order optimality condition for the point $x(k+1)$ in the minimization problem (4): there exists subgradient $s(k+1)\in\partial\Psi(x(k+1))$ such that for all $z\in\operatorname{dom}\Psi$, we have

$$\begin{aligned}0\leq\Big\langle g_{\text{ave}}\big(d(k)\big)+s(k+1)&\\ +\frac{1}{\gamma(k)}\nabla_{(2)}D_\omega\big(x(k),x(k+1)\big),z-x(k+1)\Big\rangle,&\end{aligned}$$

where $\nabla_{(2)}D_\omega(\cdot,\cdot)$ denotes the partial derivative of the Bregman distance function with respect to the second variable. Plugging the following equality

$$\nabla_{(2)}D_\omega\big(x(k),x(k+1)\big)=\nabla\omega\big(x(k+1)\big)-\nabla\omega\big(x(k)\big),$$

into the previous inequality and re-arranging terms gives

$$\begin{aligned}\frac{1}{\gamma(k)}\Big\langle\nabla\omega\big(x(k)\big)-\nabla\omega\big(x(k+1)\big),z-x(k+1)\Big\rangle&\\ \leq\Big\langle g_{\text{ave}}\big(d(k)\big)+s(k+1),z-x(k+1)\Big\rangle&\\ =\Big\langle g_{\text{ave}}\big(d(k)\big),z-x(k+1)\Big\rangle+\Big\langle s(k+1),z-x(k+1)\Big\rangle&\\ \leq\Big\langle g_{\text{ave}}\big(d(k)\big),z-x(k+1)\Big\rangle+\Psi(z)-\Psi(x(k+1))&\\ -\frac{\mu_\Psi}{2}\big\|z-x(k+1)\big\|^2,&\end{aligned}\tag{10}$$

where the last inequality follows from the (strong) convexity of Ψ. Using the well-known *three point identity* of the Bregman distance function [45], we have

$$\begin{aligned}\Big\langle\nabla\omega\big(x(k)\big)-\nabla\omega\big(x(k+1)\big),z-x(k+1)\Big\rangle=&\\ D_\omega\big(x(k),x(k+1)\big)-D_\omega\big(x(k),z\big)+D_\omega\big(x(k+1),z\big).&\end{aligned}$$

Substituting the preceding equality into (10) and re-arranging terms result in

$$\begin{aligned}\Psi(x(k+1))-\Psi(z)+\frac{1}{\gamma(k)}D_\omega\big(x(k+1),z\big)&\\ \leq\Big\langle g_{\text{ave}}\big(d(k)\big),z-x(k+1)\Big\rangle+\frac{1}{\gamma(k)}D_\omega\big(x(k),z\big)&\\ -\frac{1}{\gamma(k)}D_\omega\big(x(k),x(k+1)\big)-\frac{\mu_\Psi}{2}\big\|z-x(k+1)\big\|^2.&\end{aligned}$$

Since the distance generating function $\omega(x)$ is 1-strongly convex, we then have

$$\begin{aligned}\Psi(x(k+1))-\Psi(z)+\frac{1}{\gamma(k)}D_\omega\big(x(k+1),z\big)&\\ \leq\Big\langle g_{\text{ave}}\big(d(k)\big),z-x(k+1)\Big\rangle+\frac{1}{\gamma(k)}D_\omega\big(x(k),z\big)&\\ -\frac{1}{2\gamma(k)}\|x(k+1)-x(k)\|^2-\frac{\mu_\Psi}{2}\big\|z-x(k+1)\big\|^2.&\end{aligned}\tag{11}$$

According to Assumption 2, $\nabla F(x,\xi)$ and, hence, $\nabla f(x)$ are Lipschitz continuous with the constant L. By using the L-Lipschitz continuity of ∇f and then the convexity of f, we have

$$\begin{aligned}f(x(k+1))&\leq f(x(d(k)))+\langle\nabla f(x(d(k))),x(k+1)-x(d(k))\rangle\\ &\quad+\frac{L}{2}\|x(k+1)-x(d(k))\|^2\\ &\leq f\big(z\big)+\big\langle\nabla f\big(x(d(k))\big),x(k+1)-z\big\rangle\\ &\quad+\frac{L}{2}\big\|x(k+1)-x(d(k))\big\|^2,\end{aligned}\tag{12}$$

for any $z\in\operatorname{dom}\Psi$. Combining inequalities (11) and (12), and recalling that $\phi(x)=f(x)+\Psi(x)$, we obtain

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2016.2525015, IEEE Transactions on Automatic Control

$$\begin{aligned}
&\phi(x(k+1)) - \phi(z) + \frac{1}{\gamma(k)} D_\omega\big(x(k+1), z\big) \\
&\quad \le \big\langle \nabla f\big(x(d(k))\big) - g_{\text{ave}}\big(d(k)\big), x(k+1) - z\big\rangle + \frac{1}{\gamma(k)} D_\omega\big(x(k), z\big) \\
&\qquad - \frac{1}{2\gamma(k)}\|x(k+1) - x(k)\|^2 - \frac{\mu_\Psi}{2}\big\|z - x(k+1)\big\|^2 \\
&\qquad\qquad + \frac{L}{2}\|x(k+1) - x(d(k))\|^2.
\end{aligned}$$

