# Distributed Delayed Stochastic Optimization

**Alekh Agarwal** **John C. Duchi**
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720
{alekh,jduchi}@eecs.berkeley.edu

## Abstract

We analyze the convergence of gradient-based optimization algorithms whose updates depend on delayed stochastic gradient information. The main application of our results is to the development of distributed minimization algorithms where a master node performs parameter updates while worker nodes compute stochastic gradients based on local information in parallel, which may give rise to delays due to asynchrony. Our main contribution is to show that for smooth stochastic problems, the delays are asymptotically negligible. In application to distributed optimization, we show $n$-node architectures whose optimization error in stochastic problems—in spite of asynchronous delays—scales asymptotically as $\mathcal{O}(1/\sqrt{nT})$, which is known to be optimal even in the absence of delays.

## 1 Introduction

We focus on stochastic convex optimization problems of the form

$$\operatorname*{minimize}_{x \in \mathcal{X}} \; f(x) \quad \text{for} \quad f(x) := \mathbb{E}_P[F(x;\xi)] = \int_\Xi F(x;\xi) dP(\xi), \tag{1}$$

where $\mathcal{X} \subseteq \mathbb{R}^d$ is a closed convex set, $P$ is a probability distribution over $\Xi$, and $F(\cdot\,;\xi)$ is convex for all $\xi \in \Xi$, so that $f$ is convex. Classical stochastic gradient algorithms [18, 16] iteratively update a parameter $x(t) \in \mathcal{X}$ by sampling $\xi \sim P$, computing $g(t) = \nabla F(x(t);\xi)$, and performing the update $x(t+1) = \Pi_{\mathcal{X}}(x(t) - \alpha(t)g(t))$, where $\Pi_{\mathcal{X}}$ denotes projection onto the set $\mathcal{X}$ and $\alpha(t) \in \mathbb{R}$ is a stepsize. In this paper, we analyze asynchronous gradient methods, where instead of receiving current information $g(t)$, the procedure receives out of date gradients $g(t - \tau(t)) = \nabla F(x(t - \tau(t)), \xi)$, where $\tau(t)$ is the (potentially random) delay at time $t$. The central contribution of this paper is to develop algorithms that—under natural assumptions about the functions $F$ in the objective (1)—achieve asymptotically optimal convergence rates for stochastic convex optimization in spite of delays.

Our model of delayed gradient information is particularly relevant in distributed optimization scenarios, where a master maintains the parameters $x$ while workers compute stochastic gradients of the objective (1) using a local subset of the data. Master-worker architectures are natural for distributed computation, and other researchers have considered models similar to those in this paper [12, 10]. By allowing delayed and asynchronous updates, we can avoid synchronization issues that commonly handicap distributed systems.

Distributed optimization has been studied for several decades, tracing back at least to seminal work of Bertsekas and Tsitsiklis ([3, 19, 4]) on asynchronous computation and minimization of smooth functions where the parameter vector is distributed. More recent work has studied problems in which each processor or node $i$ in a network has a local function $f_i$, and the goal is to minimize the sum $f(x) = \frac{1}{n}\sum_{i=1}^n f_i(x)$ [12, 13, 17, 7]. Our work is closest to Nedić et al.'s asynchronous incremental subgradient method [12], who

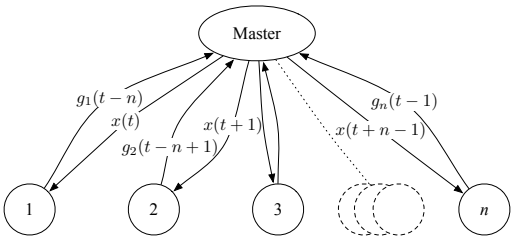



Figure 1: Cyclic delayed update architecture. Workers compute gradients in parallel, passing out-of-date (stochastic) gradients $g_i(t-\tau) = \nabla f_i(x(t-\tau))$ to master. Master responds with current parameters. Diagram shows parameters and gradients communicated between rounds $t$ and $t+n-1$.

analyze gradient projection steps taken using out-of-date gradients. See Figure 1 for an illustration. Nedić et al. prove that the procedure converges, and a slight extension of their results shows that the optimization error of the procedure after $T$ iterations is at most $\mathcal{O}(\sqrt{\tau/T})$, $\tau$ being the delay in gradients. Without delay, a centralized stochastic gradient algorithm attains convergence rate $\mathcal{O}(1/\sqrt{T})$. All the approaches mentioned above give slower convergence than this centralized rate in distributed settings, paying a penalty for data being split across a network; as Dekel et al. [5] note, one would expect that parallel computation actually speeds convergence. Langford et al. [10] also study asynchronous methods in the setup of stochastic optimization and attempt to remove the penalty for the delayed procedure under an additional smoothness assumption; however, their paper has a technical error (see the long version [2] for details). The main contributions of our paper are (1) to remove the delay penalty for smooth functions and (2) to demonstrate benefits in convergence rate by leveraging parallel computation even in spite of delays.

