AdaDelay: Delay Adaptive Distributed Stochastic Optimization

Suvrit Sra
MIT

Adams Wei Yu
CMU

Mu Li
CMU

Alexander J. Smola
CMU

Abstract

We develop distributed stochastic convex optimization algorithms under a delayed gradient model in which server nodes update parameters and worker nodes compute stochastic (sub)gradients. Our setup is motivated by the behavior of real-world distributed computation systems; in particular, we analyze a setting wherein worker nodes can be differently slow at different times. In contrast to existing approaches, we do not impose a worst-case bound on the delays experienced but rather allow the updates to be sensitive to the actual delays experienced. This sensitivity allows use of larger stepsizes, which can help speed up initial convergence without having to wait too long for slower machines; the global convergence rate is still preserved. We experiment with different delay patterns, and obtain noticeable improvements for large-scale real datasets with billions of examples and features.

1 Introduction

We study the stochastic convex optimization problem

$$\min_{x\in\mathcal{X}} \quad f(x) := \mathbb{E}[F(x;\xi)], \tag{1.1}$$

where $\mathcal{X} \subset \mathbb{R}^d$ is a compact convex set, $F(\cdot, \xi)$ is a convex loss for each $\xi \sim \mathbb{P}$, and $\mathbb{P}$ is a (possibly unknown) probability distribution from which we can draw i.i.d. samples. Problem (1.1) is important throughout optimization and machine learning [7, 14, 19–21]. It should be distinguished from (the easier) finite-sum optimization problems [3, 18, 23, 24], for which sharper results on the empirical loss are possible but not on the generalization error.

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51.

A classic approach to solve (1.1) is stochastic gradient descent (SGD) [17] (or stochastic approximation [14]). SGD iteratively computes $x_{t+1} \leftarrow \Pi_{\mathcal{X}}(x_t - \alpha_t g_t)$, where $\Pi_{\mathcal{X}}$ denotes orthogonal projection onto $\mathcal{X}$, while $\alpha_t \geq 0$ is a suitable stepsize and g_t is an unbiased stochastic gradient, i.e., $\mathbb{E}[g_t] \in \partial f(x_t)$.

Although much more scalable than gradient descent, SGD is a sequential method that does not immediately apply to huge-scale problems which need distributed optimization [2]. This setting is central to real-world machine learning and has attracted great research interest, a large part of which is dedicated to scaling up SGD [1, 6, 9, 11, 15, 16].

Motivation. We also focus on huge-scale problems, and develop a new distributed SGD method that incorporates (and benefits from) more precise models of real-world cloud computing networks. Indeed, the delay properties exhibited by machines in cloud computing settings are often quite different from what one may observe on small clusters owned by individuals or small labs. Cloud resources are shared across users who run variegated tasks. Consequently, the cloud environment is invariably more diverse in its availability of resources such as CPU, disk, or network bandwidth, in contrast to environments where resources are shared by a small number of individuals. Thus, being able to model network delays in a more fine-grained way, and to use them to guide the optimization procedure can be of great value to both providers and users of large-scale distributed computing.

We investigate a new delay sensitive asynchronous SGD algorithm that adapts to the actual delays experienced, rather than relying on pessimistic global worst-case "bounded delays". One may envision the following practical scheme: In the beginning, the server updates parameters as soon as it receives a gradient from any machine, weighting it inversely proportional to the actual delay. Towards the end, the server takes larger update steps when it obtains gradients from infrequent contributors, and smaller ones with gradients from frequent contributors, to reduce the bias caused by the initial aggressive steps. This broad scheme partially motivates our approach.

Contributions. We introduce and analyze *AdaDelay* (**Ada**ptive **Delay**), an asynchronous SGD algorithm that more closely follows the actual delays experienced during computation. Specifically, AdaDelay uses step sizes sensitive to the actual delays observed. While this allows us to use larger stepsizes, it requires somewhat more intricate analysis because: (i) step sizes are no longer guaranteed to be monotonically decreasing; and (ii) residuals that measure progress are not independent across time as they are coupled by the delay random variable.

We validate AdaDelay by experimenting with real-world large-scale datasets containing over a billion samples and features. The experiments reveal that the models that we introduce on network delays are a reasonable approximation to the actual observed delays, and that in the regime of large delays (e.g., when there are stragglers) using delay sensitive steps is very helpful for faster generalization, that is, for models that converge more quickly on *test accuracy*; this is revealed by experiments where using AdaDelay leads to significant improvements on the test error (AUC).

Related Work. Our work is built on [1]. However, the most important difference is that [1] uses worst case delays that can be overly pessimistic, while AdaDelay uses the actual delays experienced, though at the cost of more involved theoretical analysis.

The body of related work on distributed optimization is quite large, so we can hardly be comprehensive. We summarize below a few closely related works, breaking up our summary into two typical settings: synchronous and asynchronous approaches.

Synchronous methods usually proceed in epochs, where a central node (parameter server) updates the global parameters, and waits until all the workers have finished their updates. For example, [22] and [8] leverage the finite sum structure of empirical risk minimization problems to derive a separable dual formulation for which a synchronous distributed coordinate ascent algorithm is adopted. Both [25] and [26] use an extreme strategy that only carries out a single round of communication at the end of the algorithm and then simply averages the results trained on the local workers. Performance of these algorithms, however, depends heavily on the slowest machine; commonly seen delay phenomena in commercial cloud computing systems involve multiple uncontrollable factors, and can contribute to a huge waste of resources due to waiting.