We now rewrite the above inequality in terms of the error $e(d(k)) = \nabla f(x(d(k))) - g_{\text{ave}}(d(k))$ as follows:

$$\begin{aligned}
&\phi(x(k+1)) - \phi(z) + \frac{1}{\gamma(k)} D_\omega\big(x(k+1), z\big) \\
&\quad \le \big\langle e\big(d(k)\big), x(k+1) - z\big\rangle + \frac{1}{\gamma(k)} D_\omega\big(x(k), z\big) \\
&\qquad - \frac{1}{2\gamma(k)}\|x(k+1) - x(k)\|^2 - \frac{\mu_\Psi}{2}\big\|z - x(k+1)\big\|^2 \\
&\qquad\qquad + \frac{L}{2}\|x(k+1) - x(d(k))\|^2 \\
&\quad = \underbrace{\big\langle e\big(d(k)\big), x(k+1) - x(k)\big\rangle}_{U_1} \\
&\qquad + \big\langle e\big(d(k)\big), x(k) - z\big\rangle + \frac{1}{\gamma(k)} D_\omega\big(x(k), z\big) \\
&\qquad - \frac{1}{2\gamma(k)}\|x(k+1) - x(k)\|^2 - \frac{\mu_\Psi}{2}\big\|z - x(k+1)\big\|^2 \\
&\qquad\qquad + \frac{L}{2}\underbrace{\|x(k+1) - x(d(k))\|^2}_{U_2}. \qquad (13)
\end{aligned}$$

We will seek upper bounds on U_1 and U_2. Let $\{\eta(k)\}_{k\in\mathbb{N}_0}$ be a sequence of positive numbers. For U_1, we have

$$\begin{aligned}
U_1 &\le \left|\left\langle \frac{1}{\sqrt{\eta(k)}} e\big(d(k)\big), \sqrt{\eta(k)}\big(x(k+1) - x(k)\big)\right\rangle\right| \\
&\le \frac{1}{2\eta(k)}\big\|e\big(d(k)\big)\big\|_*^2 + \frac{\eta(k)}{2}\big\|x(k+1) - x(k)\big\|^2, \qquad (14)
\end{aligned}$$

where the second inequality follows from Fenchel's inequality applied to the conjugate pair $\frac{1}{2}\|\cdot\|^2$ and $\frac{1}{2}\|\cdot\|_*^2$, *i.e.*,

$$\big|\langle a, b\rangle\big| \le \frac{1}{2}\big\|a\big\|_*^2 + \frac{1}{2}\big\|b\big\|^2.$$

We turn to U_2. It follows from definition $\tau(k) = k - d(k)$ that

$$\begin{aligned}
U_2 &= \big(k - d(k) + 1\big)^2 \left\| \sum_{j=0}^{k-d(k)} \frac{x(k-j) - x(k-j+1)}{k - d(k) + 1} \right\|^2 \\
&= \big(\tau(k) + 1\big)^2 \left\| \sum_{j=0}^{\tau(k)} \frac{x(k-j) - x(k-j+1)}{\tau(k) + 1} \right\|^2.
\end{aligned}$$

Then, by the convexity of the norm $\|\cdot\|$, we conclude that

$$\begin{aligned}
U_2 &\le \big(\tau(k) + 1\big) \sum_{j=0}^{\tau(k)} \big\|x(k-j) - x(k-j+1)\big\|^2 \\
&\le \big(\tau_{\max} + 1\big) \sum_{j=0}^{\tau_{\max}} \big\|x(k-j) - x(k-j+1)\big\|^2, \qquad (15)
\end{aligned}$$

where the last inequality comes from our assumption that $\tau(k) \le \tau_{\max}$ for all $k \in \mathbb{N}_0$. Substituting inequalities (14) and (15) into the bound (13) and simplifying yield

$$\begin{aligned}
&\phi(x(k+1)) - \phi(z) + \frac{1}{\gamma(k)} D_\omega\big(x(k+1), z\big) \\
&\quad \le \frac{1}{2\eta(k)}\big\|e\big(d(k)\big)\big\|_*^2 \\
&\qquad + \big\langle e\big(d(k)\big), x(k) - z\big\rangle + \frac{1}{\gamma(k)} D_\omega\big(x(k), z\big) \\
&\qquad + \frac{L(\tau_{\max}+1)}{2} \sum_{j=0}^{\tau_{\max}} \big\|x(k-j) - x(k-j+1)\big\|^2 \\
&\qquad - \frac{1}{2}\left(\frac{1}{\gamma(k)} - \eta(k)\right)\|x(k+1) - x(k)\|^2 \\
&\qquad - \frac{\mu_\Psi}{2}\big\|z - x(k+1)\big\|^2.
\end{aligned}$$