We build on results of Dekel et al. [5], who give reductions of stochastic optimization algorithms (e.g. [8, 9]) to show that for smooth objectives $f$, when $n$ processors compute stochastic gradients in parallel using a common parameter $x$ it is possible to achieve convergence rate $\mathcal{O}(1/\sqrt{Tn})$. The rate holds so long as most processors remain synchronized for most of the time [6]. We show similar results, but we analyze the effects of asynchronous gradient updates where all the nodes in the network can suffer delays, quantifying the impact of the delays. In application to distributed optimization, we show that under different network assumptions, we achieve convergence rates ranging from $\mathcal{O}(\min\{n^3/T, (n/T)^{2/3}\} + 1/\sqrt{Tn})$ to $\mathcal{O}(\min\{n/T, 1/T^{2/3}\} + 1/\sqrt{Tn})$, which is $\mathcal{O}(1/\sqrt{nT})$ asymptotically in $T$. The time necessary to achieve $\epsilon$-optimal solution to the problem (1) is asymptotically $\mathcal{O}(1/n\epsilon^2)$, a factor of *n–the size of the network–*better than a centralized procedure in spite of delay. Proofs of our results can be found in the long version of this paper [2].

**Notation** We denote a general norm by $\|\cdot\|$, and its associated dual norm $\|\cdot\|_*$ is defined as $\|z\|_* := \sup_{x:\|x\|\leq 1} \langle z, x\rangle$. The subdifferential set of a function $f$ at a point $x$ is $\partial f(x) := \left\{g \in \mathbb{R}^d \mid f(y) \geq f(x) + \langle g, y - x\rangle \text{ for all } y \in \operatorname{dom} f\right\}$. A function $f$ is $G$-Lipschitz w.r.t. the norm $\|\cdot\|$ on $\mathcal{X}$ if $\forall x, y \in \mathcal{X}$, $|f(x) - f(y)| \leq G\|x - y\|$, and $f$ is $L$-smooth on $\mathcal{X}$ if

$$\|\nabla f(x) - \nabla f(y)\|_* \leq L\|x - y\|, \quad \text{equivalently,} \quad f(y) \leq f(x) + \langle \nabla f(x), y - x\rangle + \frac{L}{2}\|x - y\|^2.$$

A convex function $h$ is $c$-strongly convex with respect to a norm $\|\cdot\|$ over $\mathcal{X}$ if

$$h(y) \geq h(x) + \langle g, y - x\rangle + \frac{c}{2}\|x - y\|^2 \quad \text{for all } x, y \in \mathcal{X} \text{ and } g \in \partial h(x). \tag{2}$$

## 2 Setup and Algorithms

To build intuition for the algorithms we analyze, we first describe the delay-free algorithm underlying our approach: the dual averaging algorithm of Nesterov [15].[1] The dual averaging algorithm is based on a strongly convex proximal function $\psi(x)$; we assume without loss that $\psi(x) \geq 0$ for all $x \in \mathcal{X}$ and (by scaling) that $\psi$ is 1-strongly convex.

[1]Essentially identical results to those we present here also hold for extensions of mirror descent [14], but we omit these for lack of space.

At time $t$, the algorithm updates a dual vector $z(t)$ and primal vector $x(t) \in \mathcal{X}$ using a subgradient $g(t) \in \partial F(x(t); \xi(t))$, where $\xi(t)$ is drawn i.i.d. according to $P$:

$$z(t+1) = z(t) + g(t) \quad \text{and} \quad x(t+1) = \operatorname*{argmin}_{x \in \mathcal{X}} \left\{ \langle z(t+1), x \rangle + \frac{1}{\alpha(t+1)} \psi(x) \right\}. \tag{3}$$

For the remainder of the paper, we will use the following three essentially standard assumptions [8, 9, 20] about the stochastic optimization problem (1).

**Assumption I** (Lipschitz Functions)**.** *For $P$-a.e. $\xi$, the function $F(\cdot\,; \xi)$ is convex. Moreover, for any $x \in \mathcal{X}$, and $v \in \partial F(x; \xi)$, $\mathbb{E}[\|v\|_*^2] \le G^2$.*

**Assumption II** (Smooth Functions)**.** *The expected function $f$ has $L$-Lipschitz continuous gradient, and for all $x \in \mathcal{X}$ the variance bound $\mathbb{E}[\|\nabla f(x) - \nabla F(x; \xi)\|_*^2] \le \sigma^2$ holds.*

**Assumption III** (Compactness)**.** *For all $x \in \mathcal{X}$, $\psi(x) \le R^2/2$.*

Several commonly used functions satisfy the above assumptions, for example:

(i) The *logistic loss*: $F(x; \xi) = \log[1+\exp(\langle x, \xi \rangle)]$. The objective $F$ satisfies Assumptions I and II so long as $\|\xi\|_*$ has finite second moment.