Asynchronous algorithms operate by letting each worker node run its local update without waiting for the others. Since network delays are inevitable, asynchronous methods ameliorate slow downs by avoiding waiting; thus, any updates computed by a local worker can be immediately used to update the global parameter. The work [2] is a classic reference that introduces important asynchronous strategies; several more recent key works are [1, 9, 13, 19]. In [4, 15] the authors base their convergence on sparse data or variable settings; in comparison, our framework is more general, and thus capable of covering more applications. Of particular relevance to our paper is the recent work on delay adaptive gradient scaling in an AdaGrad [5] like framework [12]. The work [12] claims substantial improvements under specialized settings over [4]. Our experiments also confirm [12]'s claims that their best learning rate is insensitive to maximum delays. However, in our experience the method of [12] overly smooths the optimization path, which can have adverse effects on real-world data (see Section 4).

Finally, to our knowledge, previous works on asynchronous SGD (and its AdaGrad variants) assume monotonically decreasing step sizes. Our analysis involves non-monotonic steps to allow using actual delays instead of worst-case bounds; this proves to be quite beneficial in realistic settings. For instance, when there are stragglers that can slow down progress for all the machines in a worst-case delay model.

2 Problem Setup and Algorithm

We build on groundwork laid by [1, 13]; like them, we also optimize (1.1) under a delayed gradient model. We use the parameter-server computational framework [11], so that a central server[1] maintains the global parameters, while worker nodes compute stochastic gradients using their share of the data. The workers communicate their gradients back to the central server (using powerful communication saving techniques implemented in the work of [10]), which then updates the shared parameters and communicates them back.

To highlight our key ideas and avoid getting bogged down in unilluminating details, we consider only smooth stochastic optimization, i.e., $f \in C_L^1$, in this paper. Straightforward, (though tedious) extensions are possible to non-smooth problems, strongly convex costs, mirror descent versions, proximal splitting settings. We leave these details to the future.

As in [1, 7, 14], we make some standard assumptions:[2]

Assumption 2.1 (Lipschitz gradients)**.** The function f has a locally L-Lipschitz gradients. That is,

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \qquad \forall \;\; x, y \in \mathcal{X}.$$

[1]This server is virtual; its physical realization may involve several machines, e.g., [10].

[2]These are easily satisfied for logistic-regression, least-squares, if the training data are bounded.

Assumption 2.2 (Bounded variance)**.** There exists a constant $\sigma < \infty$ such that

$$\mathbb{E}_\xi[\|\nabla f(x) - \nabla F(x;\xi)\|^2] \le \sigma^2, \qquad \forall\, x \in \mathcal{X}.$$

Assumption 2.3 (Compact domain)**.** Let $x^* \in \operatorname{argmin}_{x\in\mathcal{X}} f(x)$. Then,

$$\max_{x\in\mathcal{X}} \|x - x^*\| \le R.$$

Finally, an additional assumption, also made in [1] is that of bounded gradients.

Assumption 2.4 (Bounded gradient)**.** There exists a constant $G > 0$ so that

$$\|\nabla f(x)\| \le G \qquad \forall x \in \mathcal{X}.$$

These assumptions are typically satisfied in many machine learning problems, for instance, with logistic and least-squares losses, as long as the data samples ξ remain bounded, which is easy to satisfy.

Notation: Whenever a worker node returns an updated gradient at time t, we denote its associated random delay as τ_t , the delayed gradient as $g(t-\tau_t)$, and the step size as $\alpha(t,\tau_t)$. For a differentiable convex function h, the corresponding *Bregman divergence* is $D_h(x,y) := h(x) - h(y) - \langle\nabla h(y),\, x-y\rangle$. For simplicity, all norms are assumed to be Euclidean.

2.1 Delay model

Assumption 2.5 (Delay)**.** We consider the following two practical delay models:

(A) **Uniform:** Here $\tau_t \sim U(\{0, 2\bar\tau\})$. This model is a reasonable approximation to observed delays after an initial startup time of the network. We could make a more refined assumption that for $1 \le t \le T_1$ the delays are uniform on $\{0,\dots,T_1-1\}$. The analysis can be easily modified to handle this case; we omit it for brevity. Furthermore, the analysis also extends to delays having distributions with bounded support. Therefore, it indeed captures a wide spectrum of practical models.

(B) **Scaled:** For each t, there is a $\theta_t \in (0,1)$ such that $\tau_t < \theta_t t$. Moreover, assume that

$$\mathbb{E}[\tau_t] = \bar\tau_t, \qquad \mathbb{E}[\tau_t^2] = B_t^2,$$

where $\bar\tau_t$ and B_t are constants that do not grow with t (the subscript only indicates that for each t, the random variable τ_t may have a different distribution). This model allows delay processes that are richer than uniform, as it no longer requires the support to be bounded. What it needs instead are bounded first and second moments.