Setting $z = x^\star$, where $x^\star \in X^\star$, completes the proof. ■

Lemma 2: Let Assumptions 1–5 hold. Assume also that $\{\gamma(k)\}_{k\in\mathbb{N}_0}$ is set to

$$\gamma(k) = \frac{1}{\eta(k) + L(\tau_{\max}+1)^2}, \quad k \in \mathbb{N}_0,$$

where $\eta(k)$ is positive for all k. Then, the iterates $\{x(k)\}_{k\in\mathbb{N}_0}$ produced by Algorithm 1 satisfy

$$\begin{aligned}
&\sum_{k=0}^{T-1} \big(\phi(x(k+1)) - \phi^\star\big) \\
&\quad \le \sum_{k=0}^{T-1} \frac{1}{2\eta(k)}\big\|e\big(d(k)\big)\big\|_*^2 \\
&\qquad + \sum_{k=0}^{T-1} \big\langle e\big(d(k)\big), x(k) - x^\star\big\rangle + \frac{1}{\gamma(0)} D_\omega\big(x(0), x^\star\big) \\
&\qquad + \sum_{k=0}^{T-1} \left(\frac{1}{\gamma(k+1)} - \frac{1}{\gamma(k)}\right) D_\omega\big(x(k+1), x^\star\big) \\
&\qquad - \frac{\mu_\Psi}{2} \sum_{k=0}^{T-1} \big\|x(k+1) - x^\star\big\|^2.
\end{aligned}$$

Proof: Applying Lemma 1 with

$$\eta(k) = \frac{1}{\gamma(k)} - L(\tau_{\max}+1)^2,$$

adding and subtracting $\gamma(k+1)^{-1} D_\omega\big(x(k+1), x^\star\big)$ to the left-hand side of (9), and re-arranging terms, we obtain

$$\begin{aligned}
&\phi(x(k+1)) - \phi^\star + \frac{1}{\gamma(k+1)} D_\omega\big(x(k+1), x^\star\big) \\
&\quad \le \frac{1}{2\eta(k)}\big\|e\big(d(k)\big)\big\|_*^2 \\
&\qquad + \big\langle e\big(d(k)\big), x(k) - x^\star\big\rangle + \frac{1}{\gamma(k)} D_\omega\big(x(k), x^\star\big) \\
&\qquad + \left(\frac{1}{\gamma(k+1)} - \frac{1}{\gamma(k)}\right) D_\omega\big(x(k+1), x^\star\big) \\
&\qquad + \frac{L(\tau_{\max}+1)}{2} \sum_{j=0}^{\tau_{\max}} \big\|x(k-j) - x(k-j+1)\big\|^2 \\
&\qquad - \frac{L(\tau_{\max}+1)^2}{2}\|x(k+1) - x(k)\|^2 \\
&\qquad - \frac{\mu_\Psi}{2}\big\|x(k+1) - x^\star\big\|^2.
\end{aligned}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2016.2525015, IEEE Transactions on Automatic Control

Summing the preceding inequality over $k = 0, \dots, T-1$, $T \in \mathbb{N}$, yields

$$\begin{aligned}
&\sum_{k=0}^{T-1} \big(\phi(x(k+1)) - \phi^\star\big) + \frac{1}{\gamma(T)} D_\omega\big(x(T), x^\star\big) \\
&\quad \le \sum_{k=0}^{T-1} \frac{1}{2\eta(k)} \big\|e\big(d(k)\big)\big\|_*^2 \\
&\qquad + \sum_{k=0}^{T-1} \big\langle e\big(d(k)\big), x(k) - x^\star \big\rangle + \frac{1}{\gamma(0)} D_\omega\big(x(0), x^\star\big) \\
&\qquad + \sum_{k=0}^{T-1} \left(\frac{1}{\gamma(k+1)} - \frac{1}{\gamma(k)}\right) D_\omega\big(x(k+1), x^\star\big) \\
&\qquad + \frac{L(\tau_{\max}+1)}{2} \sum_{k=0}^{T-1} \sum_{j=0}^{\tau_{\max}} \big\|x(k-j) - x(k-j+1)\big\|^2 \\
&\qquad - \frac{L(\tau_{\max}+1)^2}{2} \sum_{k=0}^{T-1} \|x(k+1) - x(k)\|^2 \\
&\qquad - \frac{\mu_\Psi}{2} \sum_{k=0}^{T-1} \big\|x(k+1) - x^\star\big\|^2 \\
&\quad \le \sum_{k=0}^{T-1} \frac{1}{2\eta(k)} \big\|e\big(d(k)\big)\big\|_*^2 \\
&\qquad + \sum_{k=0}^{T-1} \big\langle e\big(d(k)\big), x(k) - x^\star \big\rangle + \frac{1}{\gamma(0)} D_\omega\big(x(0), x^\star\big) \\
&\qquad + \sum_{k=0}^{T-1} \left(\frac{1}{\gamma(k+1)} - \frac{1}{\gamma(k)}\right) D_\omega\big(x(k+1), x^\star\big) \\
&\qquad - \frac{\mu_\Psi}{2} \sum_{k=0}^{T-1} \big\|x(k+1) - x^\star\big\|^2, \qquad (16)
\end{aligned}$$