(ii) *Least squares*: $F(x; \xi) = (a - \langle x, b \rangle)^2$ where $\xi = (a, b)$ for $a \in \mathbb{R}^d$ and $b \in \mathbb{R}$, satisfies Assumptions I and II if $\mathcal{X}$ is compact and $\|\xi\|_*$ has finite fourth moment.

Under Assumption III, assumptions I and II imply finite-sample convergence rates for the update (3). Define the time averaged vector $\widehat{x}(T) := \frac{1}{T} \sum_{t=1}^{T} x(t+1)$. Under Assumption I, dual averaging satisfies $\mathbb{E}[f(\widehat{x}(T))] - f(x^*) = \mathcal{O}(RG/\sqrt{T})$ for the stepsize choice $\alpha(t) = R/(G\sqrt{t})$ [15, 20]. The result is sharp to constant factors [14, 1], but can be further improved using Assumption II. Building on work of Juditsky et al. [8] and Lan [9], Dekel et al. [5, Appendix A] show that the stepsize choice $\alpha(t)^{-1} = L + \sigma R \sqrt{t}$, yields the convergence rate

$$\mathbb{E}[f(\widehat{x}(T))] - f(x^*) = \mathcal{O}\left( \frac{LR^2}{T} + \frac{\sigma R}{\sqrt{T}} \right). \tag{4}$$

**Delayed Optimization Algorithms** We now turn to extending the dual averaging (3) update to the setting in which instead of receiving a current gradient $g(t)$ at time $t$, the procedure receives a gradient $g(t - \tau(t))$, that is, a stochastic gradient of the objective (1) computed at the point $x(t - \tau(t))$. Our analysis admits any sequence $\tau(t)$ of delays as long as the mapping $t \mapsto t - \tau(t)$ is one-to-one, and satisfies $\mathbb{E}[\tau(t)^2] \le B^2 < \infty$.

We consider the dual averaging algorithm with $g(t)$ replaced by $g(t - \tau(t))$:

$$z(t+1) = z(t) + g(t - \tau(t)) \quad \text{and} \quad x(t+1) = \operatorname*{argmin}_{x \in \mathcal{X}} \left\{ \langle z(t+1), x \rangle + \frac{1}{\alpha(t+1)} \psi(x) \right\}. \tag{5}$$

By combining the techniques Nedić et al. [12] developed with the convergence proofs of dual averaging [15], it is possible to show that so long as $\mathbb{E}[\tau(t)] \le B < \infty$ for all $t$, Assumptions I and III and the stepsize choice $\alpha(t) = \frac{R}{G\sqrt{Bt}}$ give $\mathbb{E}[f(\widehat{x}(T))] - f(x^*) = \mathcal{O}(RG\sqrt{B}/\sqrt{T})$. In the next section we show how to avoid the $\sqrt{B}$ penalty.

## 3 Convergence rates for delayed optimization of smooth functions

We now state and discuss the implications of two results for asynchronous stochastic gradient methods. Our first convergence result is for the update rule (5), while the second averages several stochastic subgradients for every update, each with a potentially different delay.

### 3.1 Simple delayed optimization

Our focus in this section is to remove the $\sqrt{B}$ penalty for the delayed update rule (5) using Assumption II, which arises for non-smooth optimization because subgradients can vary drastically even when measured at near points. We show that under the smoothness condition, the errors from delay become second order: the penalty is asymptotically negligible.

**Theorem 1.** *Let $x(t)$ be defined by the update (5). Define $\alpha(t)^{-1} = L + \eta(t)$, where $\eta(t) = \eta\sqrt{t}$ or $\eta(t) \equiv \eta\sqrt{T}$ for all $t$. The average $\widehat{x}(T) = \sum_{t=1}^{T} x(t+1)/T$ satisfies*

$$\mathbb{E}f(\widehat{x}(T)) - Tf(x^*) \le \frac{LR^2 + 6\tau GR}{T} + \frac{2\eta R^2}{\sqrt{T}} + \frac{\sigma^2}{\eta\sqrt{T}} + 4\frac{LG^2(\tau+1)^2 \log T}{\eta^2 T}.$$

We make a few remarks about the theorem. The $\log T$ factor on the last term is not present when using the fixed stepsize of $\eta\sqrt{T}$. Furthermore, though we omit it here for lack of space, the analysis also extends to random delays as long as $\mathbb{E}[\tau(t)^2] \le B^2$; see the long version [2] for details. Finally, based on Assumption II, we can set $\eta = \sigma/R$, which makes the rate asymptotically $\mathcal{O}(\sigma R/\sqrt{T})$, which is the same as the delay-free case so long as $\tau = o(T^{1/4})$. The take-home message from Theorem 1 is thus that the penalty in convergence rate due to the delay $\tau(t)$ is asymptotically negligible. In the next section, we show the implications of this result for robust distributed stochastic optimization algorithms.