Note. Our analysis is flexible, and can actually cover many other delay distributions by combining the above two delay models. For example, with Gaussian delays (where τ_t obeys a Gaussian distribution but its support is truncated, since $t > 0$) may be seen as a combination of the following: 1) When $t \ge C$ (a suitable constant), the Gaussian assumption indicates $\tau_t < \theta t$, which falls under our second delay model; 2) When $0 \le t \le C$, our proof technique with bounded support (same as uniform model) applies. Of course, more refined analysis for specific delay models may help tighten constants.

2.2 Algorithm

Under the above delay models, we consider the following projected stochastic gradient iteration:

$$x_{t+1} \leftarrow \operatorname*{argmin}_{x\in\mathcal{X}} \Big[\langle g(t-\tau_t),\, x\rangle + \frac{1}{2\alpha(t,\tau_t)}\|x - x_t\|^2\Big], \tag{2.1}$$

where the stepsize $\alpha(t,\tau_t)$ is sensitive to the actual delay observed. Whenever a worker transmits a delayed gradient $g(t-\tau_t)$ at time t, the parameter server conducts an update of (2.1). Thus, (2.1) generates a sequence of iterates $\{x_t\}_{t\ge1}$; the server also maintains the averaged iterate

$$\bar{x}_T := \frac{1}{T}\sum_{t=1}^{T} x_{t+1}; \tag{2.2}$$

our convergence analysis is stated for this iterate.

3 Convergence analysis

We use stepsizes of the form $\alpha(t,\tau_t) = (L + \eta(t,\tau_t))^{-1}$, where the step offsets $\eta(t,\tau_t)$ are chosen to be sensitive to the actual delay of the incoming gradients. We typically use

$$\eta(t,\tau_t) = c\sqrt{t+\tau_t}, \tag{3.1}$$

for some constant c (to be chosen later). We can also consider time-varying c_t multipliers in (3.1) (see Corollary 3.4), but initially for clarity of presentation we let c be independent of t. If there are no delays, then $\tau_t = 0$ and iteration (2.1) reduces to the usual synchronous SGD. The constant c is used to trade off contributions to the error bound from the noise variance σ, the feasible set radius R, and the bounds on gradient norms.

Our convergence analysis builds on the groundwork of [1]. But the key difference is that our step sizes $\alpha(t,\tau_t)$ depend on the actual delay τ_t experienced, rather than on a fixed worst-case bounds on the maximum possible delay. These delay sensitive step sizes necessitate a more intricate analysis. There are two main reasons for this: (i) the stepsize $\alpha(t,\tau_t)$ is no longer independent of the actual delay τ_t; whereby (ii) the $\alpha(t,\tau_t)$ values are *no longer* monotonically decreasing, a property that the analysis of [1] relies on (and usual SGD convergence analysis also uses). We highlight our main theoretical results below; to streamline presentation, auxiliary technical lemmas are available in the supplement.

Theorem 3.1. *Let x_t be generated according to (2.1). Under Assumption 2.5 (A) (uniform delay) we have*

$$\mathbb{E}\left[\sum_{t=1}^{T}\big(f(x_{t+1})-f(x^*)\big)\right] \leq \left(\sqrt{2}cR^2\bar\tau+\frac{\sigma^2}{c}\right)\sqrt{T}+\frac{LG^2(4\bar\tau+3)(\bar\tau+1)}{6c^2}\log T + \tfrac{1}{2}(L+c)R^2+\bar\tau GR+\frac{LG^2\bar\tau(\bar\tau+1)(2\bar\tau+1)^2}{6(L^2+c^2)},$$

while under Assumption 2.5 (B) (scaled delay) we have

$$\mathbb{E}\left[\sum_{t=1}^{T}\big(f(x_{t+1})-f(x^*)\big)\right] \leq \frac{\sigma^2}{c}\sqrt{T}+\frac{1}{2}cR^2\sum_{t=2}^{T}\frac{\bar\tau_t+1}{\sqrt{2t-1}}+GR\left[1+\sum_{t=1}^{T-1}\frac{B_t^2}{(T-t)^2}\right] + G^2\sum_{t=1}^{T}\frac{B_t^2+1+\bar\tau_t}{L^2+c^2(1-\theta_t)t}+\frac{1}{2}R^2(L+c).$$

Proof Sketch. The proof begins by analyzing the difference $f(x_{t+1})-f(x^*)$; Lemma A.2 (provided in the supplement) bounds this difference, ultimately leading to an inequality of the form:

$$\mathbb{E}\big[\sum_{t=1}^{T}\big(f(x_{t+1})-f(x^*)\big)\big] \leq \mathbb{E}\big[\sum_{t=1}^{T}\Delta(t)+\Gamma(t)+\Sigma(t)\big].$$

The random variables $\Delta(t)$, $\Gamma(t)$, $\Sigma(t)$ are defined as

$$\Delta(t) := \frac{1}{2\alpha(t,\tau_t)}\left[\|x^*-x_t\|^2-\|x^*-x_{t+1}\|^2\right]; \quad (3.2)$$

$$\Gamma(t) := \langle \nabla f(x_t)-\nabla f(x_{t-\tau_t}),\, x_{t+1}-x^*\rangle; \quad (3.3)$$

$$\Sigma(t) := \tfrac{1}{2\eta(t,\tau_t)}\|\nabla f(x_{t-\tau_t})-g(t-\tau_t)\|^2. \quad (3.4)$$