where the second inequality used the facts

$$\begin{aligned}
&\sum_{k=0}^{T-1} \sum_{j=0}^{\tau_{\max}} \big\|x(k-j) - x(k-j+1)\big\|^2 \\
&\qquad = \sum_{j=0}^{\tau_{\max}} \sum_{k=-j}^{T-j-1} \big\|x(k) - x(k+1)\big\|^2 \\
&\qquad = \sum_{j=0}^{\tau_{\max}} \sum_{k=0}^{T-j-1} \big\|x(k) - x(k+1)\big\|^2 \\
&\qquad \le \sum_{j=0}^{\tau_{\max}} \sum_{k=0}^{T-1} \big\|x(k) - x(k+1)\big\|^2 \\
&\qquad \le (\tau_{\max}+1) \sum_{k=0}^{T-1} \big\|x(k) - x(k+1)\big\|^2,
\end{aligned}$$

and $x(k) = x(0)$ for all $k \le 0$. Dropping the second term on the left-hand side of (16) concludes the proof. ∎

Lemma 3: Let $\|\cdot\|$ be a norm over $\mathbb{R}^n$ and let $\|\cdot\|_\star$ be its dual norm. Let ω be a 1-strongly convex function with respect to $\|\cdot\|$ over $\mathbb{R}^n$. If $y_1, \dots, y_b \in \mathbb{R}^n$ are mean zero random variables drawn i.i.d. from a distribution $\mathcal{P}$, then

$$\mathbb{E}\left[\left\|\frac{1}{b}\sum_{i=1}^{b} y_i\right\|_*^2\right] \le \frac{c}{b^2} \sum_{i=1}^{b} \mathbb{E}\left[\|y_i\|_*^2\right],$$

where $c \in [1, b]$ is given by

$$c = \begin{cases} 1, & \text{if } \|\cdot\|_* = \|\cdot\|_2, \\ 2\max_{\|x\|=1} \omega(x), & \text{otherwise.} \end{cases}$$

Proof: The result follows from [46, Lemma B.2] and convexity of the norm $\|\cdot\|_*$. For further details, see [19, §4.1]. ∎

A. Proof of Theorem 1

Assume that the step-size $\{\gamma(k)\}_{k\in\mathbb{N}_0}$ is set to

$$\gamma(k) = \gamma = \frac{1}{\eta + L(\tau_{\max}+1)^2},$$

for some $\eta > 0$. It is clear that γ satisfies (5). Applying Lemma 2 with $\mu_\Psi = 0$, $\gamma(k) = \gamma$ and $\eta(k) = \eta$, we obtain

$$\begin{aligned}
&\sum_{k=0}^{T-1} \big(\phi(x(k+1)) - \phi^\star\big) \\
&\quad \le \sum_{k=0}^{T-1} \frac{1}{2\eta} \big\|e\big(d(k)\big)\big\|_*^2 \\
&\qquad + \sum_{k=0}^{T-1} \big\langle e\big(d(k)\big), x(k) - x^\star \big\rangle + \frac{D_\omega\big(x(0), x^\star\big)}{\gamma}, \qquad (17)
\end{aligned}$$

for $T \in \mathbb{N}$. Each $x(k)$, $k \in \mathbb{N}$, is a deterministic function of the history $\xi_{[k-1]} := \{\xi_i(t) \mid i = 1, \dots, b,\ t = 0, \dots, k-1\}$ but not of $\xi_i(k)$. Since $\nabla f(x) = \mathbb{E}_\xi[\nabla_x F(x, \xi)]$,

$$\mathbb{E}_{|\xi_{[k-1]}}\left[\big\langle e\big(d(k)\big), x(k) - x^\star\big\rangle\right] = 0.$$

Moreover, as ξ_i and ξ_j are independent whenever $i \ne j$, it follows from Lemma 3 that

$$\begin{aligned}
&\mathbb{E}\big[\|e\big(d(k)\big)\|_*^2\big] \\
&\quad = \mathbb{E}\left[\left\|\frac{1}{b}\sum_{i=1}^{b} \big(\nabla f(x(d(k))) - \nabla_x F(x(d(k)), \xi_i)\big)\right\|_*^2\right] \\
&\quad \le \frac{c}{b^2} \sum_{i=1}^{b} \mathbb{E}\left[\|\nabla f(x(d(k))) - \nabla_x F(x(d(k)), \xi_i)\|_*^2\right] \\
&\quad \le \frac{c\sigma^2}{b},
\end{aligned}$$

where the last inequality follows from Assumption 3. Taking expectation on both sides of (17) and using the above observations yield

$$\sum_{k=1}^{T} \big(\mathbb{E}[\phi(x(k))] - \phi^\star\big) \le \frac{c\sigma^2}{2\eta b} T + \frac{D_\omega\big(x(0), x^\star\big)}{\gamma}.$$

By the convexity of ϕ, we have

$$\phi\big(x_{\text{ave}}(T)\big) = \phi\left(\frac{1}{T}\sum_{k=1}^{T} x(k)\right) \le \frac{1}{T}\sum_{k=1}^{T} \phi\big(x(k)\big),$$

which implies that

$$\mathbb{E}\big[\phi(x_{\text{ave}}\big(T)\big)\big] - \phi^\star \le \frac{c\sigma^2}{2\eta b} + \frac{D_\omega\big(x(0), x^\star\big)}{\gamma T}.$$