### 3.2 Combinations of delays

In some scenarios—including distributed settings similar to those we discuss in the next section—the procedure has access not to only a single delayed gradient but to several stochastic gradients with different delays. To abstract away the essential parts of this situation, we assume that the procedure receives $n$ stochastic gradients $g_1, \ldots, g_n \in \mathbb{R}^d$, where each has a potentially different delay $\tau(i)$. Let $\lambda = (\lambda_i)_{i=1}^n$ be (an unspecified) vector in probability simplex. Then the procedure performs the following updates at time $t$:

$$z(t+1) = z(t) + \sum_{i=1}^{n} \lambda_i g_i(t - \tau(i)), \quad x(t+1) = \operatorname*{argmin}_{x \in \mathcal{X}} \left\{ \langle z(t+1), x \rangle + \frac{1}{\alpha(t+1)}\psi(x) \right\}. \quad (6)$$

The next theorem builds on the proof of Theorem 1.

**Theorem 2.** *Under Assumptions I–III, let $\alpha(t) = (L + \eta(t))^{-1}$ and $\eta(t) = \eta\sqrt{t}$ or $\eta(t) \equiv \eta\sqrt{T}$ for all $t$. The average $\widehat{x}(T) = \sum_{t=1}^{T} x(t+1)/T$ for the update sequence (6) satisfies*

$$f(\widehat{x}(T)) - f(x^*) \le \frac{2LR^2 + 4\sum_{i=1}^{n} \lambda_i \tau(i) GR}{T} + 6\frac{\sum_{i=1}^{n} \lambda_i LG^2(\tau(i)+1)^2 \log T}{\eta^2 T}$$
$$+ \frac{4\eta R^2}{\sqrt{T}} + \frac{1}{\eta\sqrt{T}} \mathbb{E} \Big\| \sum_{i=1}^{n} \lambda_i [\nabla f(x(t - \tau(i))) - g_i(t - \tau(i))] \Big\|_*^2.$$

We illustrate the consequences of Theorem 2 for distributed optimization in the next section.

## 4 Distributed Optimization

We now turn to what we see as the main purpose and application of the above results: developing robust and efficient algorithms for distributed stochastic optimization. Our main motivations here are machine learning applications where the data is so large that it cannot fit on a single computer. Examples of the form (1) include logistic or linear regression, as described respectively in Sec. 2(i) and (ii). We consider both stochastic and online/streaming scenarios for such problems. In the simplest setting, the distribution $P$ in the objective (1) is the empirical distribution over an observed dataset, that is, $f(x) = \frac{1}{N}\sum_{i=1}^{N} F(x; \xi_i)$. We divide the $N$ samples among $n$ workers so that each worker has an $N/n$-sized subset of data. In online learning applications, the distribution $P$ is the unknown distribution generating the data, and each worker receives a stream of independent data points $\xi \sim P$. Worker $i$ uses its subset of the data, or its stream, to compute $g_i \in \mathbb{R}^d$, an estimate of the gradient $\nabla f$ of the global $f$. We assume that $g_i$ is an unbiased estimate of $\nabla f(x)$, which is satisfied, for example, in the online setting or when each worker computes the gradient $g_i$ based on samples picked at random without replacement from its subset of the data.

The architectural assumptions we make are based off of master/worker topologies, but the convergence results in Section 3 allow us to give procedures robust to delay and asynchrony. The architectures build on the naïve scheme of having each worker simultaneously compute a stochastic gradient and send it to the master, which takes a gradient step on the

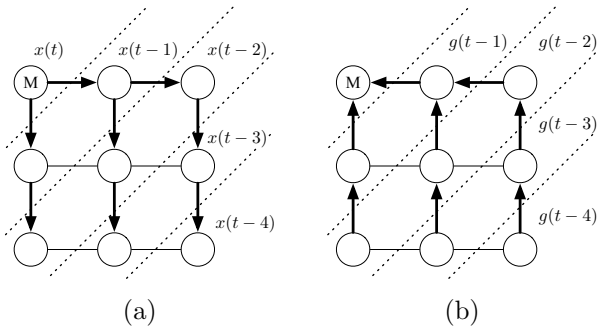



Figure 2: Master-worker averaging network. (a): parameters stored at different nodes at time $t$. A node at distance $d$ from master has the parameter $x(t-d)$. (b): gradients computed at different nodes. A node at distance $d$ from master computes gradient $g(t-d)$.

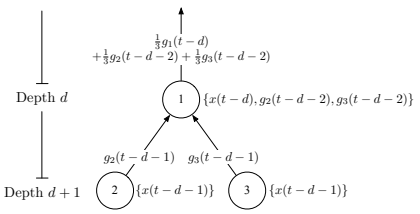



Figure 3: Communication of gradient information toward master node at time $t$ from node 1 at distance $d$ from master. Information stored at time $t$ by node $i$ in brackets to right of node.

averaged gradient. While the $n$ gradients are computed in parallel in the naïve scheme, accumulating and averaging $n$ gradients at the master takes $\Omega(n)$ time, offsetting the gains of parallelization, and the procedure is non-robust to laggard workers.