Thus, all that remains to do is bound each of these random variables and combine the bounds to obtain the claim of the theorem. Lemma A.3 bounds $\Delta(t)$ under Assumption 2.5(A), while Lemma A.4 bounds it under Assumption 2.5(B). Similarly, Lemmas A.5 and Lemma A.6 bound (3.3), while Lemma A.7 bounds (3.4). □

Theorem 3.1 has several implications, which we now present as corollaries. Corollaries 3.2 and 3.3 show that both our delay models share a similar convergence rate of $\mathcal{O}(\frac{1}{\sqrt{T}})$. Corollary 3.4 shows that such results continue to hold even if we replace the constant c with a bounded (away from zero, and from above) sequence $\{c_t\}$, a setting of great practical value. Finally, Corollary 3.5 gives the convergence of a more general choice of step sizes by considering $\eta_t = c_t(t+\tau_t)^{\beta}$ for $0<\beta<1$. It also highlights the known fact that for $\beta=0.5$, the algorithm achieves the best theoretical convergence.

Corollary 3.2. *Let τ_t satisfy Assumption 2.5 (A). Then we have the following bound on $\bar x_T$:*

$$\mathbb{E}[f(\bar x_T)-f^*] = \mathcal{O}\left(D_1\frac{\sqrt{T}}{T}+D_2\frac{\log T}{T}+D_3\frac{1}{T}\right).$$

where

$$D_1=\sqrt{2}cR^2\bar\tau+\frac{\sigma^2}{c},\ D_2=\frac{LG^2(4\bar\tau+3)(\bar\tau+1)}{6c^2},$$

$$D_3=\tfrac{1}{2}(L+c)R^2+\bar\tau GR+\frac{LG^2\bar\tau(\bar\tau+1)(2\bar\tau+1)^2}{6(L^2+c^2)}.$$

The constant D_1 captures the variance due to stochastic gradients as well as the added variance due to the nonmonotone step sizes. The terms D_2 and D_3 capture the contribution to convergence based on delays and the Lipschitz smoothness of the gradients. We believe that it may be possible to get rid of the extra $\log T$ factor in the bound by a more refined analysis.

The following corollary follows easily from adapting the proof of Theorem 3.1, and combining it with Lemma A.6 which bounds $\Gamma(t)$ under the stated assumptions on the first two moments of the delay random variables.

Corollary 3.3. *Let τ_t satisfy Assumption 2.5 (B); let $\bar\tau_t=\tau$, $\theta_t=\theta$, and $B_t=B$ for all t. Then,*

$$\mathbb{E}[f(\bar x_T)-f^*] = \mathcal{O}\left(\frac{D_4}{\sqrt{T}}+\frac{D_5\log\frac{L^2+c^2(1-\theta)T}{L^2}}{T}+\frac{D_6}{T}\right)$$

where

$$D_4=\left[\frac{1}{\sqrt{2}}cR^2(\bar\tau+1)+\frac{\sigma^2}{c}\right],\ D_5=\frac{G^2(B^2+\tau+1)}{c^2(1-\theta)},$$

$$D_6=\tfrac{1}{2}(L+c)R^2+GR\left(1+\frac{\pi^2B^2}{6}\right).$$

Here, the constant D_4 describes the contribution due to the variance introduced by stochastic gradients as well as the non-monotone step sizes. The role of D_5 and D_6 is similar to D_2 and D_3 from Corollary 3.2. As the reader may notice, the impact of the noise model on delay is on its contribution to D_5 and D_6 (which shrink as $O(\log T/T)$ and $O(1/T)$ respectively), and using properties of more specialized noise models one may be able to obtained more refined estimates of these constants.

The next corollary is of great value empirically. Theoretically, it states the impact on convergence rates for time varying choice of c_t.

Corollary 3.4. *If $\eta_t = c_t\sqrt{t+\tau_t}$ with $0 < M_1 \le c_t \le M_2$, then the conclusion of Theorem 3.1, Corollary 3.2 and 3.3 still hold, except that the term c is replaced by M_2 and $\frac{1}{c}$ by $\frac{1}{M_1}$.*

Finally, if one wishes to use step size offsets $\eta_t = c_t(t+\tau_t)^\beta$ where $0 < \beta < 1$, we obtain a convergence bound of the form stated below (we report only the asymptotically worst term, as this result is of limited importance).

Corollary 3.5. *Let $\eta_t = c_t(t+\tau_t)^\beta$ with $0 < M_1 \le c_t \le M_2$ and $0 < \beta < 1$. Then, there exists a constant D_7 such that*

$$\mathbb{E}[f(\bar{x}_T) - f^*] = \mathcal{O}\left(\frac{D_7}{T^{\min(\beta, 1-\beta)}}\right).$$

4 Experiments

We now evaluate the efficiency of AdaDelay in a distributed environment using large-scale real datasets.

4.1 Datasets and setup

We collected two click-through rate datasets for evaluation, which are shown in Table 1. One is the Criteo dataset[3], where the first 8 days are used for training while the following 2 days are used for validation. We applied one-hot encoding for category and string features. The other dataset, named CTR2, is collected from a large Internet company. We sampled 100 million examples from three weeks for training, and 20 millions examples from the next week for validation. We extracted 2 billion unique features using the on-production feature extraction module. These two datasets have comparable size, but different example-feature ratios. We adopt Logistic Regression as our classification model.