Substituting $\eta = \gamma^{-1} - L(\tau_{\max}+1)^2$ into the above inequality proves the theorem.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2016.2525015, IEEE Transactions on Automatic Control

B. Proof of Theorem 2

Assume that the step-size $\{\gamma(k)\}_{k\in\mathbb{N}_0}$ is chosen such that $\gamma(k)^{-1} = L(\tau_{\max}+1)^2 + \alpha(k)$ where

$$\alpha(k) = \frac{\sigma\sqrt{c}\sqrt{k+1}}{R\sqrt{b}}.$$

Since $\gamma(k)$ is a non-increasing sequence, and $D_\omega(x,y) \le R^2$ for all $x, y \in$ dom Ψ, we have

$$\sum_{k=0}^{T-1}\left(\frac{1}{\gamma(k+1)} - \frac{1}{\gamma(k)}\right) D_\omega\big(x(k+1), x^\star\big) \le \left(\frac{1}{\gamma(T)} - \frac{1}{\gamma(0)}\right) R^2.$$

Applying Lemma 2 with $\mu_\Psi = 0$ and $\eta(k) = \alpha(k)$, taking expecation, and using Lemma 3 completely identically to the proof of Theorem 1, we then obtain

$$\sum_{k=1}^{T}\big(\mathbb{E}[\phi(x(k))] - \phi^\star\big) \le \frac{R^2}{\gamma(T)} + \frac{c\sigma^2}{2b}\sum_{k=0}^{T-1}\frac{1}{\alpha(k)}. \tag{18}$$

Viewing the sum as an lower-estimate of the integral of the function $y(t) = 1/\sqrt{t+1}$, one can verify that

$$\sum_{k=0}^{T-1}\frac{1}{\alpha(k)} = \sum_{k=0}^{T-1}\frac{1}{\widetilde{\alpha}\sqrt{k+1}} \le \frac{1}{\widetilde{\alpha}}\left(1 + \int_0^{T-1}\frac{dt}{\sqrt{t+1}}\right) \le \frac{2\sqrt{T}}{\widetilde{\alpha}},$$

where $\widetilde{\alpha} = (\sigma\sqrt{c})/(R\sqrt{b})$. Substituting this inequality into the bound (18), we obtain the claimed guaranteed bound.

C. Proof of Theorem 3

Assume that the step-size $\{\gamma(k)\}_{k\in\mathbb{N}_0}$ in Algorithm 1 is set to $\gamma(k)^{-1} = 2L(\tau_{\max}+1)^2 + \beta(k)$, with

$$\beta(k) = \frac{\mu_\Psi}{3Q}\big(k + \tau_{\max} + 1\big).$$

We first describe some important properties of $\gamma(k)$ relevant to our proof. Clearly, $\gamma(k)$ is non-increasing, *i.e.*,

$$\frac{1}{\gamma(k)} \le \frac{1}{\gamma(k+1)}, \tag{19}$$

for all $k \in \mathbb{N}_0$. Since $\gamma(0)^{-1} \le \gamma(k)^{-1}$, we have

$$2L(\tau_{\max}+1)^2 + \frac{\mu_\Psi\tau_{\max}}{3Q} \le \frac{1}{\gamma(k)}. \tag{20}$$

Moreover, one can easily verify that

$$\begin{aligned}\frac{1}{\gamma(k+1)^2} - \frac{1}{\gamma(k)^2} &= \frac{\mu_\Psi}{Q}\left(\frac{4L}{3}(\tau_{\max}+1)^2 + \frac{\mu_\Psi}{3Q}\left(\frac{2}{3}(k+\tau_{\max}) + 1\right)\right)\\ &\le \frac{\mu_\Psi}{Q}\left(2L(\tau_{\max}+1)^2 + \frac{\mu_\Psi}{3Q}\big(k+\tau_{\max}+1\big)\right)\\ &= \frac{\mu_\Psi}{Q}\frac{1}{\gamma(k)},\end{aligned}$$

which implies that

$$\frac{1}{\gamma(k+1)^2} \le \frac{1}{\gamma(k)}\left(\frac{1}{\gamma(k)} + \frac{\mu_\Psi}{Q}\right), \tag{21}$$

for all $k \in \mathbb{N}_0$. Finally, by the definition of $\gamma(k)$, we have

$$\begin{aligned}\frac{\gamma(k)}{\gamma(k+\tau_{\max})} &= 1 + \frac{\frac{\mu_\Psi}{3Q}\tau_{\max}}{2L(\tau_{\max}+1)^2 + \frac{\mu_\Psi}{3Q}\big(k+\tau_{\max}+1\big)}\\ &\le 1 + \frac{\mu_\Psi\tau_{\max}}{6LQ(\tau_{\max}+1)^2},\end{aligned}$$

and hence,

$$\frac{1}{\gamma(k+\tau_{\max})} \le \left(1 + \frac{\mu_\Psi\tau_{\max}}{6LQ(\tau_{\max}+1)^2}\right)\frac{1}{\gamma(k)}. \tag{22}$$