**Cyclic Delayed Architecture** This protocol is the delayed update algorithm mentioned in the introduction, and it computes $n$ stochastic gradients of $f(x)$ in parallel. Formally, worker $i$ has parameter $x(t-\tau)$ and computes $g_i(t-\tau) = \nabla F(x(t-\tau); \xi_i(t)) \in \mathbb{R}^d$, where $\xi_i(t)$ is a random variable sampled at worker $i$ from the distribution $P$. The master maintains a parameter vector $x \in \mathcal{X}$. At time $t$, the master receives $g_i(t-\tau)$ from some worker $i$, computes $x(t+1)$ and passes it back to worker $i$ only. Other workers do not see $x(t+1)$ and continue their gradient computations on stale parameter vectors. In the simplest case, each node suffers a delay of $\tau = n$, though our analysis applies to random delays as well. Recall Fig. 1 for a description of the process.

**Locally Averaged Delayed Architecture** At a high level, the protocol we now describe combines the delayed updates of the cyclic delayed architecture with averaging techniques of previous work [13, 7]. We assume a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of $n$ nodes (workers) and $\mathcal{E}$ are the edges between the nodes. We select one of the nodes as the master, which maintains the parameter vector $x(t) \in \mathcal{X}$ over time.

The algorithm works via a series of multicasting and aggregation steps on a spanning tree rooted at the master node. In the broadcast phase, the master sends its current parameter vector $x(t)$ to its immediate neighbors. Simultaneously, every other node broadcasts its current parameter vector (which, for a depth $d$ node, is $x(t-d)$) to its children in the spanning tree. See Fig. 2(a). Every worker computes its local gradient at its new parameter (see Fig. 2(b)). The communication then proceeds from leaves toward the root. The leaf nodes communicate their gradients to their parents, and the parent takes the gradients of the leaf nodes from the previous round (received at iteration $t-1$) and averages them with its own gradient, passing this averaged gradient back up the tree. Again simultaneously, each node takes the averaged gradient vectors of its children from the previous rounds, averages them with its current gradient vector, and passes the result up the spanning tree. See Fig. 3. The master node receives an average of delayed gradients from the entire tree, giving rise to updates of the form (6). We note that this is similar to the MPI all-reduce operation, except our implementation is non-blocking since we average delayed gradients with different delays at different nodes.

### 4.1 Convergence rates for delayed distributed minimization

We turn now to corollaries of the results from the previous sections that show even asynchronous distributed procedures achieve asymptotically faster rates (over centralized procedures). The key is that workers can pipeline updates by computing asynchronously and in parallel, so each worker can compute a low variance estimate of the gradient $\nabla f(x)$. We ignore the constants $L$, $G$, $R$, and $\sigma$, which are not dependent on the characteristics of the network. We also assume that each worker $i$ uses $m$ independent samples $\xi_i(j) \sim P$, $j = 1, \ldots, m$, to compute the stochastic gradient as $g_i(t) = \frac{1}{m}\sum_{j=1}^{m} \nabla F(x(t); \xi_i(j))$. Using the cyclic protocol as in Fig. 1, Theorem 1 gives the following result.

**Corollary 1.** *Let $\psi(x) = \frac{1}{2}\|x\|_2^2$, assume the conditions in Theorem 1, and assume that each worker uses $m$ samples $\xi \sim P$ to compute the gradient it communicates to the master. Then with the choice $\eta(t) = \max\{\tau^{2/3}T^{-1/3}, \sqrt{T/m}\}$ the update (5) satisfies*

$$\mathbb{E}[f(\widehat{x}(T))] - f(x^*) = \mathcal{O}\left(\min\left\{\frac{\tau^{2/3}}{T^{2/3}}, \frac{\tau^2 m}{T}\right\} + \frac{1}{\sqrt{Tm}}\right).$$

**Proof** Noting that $\sigma^2 = \mathbb{E}[\|\nabla f(x) - g_i(t)\|_2^2] = \mathbb{E}[\|\nabla f(x) - \nabla F(x;\xi)\|_2^2]/m = \mathcal{O}(1/m)$ when workers use $m$ independent stochastic gradient samples, the corollary is immediate. □

As in Theorem 1, the corollary generalizes to random delays as long as $\mathbb{E}\tau^2(t) \le B^2 < \infty$, with $\tau$ replaced by $B$ in the result. So long as $B = o(T^{1/4})$, the first term in the bound is asymptotically negligible, and we achieve a convergence rate of $\mathcal{O}(1/\sqrt{Tn})$ when $m = \mathcal{O}(n)$.