	# train	# test	# features	nnz
Criteo	1.5B	400M	360M	58B
CTR2	110M	20M	1.9B	13B

Table 1: CTR datasets. M denotes millions; B billions.

All experiments were carried on a cluster with 20 machines. Most machines are equipped with dual Intel Xeon 2.40GHz CPUs, 32 GB memory and 1 Gbit/s Ethernet.

[3] http://labs.criteo.com/downloads/download-terabyte-click-logs/

4.2 Algorithms

We compare AdaDelay with two related methods AsyncAdaGrad [1] and AdaptiveRevision [12]. Their main difference lies in the choice of the learning rate at time t: $\alpha(t, \tau_t) = (L + \eta(t, \tau_t))^{-1}$. Denote by $\eta_j(t, \tau_t)$ the j-th element of $\eta(t, \tau_t)$, and similarly $g_j(t - \tau_t)$ the delayed gradient on feature j. AsyncAdaGrad adopts a scaled learning rate $\eta_j(t, \tau_t) = \sqrt{\sum_{i=1}^t g_j^2(i, \tau_i)}$. AdaptiveRevision takes into account actual delays by considering $g_j^{\text{bak}}(t, \tau_t) = \sum_{i=t-1}^{t-\tau} g_j(i, \tau_i)$. It uses a non-decreasing learning rate based on $\sqrt{\sum_{i=1}^t g_j^2(i, \tau_i) + 2g_j(t, \tau_t)g_j^{\text{bak}}(t, \tau_t)}$.

Similar to AsyncAdaGrad and AdaptiveRevision, we use a scaled learning rate in AdaDelay to better model the nonuniform sparsity of the dataset (this step size choice falls within the purview of Corollary 3.4). In other words, we set $\eta_j(t, \tau_t) = c_j\sqrt{t+\tau_t}$, where $c_j = \sqrt{\frac{1}{t}\sum_{i=1}^t \frac{i}{i+\tau_i} g_j^2(i - \tau_i)}$ averages the weighted delayed gradients on feature j. We follow the common practice of fixing L to 1 while choosing the best $\alpha(t, \tau_t) = \alpha_0(L + \eta(t, \tau_t))^{-1}$ by a grid search over α_0.

We set the minibatch size to 10^5 and 10^4 for Criteo and CTR2, respectively, to reduce the communication frequency for better system performance[4]. We search for α_0 in the range $[10^{-4}, 1]$ and report the best results for each compared algorithm.

4.3 Implementation

We implemented these three methods in the parameter server framework [11], which is a high-performance asynchronous communication library supporting various data consistency models. There are two groups of nodes in this framework: workers and servers. Worker nodes run independently from each other. At each time, a worker first reads a minibatch of data from a distributed filesystem, and then pulls the relevant recent working set of parameters, namely the weights of the features that appear in this minibatch, from the server nodes. It next computes the gradients and then pushes these gradients to the server nodes.

The server nodes maintain the weights. For each feature, both AsyncAdaGrad and AdaDelay store the weight and the accumulated gradient which is used to compute the scaled learning rate. While AdaptiveRevision needs two more entries for each feature.

To compute the actual delay τ for AdaDelay, we let the server nodes record the time $t(w, i)$ when worker

[4] Probably due to the scale and the sparsity of the datasets, we observed no significant improvement when decreasing the minibatch size.

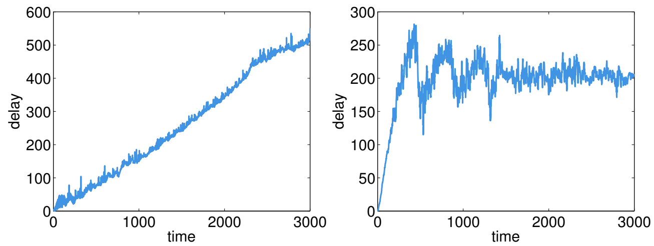


Figure 1: The first 3,000 observed delays at server nodes. Left: Criteo dataset with 1,600 workers; Right: CTR2 dataset with 400 workers.

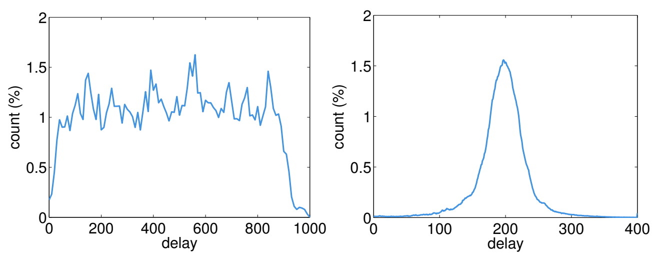


Figure 2: Histogram of all delays (left: Criteo with 1,600 workers; right: CTR2 with 400 workers).