We are ready to prove Theorem 3. Applying Lemma 1 with

$$\eta(k) = \frac{1}{2\gamma(k)}, \quad k \in \mathbb{N}_0,$$

and using the fact

$$D_\omega\big(x(k+1), x^\star\big) \le \frac{Q}{2}\big\|x(k+1) - x^\star\big\|^2,$$

by Assumption 6, we obtain

$$\begin{aligned}&\phi\big(x(k+1)\big) - \phi^\star + \left(\frac{1}{\gamma(k)} + \frac{\mu_\Psi}{Q}\right) D_\omega\big(x(k+1), x^\star\big)\\ &\quad\le \gamma(k)\big\|e\big(d(k)\big)\big\|_*^2\\ &\qquad + \big\langle e\big(d(k)\big), x(k) - x^\star\big\rangle + \frac{1}{\gamma(k)}D_\omega\big(x(k), x^\star\big)\\ &\qquad + \frac{L(\tau_{\max}+1)}{2}\sum_{j=0}^{\tau_{\max}}\big\|x(k-j) - x(k-j+1)\big\|^2\\ &\qquad - \frac{1}{4\gamma(k)}\|x(k+1) - x(k)\|^2.\end{aligned}$$

Multiplying both sides of this relation by $1/\gamma(k)$, and then using (21), we have

$$\begin{aligned}&\frac{1}{\gamma(k)}\big(\phi\big(x(k+1)\big) - \phi^\star\big) + \frac{1}{\gamma(k+1)^2}D_\omega\big(x(k+1), x^\star\big)\\ &\quad\le \big\|e\big(d(k)\big)\big\|_*^2\\ &\qquad + \frac{1}{\gamma(k)}\big\langle e\big(d(k)\big), x(k) - x^\star\big\rangle + \frac{1}{\gamma(k)^2}D_\omega\big(x(k), x^\star\big)\\ &\qquad + \frac{L(\tau_{\max}+1)}{2\gamma(k)}\sum_{j=0}^{\tau_{\max}}\big\|x(k-j) - x(k-j+1)\big\|^2\\ &\qquad - \frac{1}{4\gamma(k)^2}\|x(k+1) - x(k)\|^2.\end{aligned}$$

Summing the above inequality from $k = 0$ to $k = T-1$, $T \in \mathbb{N}$, and dropping the first term on the left-hand side yield

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2016.2525015, IEEE Transactions on Automatic Control

$$\begin{aligned}\frac{1}{\gamma(T)^2}D_\omega\big(x(T),x^\star\big) &\le \sum_{k=0}^{T-1}\big\|e\big(d(k)\big)\big\|_*^2 \\ &+ \sum_{k=0}^{T-1}\frac{1}{\gamma(k)}\big\langle e\big(d(k)\big),x(k)-x^\star\big\rangle + \frac{1}{\gamma(0)^2}D_\omega\big(x(0),x^\star\big) \\ &+ \frac{L(\tau_{\max}+1)}{2}\sum_{k=0}^{T-1}\sum_{j=0}^{\tau_{\max}}\frac{1}{\gamma(k)}\big\|x(k-j)-x(k-j+1)\big\|^2 \\ &- \frac{1}{4}\sum_{k=0}^{T-1}\frac{1}{\gamma(k)^2}\|x(k+1)-x(k)\|^2. \qquad (23)\end{aligned}$$

What remains is to bound the third term on the right-hand side of (23). It follows from (19)–(22) that

$$\begin{aligned}&\frac{L(\tau_{\max}+1)}{2}\sum_{k=0}^{T-1}\sum_{j=0}^{\tau_{\max}}\frac{1}{\gamma(k)}\big\|x(k-j)-x(k-j+1)\big\|^2 \\ &= \frac{L(\tau_{\max}+1)}{2}\sum_{j=0}^{\tau_{\max}}\sum_{k=0}^{T-j-1}\frac{1}{\gamma(k+j)}\big\|x(k)-x(k+1)\big\|^2 \\ &\le \frac{L(\tau_{\max}+1)}{2}\sum_{j=0}^{\tau_{\max}}\sum_{k=0}^{T-1}\frac{1}{\gamma(k+j)}\big\|x(k)-x(k+1)\big\|^2 \\ &\overset{(19)}{\le} \frac{L(\tau_{\max}+1)}{2}\sum_{j=0}^{\tau_{\max}}\sum_{k=0}^{T-1}\frac{1}{\gamma(k+\tau_{\max})}\big\|x(k)-x(k+1)\big\|^2 \\ &= \frac{L(\tau_{\max}+1)^2}{2}\sum_{k=0}^{T-1}\frac{1}{\gamma(k+\tau_{\max})}\big\|x(k)-x(k+1)\big\|^2 \\ &\overset{(22)}{\le} \frac{2L(\tau_{\max}+1)^2+\frac{\mu_\Psi\tau_{\max}}{3Q}}{4}\sum_{k=0}^{T-1}\frac{1}{\gamma(k)}\big\|x(k)-x(k+1)\big\|^2 \\ &\overset{(20)}{\le} \frac{1}{4}\sum_{k=0}^{T-1}\frac{1}{\gamma(k)^2}\big\|x(k)-x(k+1)\big\|^2.\end{aligned}$$