The cyclic delayed architecture has the drawback that information from a worker can take $\tau = \mathcal{O}(n)$ time to reach the master. While the algorithm is robust to delay, the downside of the architecture is that the essentially $\tau^2 m$ or $\tau^{2/3}$ term in the bounds above can be quite large. To address the large $n$ drawback, we turn our attention to the locally averaged architecture described by Figs. 2 and 3, where delays can be smaller since they depend only on the height of a spanning tree in the network. As a result of the communication procedure, the master receives a convex combination of the stochastic gradients evaluated at each worker $i$. Specifically, the master receives gradients of the form $g_\lambda(t) = \sum_{i=1}^{n} \lambda_i g_i(t - \tau(i))$ for some $\lambda$ in the simplex, where $\tau(i)$ is the delay of worker $i$, which puts us in the setting of Theorem 2. We now make the reasonable assumption that the gradient errors $\nabla f(x(t)) - g_i(t)$ are uncorrelated across the nodes in the network.[2] In statistical applications, for example, each worker may own independent data or receive streaming data from independent sources. We also set $\psi(x) = \frac{1}{2}\|x\|_2^2$, and observe

$$\mathbb{E}\left\|\sum_{i=1}^{n} \lambda_i \nabla f(x(t-\tau(i))) - g_i(t-\tau(i))\right\|_2^2 = \sum_{i=1}^{n} \lambda_i^2 \mathbb{E}\left\|\nabla f(x(t-\tau(i))) - g_i(t-\tau(i))\right\|_2^2.$$

This gives the following corollary to Theorem 2.

**Corollary 2.** *Set $\lambda_i = \frac{1}{n}$ for all $i$, $\psi(x) = \frac{1}{2}\|x\|_2^2$, and $\eta(t) = \sigma\sqrt{T}/R\sqrt{n}$. Let $\bar{\tau}$ and $\overline{\tau^2}$ denote the average of the delays $\tau(i)$ and $\tau(i)^2$. Under the conditions of Theorem 2,*

$$\mathbb{E}\left[f(\widehat{x}(T)) - f(x^*)\right] = \mathcal{O}\left(\frac{LR^2}{T} + \frac{\bar{\tau}GR}{T} + \frac{LG^2R^2n\overline{\tau^2}}{\sigma^2 T} + \frac{R\sigma}{\sqrt{Tn}}\right).$$

Asymptotically, $\mathbb{E}[f(\widehat{x}(T))] - f(x^*) = \mathcal{O}(1/\sqrt{Tn})$. In this architecture, the delay $\tau$ is bounded by the graph diameter $D$. Furthermore, we can use a slightly different stepsize setting as in Corollary 1 to get an improved rate of $\mathcal{O}(\min\{(D/T)^{2/3}, nD^2/T\} + 1/\sqrt{Tn})$. It is also possible—but outside the scope of this extended abstract—to give fast(er) convergence rates dependent on communication costs (details can be found in the long version [2]).

### 4.2 Running-time comparisons

We now explicitly study the running times of the centralized stochastic gradient algorithm (3), the cyclic delayed protocol with the update (5), and the locally averaged architecture with the update (6). To make comparisons more cleanly, we avoid constants,

[2] Similar results continue to hold under weak correlation.

| | |
|---|---|
| Centralized (3) | $\mathbb{E}f(\widehat{x}) - f(x^*) = \mathcal{O}\left(\sqrt{\frac{1}{T}}\right)$ |
| Cyclic (5) | $\mathbb{E}f(\widehat{x}) - f(x^*) = \mathcal{O}\left(\min\left(\frac{n^{2/3}}{T^{2/3}}, \frac{n^3}{T}\right) + \frac{1}{\sqrt{Tn}}\right)$ |
| Local (6) | $\mathbb{E}f(\widehat{x}) - f(x^*) = \mathcal{O}\left(\min\left(\frac{D^{2/3}}{T^{2/3}}, \frac{nD^2}{T}\right) + \frac{1}{\sqrt{nT}}\right)$ |

Table 1: Upper bounds on optimization error after $T$ units of time. See text for details.

assuming that the bound $\sigma^2$ on $\mathbb{E}\left\|\nabla f(x) - \nabla F(x;\xi)\right\|^2$ is 1, and that sampling $\xi \sim P$ and evaluating $\nabla F(x;\xi)$ requires unit time. It is also clear that if we receive $m$ uncorrelated samples of $\xi$, the variance $\mathbb{E}\|\nabla f(x) - \frac{1}{m}\sum_{j=1}^m \nabla F(x;\xi_j)\|_2^2 \le \frac{1}{m}$.