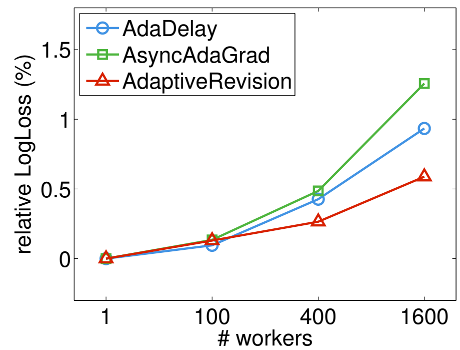
w is pulling the weight for minibatch i. Denote by $t'(w,i)$ the time when the server nodes are updating the weight by using the gradients of this minibatch. Then the actual delay of this minibatch can be obtained by $t'(w,i) - t(w,i)$.

AdaptiveRevision needs gradient components g_j^{bck} for each feature j to calculate its learning rate. If we send g_j^{bck} over the network by following [12], we increase the total network communication by 50%, which greatly harms system performance due to the limited network bandwidth. Instead, we store g_j^{bck} at the server node during while processing this minibatch. This incurs no extra network overhead, however, it increases the memory consumption of the server nodes.

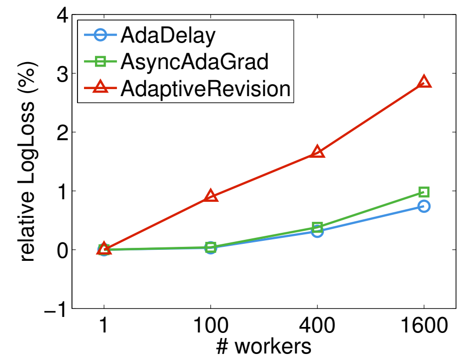
The parameter server implements a node using an operating system process, which has its own communication and computation threads. In order to run thousands of workers on our limited hardware, we may combine server workers into a single process to reduce the system overhead.

4.4 Results

Delays. We first visualize the actual delays observed at server nodes. As noted from Figure 1, delay τ_t is around θt at the early stage of the training, where the scaling constant θ varies for different tasks. For example, it is close to 0.2 when training the Criteo dataset with 1,600 workers, while it increases to 1 for the CTR2 dataset with 400 workers. After the delay



(a) Criteo



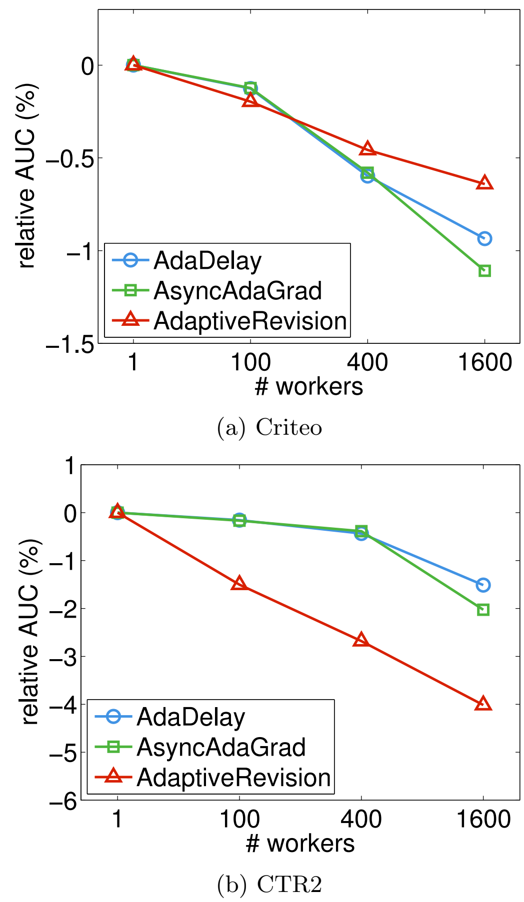
(b) CTR2

Figure 3: Relative (% worsening) of online LogLoss as function of maximal delays (lower is better).

hits the value u, which is often half of the number of worker nodes, it behaves like a Gaussian distribution with mean u, which are shown in Figure 2.

Loss and AUC. Next, we present the comparison results of these three algorithms by varying the number of workers. Following [12] we use the online LogLoss as the criterion. That is, given example with feature vector d and label $y \in \{+1, -1\}$, we calculate the loss function $f(x, (d, y)) = \log(1 + \exp(-y\langle d, x\rangle))$ before updating x using (y, d). Similar to [12], we report the average LogLoss over the second half of the training data to ignore the possible large values when starting training.