Substituting the above inequality into (23), and then taking expectation on both sides (similarly to the proof of Theorems 1 and 2), we have

$$\frac{1}{\gamma(T)^2}\mathbb{E}\big[D_\omega\big(x(T),x^\star\big)\big] \le \frac{c\sigma^2T}{b} + \frac{1}{\gamma(0)^2}D_\omega\big(x(0),x^\star\big). \qquad (24)$$

According to Remark 1,

$$\frac{1}{2}\|x(T)-x^\star\|^2 \le D_\omega\big(x(T),x^\star\big).$$

Moreover, by the definition of $\gamma(k)$,

$$\frac{\mu_\Psi(T+1)}{3Q} \le \beta(T) \le \frac{1}{\gamma(T)}.$$

Combing these inequalities with the bound (24), we conclude

$$\begin{aligned}\mathbb{E}\big[\|x(T)-x^\star\|^2\big] &\le \frac{18c\sigma^2Q^2}{b\mu_\Psi^2(T+1)} \\ &+ \frac{2\left(\frac{6LQ}{\mu_\Psi}+1\right)^2(\tau_{\max}+1)^4}{(T+1)^2}D_\omega\big(x(0),x^\star\big).\end{aligned}$$

The proof is complete.

ACKNOWLEDGMENTS

We would like to thank the Associate Editor and the anonymous reviewers for suggestions that helped to improve the quality of the paper. The research was sponsored in part by the Swedish Foundation for Strategic Research (SSF) and the Swedish Science Foundation (VR).

REFERENCES

[1] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.
[2] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*. Springer, 2009.
[3] S. Shalev-Shwartz and A. Tewari, "Stochastic methods for l_1-regularized loss minimization," *Journal of Machine Learning Research*, vol. 12, pp. 1865–1892, 2011.
[4] H. Robbins and S. Monro, "A stochastic approximation method," *Annals of Mathematical Statistics*, pp. 400–407, 1951.
[5] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1574–1609, 2009.
[6] G. Lan, "An optimal method for stochastic composite optimization," *Mathematical Programming*, vol. 133, pp. 365–397, 2012.
[7] C. Hu, W. Pan, and J. T. Kwok, "Accelerated gradient methods for stochastic optimization and online learning," *Advances in Neural Information Processing Systems*, pp. 781–789, 2009.
[8] S. Ghadimi and G. Lan, "Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization I," *SIAM Journal on Optimization*, vol. 22, no. 4, pp. 1469–1492, 2012.
[9] L. Xiao, "Dual averaging method for regularized stochastic learning and online optimization," *Advances in Neural Information Processing Systems*, pp. 2116–2124, 2009.
[10] D. Needell, R. Ward, and N. Srebro, "Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm," *Advances in Neural Information Processing Systems*, pp. 1017–1025, 2014.
[11] I. Lobel and A. Ozdaglar, "Distributed subgradient methods for convex optimization over random networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1291–1306, 2011.
[12] B. Recht and C. Ré, "Parallel stochastic gradient algorithms for large-scale matrix completion," *Mathematical Programming Computation*, vol. 5, no. 2, pp. 201–226, 2013.
[13] M. Li, L. Zhou, Z. Yang, A. Li, F. Xia, D. G. Andersen, and A. Smola, "Parameter server for distributed machine learning," *Big Learning NIPS Workshop*, 2013.
[14] P. Bianchi and J. Jakubowicz, "Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization," *IEEE Transactions on Automatic Control*, vol. 58, no. 2, pp. 391–405, 2013.
[15] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan, "Communication-efficient distributed dual coordinate ascent," *Advances in Neural Information Processing Systems*, 2014.
[16] P. Richtárik and M. Takác, "Parallel coordinate descent methods for big data optimization," *Mathematical Programming*, pp. 1–52, 2015.
[17] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, and Q. V. Le, "Large scale distributed deep networks," *Advances in Neural Information Processing Systems*, pp. 1223–1231, 2012.
[18] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project Adam: Building an efficient and scalable deep learning training system," *Symposium on Operating Systems Design and Implementation*, 2014.
[19] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 165–202, 2012.
[20] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*. Prentice-Hall, 1989.
[21] A. Nedić, D. P. Bertsekas, and V. S. Borkar, "Distributed asynchronous incremental subgradient methods," *Studies in Computational Mathematics*, vol. 8, pp. 381–407, 2001.
[22] K. Tsianos and M. G. Rabbat, "Distributed dual averaging for convex optimization under communication delays," *IEEE American Control Conference (ACC)*, pp. 1067–1072, 2012.
[23] B. McMahan and M. Streeter, "Delay-tolerant algorithms for asynchronous distributed online learning," *Advances in Neural Information Processing Systems*, pp. 2915–2923, 2014.