Now we state our assumptions on the relative times used by each algorithm. Let $T$ be the number of units of time allocated to each algorithm, and let the centralized, cyclic delayed and locally averaged delayed algorithms complete $T_{\text{cent}}$, $T_{\text{cycle}}$ and $T_{\text{dist}}$ iterations, respectively, in time $T$. It is clear that $T_{\text{cent}} = T$. We assume that the distributed methods use $m_{\text{cycle}}$ and $m_{\text{dist}}$ samples of $\xi \sim P$ to compute stochastic gradients. For concreteness, we assume that communication is of the same order as computing the gradient of one sample $\nabla F(x;\xi)$. In the cyclic setup of Sec. 3.1, it is reasonable to assume that $m_{\text{cycle}} = \Omega(n)$ to avoid idling of workers. For $m_{\text{cycle}} = \Omega(n)$, the master requires $\frac{m_{\text{cycle}}}{n}$ units of time to receive one gradient update, so $\frac{m_{\text{cycle}}}{n} T_{\text{cycle}} = T$. In the local communication framework, if each node uses $m_{\text{dist}}$ samples to compute a gradient, the master receives a gradient every $m_{\text{dist}}$ units of time, and hence $m_{\text{dist}} T_{\text{dist}} = T$. We summarize our assumptions by saying that in $T$ units of time, each algorithm performs the following number of iterations:

$$T_{\text{cent}} = T, \quad T_{\text{cycle}} = \frac{Tn}{m_{\text{cycle}}}, \quad \text{and} \quad T_{\text{dist}} = \frac{T}{m_{\text{dist}}}. \tag{7}$$

Combining with the bound (4) and Corollaries 1 and 2, we get the results in Table 1.

Asymptotically in the number of units of time $T$, both the cyclic and locally communicating stochastic optimization schemes have the same convergence rate. Comparing the lower order terms, since $D \le n$ for any network, the locally averaged algorithm always guarantees better performance than the cyclic algorithm. For specific graph topologies, however, we can quantify the time improvements (assuming we are in the $n^{2/3}/T^{2/3}$ regime):

- $n$-node cycle or path: $D = n$ so that both methods have the same convergence rate.
- $\sqrt{n}$-by-$\sqrt{n}$ grid: $D = \sqrt{n}$, so the distributed method has a factor of $n^{2/3}/n^{1/3} = n^{1/3}$ improvement over the cyclic architecture.
- Balanced trees and expander graphs: $D = \mathcal{O}(\log n)$, so the distributed method has a factor—ignoring logarithmic terms—of $n^{2/3}$ improvement over cyclic.

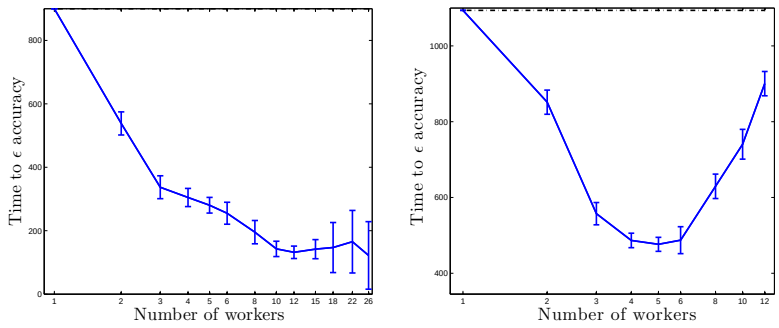
## 5 Numerical Results

Though this paper focuses mostly on the theoretical analysis of delayed stochastic methods, it is important to understand their practical aspects. To that end, we use the cyclic delayed method (6) to solve a somewhat large logistic regression problem:

$$\underset{x}{\text{minimize}} \;\; f(x) = \frac{1}{N}\sum_{i=1}^N \log(1 + \exp(-b_i \langle a_i, x\rangle)) \quad \text{subject to} \;\; \|x\|_2 \le R. \tag{8}$$

We use the Reuters RCV1 dataset [11], which consists of $N \approx 800000$ news articles, each labeled with a combination of the four labels economics, government, commerce, and medicine. In the above example, the vectors $a_i \in \{0,1\}^d$, $d \approx 10^5$, are feature vectors representing the words in each article, and the labels $b_i$ are 1 if the article is about government, $-1$ otherwise.

We simulate the cyclic delayed optimization algorithm (5) for the problem (8) for several choices of the number of workers $n$ and the number of samples $m$ computed at each worker. We summarize the results in Figure 4. We fix an $\epsilon$ (in this case, $\epsilon = .05$), then measure the




(a) (b)

Figure 4: Estimated time to compute $\epsilon$-accurate solution to the objective (8) as a function of the number of workers $n$. See text for details. Plot (a): convergence time assuming the cost of communication to the master and gradient computation are same. Plot (b): convergence time assuming the cost of communication to the master is 16 times that of gradient computation.

time it takes the stochastic algorithm (5) to output an $\widehat{x}$ such that $f(\widehat{x}) \leq \inf_{x \in \mathcal{X}} f(x) + \epsilon$. We perform each experiment ten times. The two plots differ in the amount of time $C$ required to communicate the parameters $x$ between the master and the workers (relative to the amount of time to compute the gradient on one sample in the objective (8)). For the left plot in Fig. 4(a), we assume that $C = 1$, while in Fig. 4(b), we assume that $C = 16$.