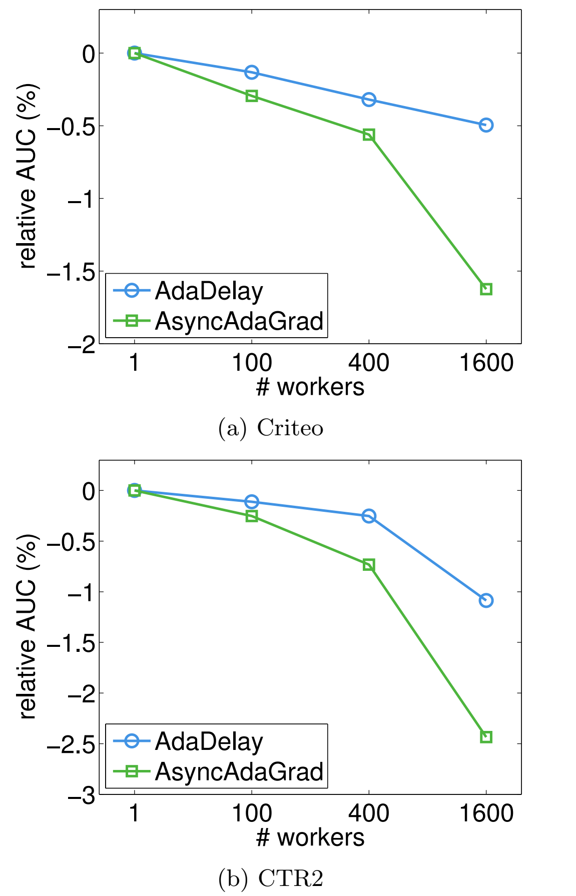
Figure 3 reports the relative change in online LogLoss for the three algorithms compared (smaller value is better). It is seen that on the Criteo dataset, AdaDelay performs better than AsyncAdaGrad, though AdaptiveRevision is slightly better than both. However, for the larger CTR2 dataset, both AdaDelay and AsyncAdaGrad are substantially better than AdaptiveRevision. The reason why it differs from [12] is probably due to the datasets we used are 1000 times larger than the ones reported by [12], and we evaluated the algorithms in a distributed environment rather than a simulated setting where a large minibatch size



(a) Criteo

(b) CTR2

Figure 4: Relative test AUC (higher is better) as function of maximal delays.



(a) Criteo

(b) CTR2

Figure 5: Relative test AUC (higher is better) as function of maximal delays with the existence of stragglers.

is necessary for the former. However, as reported [12], we also observed that AdaptiveRevision's best learning rate is insensitive to the number of workers.

AdaDelay seems to have a tiny edge over AsyncAdaGrad, and as predicted by our theory, this edge grows much bigger when there are large delays (e.g., due to stragglers)—we report on this in greater detail in Section 4.4.1.

Besides the LogLoss, AUC is another important merit for computational advertising, which measures the ranking ability of the model and often 1% difference is significant for click-through rate estimation. We made a separate validation dataset for calculating the AUC, and shown the results on Figure 4. As can be seen, the test AUC results are consistent with the online LogLoss.

4.4.1 Stragglers

Previous experiments indicate that AdaDelay improves upon AsyncAdaGrad when a large number of workers (greater than 400) is used, which means the delay adaptive learning rate takes effect when the delay can be large. To further investigate this phenomenon, we simulated an overloaded cluster where several stragglers may produce large delays; we do this by slowing down half of the workers by a random factor in $[1, 4]$ when computing gradients. The test AUC are shown in Figure 5[5]. As can be seen, AdaDelay consistently outperforms AsyncAdaGrad, which shows that adaptive modeling of the actual delay is better than using a constant worst case delay when the variance of the delays is large.

4.4.2 Scalability

Finally we report the system performance. We first present the speedup from 1 machine to 16 machines, where each machine runs 100 workers. We observed a near linear speedup of AdaDelay, which is shown in Figure 6. The main reason is due to the asynchronous updating which removes the dependencies between worker nodes. In addition, using multiple

[5]As before, the results on online LogLoss are similar to the test AUC and therefore omitted.

	AdaDelay	AsyncAdaGrad	AdaptiveRevision
Criteo	24 GB	24 GB	55 GB
CTR2	97 GB	97 GB	200 GB

Table 2: Total memory used by server nodes.

workers within a machine can fully utilize the computational resources by hiding the overhead of reading data and communicating the parameters. The results of AsyncAdaGrad and AdaptiveRevision are similar to AdaDelay because their computational workloads are identical except for parameter updating, which affects the overall system performance little.

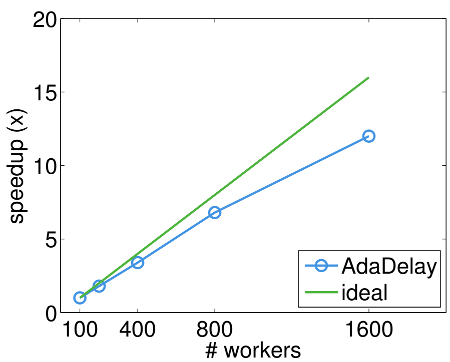


Figure 6: The speedup of AdaDelay. The results of AsyncAdaGrad and AdaptiveRevision are almost identical to AdaDelay and therefore omitted.

In the parameter server framework, worker nodes only need to cache one or a few data minibatches. Most memory is used by the server nodes to store the model. We summarize the server memory usage for the three algorithms compared in Table 2.

As expected, AdaDelay and AsyncAdaGrad have similar memory consumption because the extra storage needed by AdaDelay to track and compute the incurred delays τ_t is tiny. However AdaptiveRevision doubles memory usage, because of the extra entries that it needs for each feature, and because of the cached delayed gradient g^{bak}.

5 Conclusions

In real distributed computing environment, there are multiple factors contributing to delay, such as the CPU speed, I/O of disk, and network throughput. With the inevitable and sometimes unpredictable phenomenon of delay, we considered distributed convex optimization by developing and analyzing *AdaDelay*, an asynchronous SGD method that tolerates stale gradients.