[24] F. Niu, B. Recht, C. Ré, and S. J. Wright, "Hogwild!: A lock-free approach to parallelizing stochastic gradient descent." *Advances in Neural Information Processing Systems*, pp. 693–701, 2011.
[25] J. Liu, S. J. Wright, C. Ré, V. Bittorf, and S. Sridhar, "An asynchronous parallel stochastic coordinate descent algorithm," *International Conference on Machine Learning (ICML)*, pp. 469–477, 2014.
[26] J. Liu and S. J. Wright, "Asynchronous stochastic coordinate descent: Parallelism and convergence properties," *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 351–376, 2015.
[27] A. Agarwal and J. C. Duchi, "Distributed delayed stochastic optimization," *IEEE Conference on Decision and Control*, pp. 5451–5452, 2012.
[28] S. Ghadimi and G. Lan, "Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization II," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2061–2089, 2013.
[29] A. Rakhlin, O. Shamir, and K. Sridharan, "Making gradient descent optimal for strongly convex stochastic optimization," *International Conference on Machine Learning (ICML)*, pp. 449–456, 2012.
[30] A. Nedić and S. Lee, "On stochastic subgradient mirror-descent algorithm with weighted averaging," *SIAM Journal on Optimization*, vol. 24, no. 1, pp. 84–107, 2014.
[31] A. Beck and M. Teboulle, "Mirror descent and nonlinear projected subgradient methods for convex optimization," *Operations Research Letters*, vol. 31, no. 3, pp. 167–175, 2003.
[32] P. Tseng, "Approximation accuracy, gradient methods, and error bound for structured convex optimization," *Mathematical Programming*, vol. 125, no. 2, pp. 263–295, 2010.
[33] J. Duchi, S. Shalev-Shwartz, Y. Singer, and A. Tewari, "Composite objective mirror descent," *Conference on Learning Theory*, 2010.
[34] R. Rockafellar and R. Wets, "On the interchange of subdifferentiation and conditional expectation for convex functionals," *Journal of Probability and Stochastic Processes*, vol. 7, no. 3, pp. 173–182, 1982.
[35] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2011.
[36] The Open MPI Project. (2015) Open MPI: Open Source High Performance Computing. [Online]. Available: http://www.open-mpi.org/
[37] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *Journal of Machine Learning Research*, vol. 5, pp. 361–397, 2004.
[38] TU Berlin. (2015) Pascal large scale learning challenge. [Online]. Available: http://largescale.ml.tu-berlin.de/
[39] T. Rauber and G. Rünger, *Parallel programming: For multicore and cluster systems*. Springer Science & Business Media, 2013.
[40] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Springer, 2004.
[41] ——, "Primal-dual subgradient methods for convex problems," *Mathematical programming*, vol. 120, no. 1, pp. 221–259, 2009.
[42] X. Chen, Q. Lin, and J. Pena, "Optimal regularized dual averaging methods for stochastic optimization," *Advances in Neural Information Processing Systems*, pp. 395–403, 2012.
[43] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright, "Randomized smoothing for stochastic optimization," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 674–701, 2012.
[44] J. C. Duchi, A. Agarwal, M. Johansson, and M. I. Jordan, "Ergodic mirror descent," *SIAM Journal on Optimization*, vol. 22, no. 4, pp. 1549–1578, 2012.
[45] G. Chen and M. Teboulle, "Convergence analysis of a proximal-like minimization algorithm using Bregman functions," *SIAM Journal on Optimization*, vol. 3, no. 3, pp. 538–543, 1993.
[46] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan, "Better mini-batch algorithms via accelerated gradient methods," *Advances in Neural Information Processing Systems*, pp. 1647–1655, 2011.

Hamid Reza Feyzmahdavian received the B.Sc. and M.Sc. degrees in electrical engineering with specialization in automatic control from Sharif University of Technology, Tehran, Iran. Since 2011, he has been working toward the Ph.D. degree at the Department of Automatic Control, KTH Royal Institute of Technology, Stockholm, Sweden. His research interests include distributed optimization, machine learning, positive systems, and time-delay systems.

Arda Aytekin received his B.Sc. degrees in mechanical engineering (2009) and control engineering (2010) from Istanbul Technical University, Turkey. After obtaining his M.Sc. degree in mechanical engineering from Koc University, Istanbul, Turkey in 2013, he joined the Department of Automatic Control, KTH Royal Institute of Technology, Stockholm, Sweden as a Ph.D. student. His research interests include distributed optimization and its applications in large-scale machine learning.

Mikael Johansson received the M.Sc. and Ph.D. degrees in electrical engineering from the University of Lund, Sweden, in 1994 and 1999, respectively. He held postdoctoral positions at Stanford University and UC Berkeley before joining KTH in 2002, where he now serves as full professor. His research interests are in distributed optimization, wireless networking, and control. He has published one book and over one hundred papers, several of which are ISI-highly cited. He is an associate editor for IEEE Transactions on Control of Network Systems, has served on the editorial board of Automatica and on the program committee for conferences such as IEEE CDC, IEEE Infocom, ACM SenSys.