For Fig. 4(a), each worker uses $m = n$ samples to compute a stochastic gradient for the objective (8). The plotted results show the delayed update (5) enjoys speedup (the ratio of time to $\epsilon$-accuracy for an $n$-node system versus the centralized procedure) nearly linear in the number $n$ of worker machines until $n \geq 15$ or so. Since we use the stepsize choice $\eta(t) \propto \sqrt{t/n}$, which yields the predicted convergence rate given by Corollary 1, the $n^2 m/T \approx n^3/T$ term in the convergence rate presumably becomes non-negligible for larger $n$. This expands on earlier experimental work with a similar method [10], which experimentally demonstrated linear speedup for small values of $n$, but did not investigate larger network sizes.

In Fig. 4(b), we study the effects of more costly communication by assuming that communication is $C = 16$ times more expensive than gradient computation. As argued in the long version [2], we set the number of samples each worker computes to $m = Cn = 16n$ and correspondingly reduce the damping stepsize $\eta(t) \propto \sqrt{t/(Cn)}$. In the regime of more expensive communication—as our theoretical results predict—small numbers of workers still enjoy significant speedups over a centralized method, but eventually the cost of communication and delays mitigate some of the benefits of parallelization. The alternate choice of stepsize $\eta(t) = n^{2/3} T^{-1/3}$ gives qualitatively similar performance.

## 6 Conclusion and Discussion

In this paper, we have studied delayed dual averaging algorithms for stochastic optimization, showing applications of our results to distributed optimization. We showed that for smooth problems, we can preserve the performance benefits of parallelization over centralized stochastic optimization even when we relax synchronization requirements. Specifically, we presented methods that take advantage of distributed computational resources and are robust to node failures, communication latency, and node slowdowns. In addition, though we omit these results for brevity, it is possible to extend all of our expected convergence results to guarantees with high-probability.

## Acknowledgments

AA was supported by a Microsoft Research Fellowship and NSF grant CCF-1115788, and JCD was supported by the NDSEG Program and Google. We are very grateful to Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao for communicating of their proof of the bound (4). We would also like to thank Yoram Singer and Dimitri Bertsekas for reading a draft of this manuscript and giving useful feedback and references.

## References


[1] A. Agarwal, P. Bartlett, P. Ravikumar, and M. Wainwright. Information-theoretic lower bounds on the oracle complexity of convex optimization. In *Advances in Neural Information Processing Systems 23*, 2009.

[2] A. Agarwal and J. C. Duchi. Distributed delayed stochastic optimization. URL `http://arxiv.org/abs/1104.5525`, 2011.

[3] D. P. Bertsekas. Distributed asynchronous computation of fixed points. *Mathematical Programming*, 27:107–120, 1983.

[4] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Inc., 1989.

[5] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. URL `http://arxiv.org/abs/1012.1367`, 2010.

[6] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Robust distributed online prediction. URL `http://arxiv.org/abs/1012.1370`, 2010.

[7] J. Duchi, A. Agarwal, and M. Wainwright. Dual averaging for distributed optimization: convergence analysis and network scaling. *IEEE Transactions on Automatic Control*, to appear, 2011.

[8] A. Juditsky, A. Nemirovski, and C. Tauvel. Solving variational inequalities with the stochastic mirror-prox algorithm. URL `http://arxiv.org/abs/0809.0815`, 2008.

[9] G. Lan. An optimal method for stochastic composite optimization. *Mathematical Programming Series A*, 2010. Online first, to appear. URL http://www.ise.ufl.edu/glan/papers/OPT_SA4.pdf.

[10] J. Langford, A. Smola, and M. Zinkevich. Slow learners are fast. In *Advances in Neural Information Processing Systems 22*, pages 2331–2339, 2009.

[11] D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[12] A. Nedić, D.P. Bertsekas, and V.S. Borkar. Distributed asynchronous incremental subgradient methods. In D. Butnariu, Y. Censor, and S. Reich, editors, *Inherently Parallel Algorithms in Feasibility and Optimization and their Applications*, volume 8 of *Studies in Computational Mathematics*, pages 381–407. Elsevier, 2001.

[13] A. Nedić and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54:48–61, 2009.

[14] A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley, New York, 1983.

[15] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming A*, 120(1):261–283, 2009.

[16] B. T. Polyak. *Introduction to optimization*. Optimization Software, Inc., 1987.

[17] S. S. Ram, A. Nedić, and V. V. Veeravalli. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of Optimization Theory and Applications*, 147(3):516–545, 2010.

[18] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[19] J. Tsitsiklis. *Problems in decentralized decision making and computation*. PhD thesis, Massachusetts Institute of Technology, 1984.

[20] L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, 2010.