A key component of our work that differs from existing approaches is the use of (server-side) updates sensitive to the actual delay observed in the network. This allows us to use larger stepsizes initially, which can lead to more rapid initial convergence, and stronger ability to adapt to the environment. We discussed details of two different realistic delay models: (i) uniform (more generally, bounded support) delays, and (ii) scaled delays with constant first and second moments but not-necessarily bounded support. Under both models, we obtain theoretically optimal convergence rates.

Adapting more closely to observed delays and incorporating server-side delay sensitive gradient aggregation that combines the benefits of the adaptive revision framework [12] with our delayed gradient methods is an important future direction. Extension of our analysis to handle constrained convex optimization problems without projection oracles is an important part of future work. Finally, how to apply our techniques to large-scale nonconvex problems such as matrix factorization and deep neural networks is an important direction worth studying.

Acknowledgments

SS and AS were partially supported by NSF grant IIS-1409802.

References

[1] A. Agarwal and J. C. Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 873–881, 2011.

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

[3] D. P. Bertsekas. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. *Optimization for Machine Learning*, 2010:1–38, 2011.

[4] J. Duchi, M. I. Jordan, and B. McMahan. Estimation, optimization, and parallelism when data is sparse. In *NIPS 26*, pages 2832–2840, 2013.

[5] J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[6] J. C. Duchi, A. Agarwal, and M. J. Wainwright. Dual averaging for distributed optimization: convergence analysis and network scaling. *Automatic Control, IEEE Transactions on*, 57(3):592–606, 2012.

[7] S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization i: A generic

algorithmic framework. *SIAM Journal on Optimization*, 22(4):1469–1492, 2012.

[8] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan. Communication-efficient distributed dual coordinate ascent. In *NIPS*, pages 3068–3076, 2014.

[9] J. Langford, A. J. Smola, and M. Zinkevich. Slow learners are fast. In *Neural Information Processing Systems*, 2009. URL http://arxiv.org/abs/0911.0491.

[10] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su. Scaling distributed machine learning with the parameter server. In *Operating Systems Design and Implementation (OSDI)*, 2014.

[11] M. Li, D. G. Andersen, A. J. Smola, and K. Yu. Communication efficient distributed machine learning with the parameter server. In *NIPS 27*, pages 19–27, 2014.

[12] B. McMahan and M. Streeter. Delay-tolerant algorithms for asynchronous distributed online learning. In *NIPS 27*, pages 2915–2923, 2014.

[13] A. Nedić, D. P. Bertsekas, and V. S. Borkar. Distributed asynchronous incremental subgradient methods. *Studies in Computational Mathematics*, 8:381–407, 2001.

[14] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[15] F. Niu, B. Recht, C. Re, and S. J. Wright. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pages 693–701, 2011.

[16] 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.

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

[18] M. W. Schmidt, N. L. Roux, and F. R. Bach. Minimizing finite sums with the stochastic average gradient. *CoRR*, abs/1309.2388, 2013.

[19] O. Shamir and N. Srebro. Distributed stochastic optimization and learning. In *Proceedings of the 52nd Annual Allerton Conference on Communication, Control, and Computing*, 2014.

[20] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*, volume 16. SIAM, 2014.

[21] N. Srebro and A. Tewari. Stochastic Optimization for Machine Learning. ICML 2010 Tutorial, 2010.

[22] T. Yang. Trading computation for communication: Distributed stochastic dual coordinate ascent. In *NIPS*, pages 629–637, 2013.

[23] A. W. Yu, Q. Lin, and T. Yang. Doubly stochastic primal-dual coordinate method for regularized empirical risk minimization with factorized data. *CoRR*, abs/1508.03390, 2015.

[24] Y. Zhang and L. Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *ICML*, pages 353–361, 2015.

[25] Y. Zhang, J. C. Duchi, and M. J. Wainwright. Communication-efficient algorithms for statistical optimization. *Journal of Machine Learning Research*, 14(1):3321–3363, 2013.

[26] M. Zinkevich, M. Weimer, A. J. Smola, and L. Li. Parallelized stochastic gradient descent. In *NIPS*, pages 2595–2603, 2010.

A Technical details of the convergence analysis

We collect below some basic tools and definitions from convex analysis.

Definition A.1 (Bregman divergence)**.** Let $h : \mathcal{X} \times \mathcal{X} \to [0, \infty]$ be differentiable strictly convex function. The *Bregman divergence* generated by h is

$$D_h(x, y) := h(x) - h(y) - \langle \nabla h(y),\, x - y\rangle, \qquad x, y \in \mathcal{X}. \tag{A.1}$$

– **Fenchel conjugate:**

$$f^*(y) = \sup_{x \in \mathcal{X}} \langle x, y\rangle - f(x) \tag{A.2}$$

– **Prox operator:**

$$\operatorname{prox}_f(x) = \operatorname*{argmin}_{y \in \mathcal{X}} f(y) + \frac{1}{2}\|x - y\|_2^2, \qquad \forall\, x \in \mathcal{X} \tag{A.3}$$

– **Moreau decomposition:**

$$x = \operatorname{prox}_f(x) + \operatorname{prox}_{f^*}(x), \qquad \forall\, x \in \mathcal{X} \tag{A.4}$$

– **Fenchel-Young inequality:**

$$\langle x,\, y\rangle \le f(x) + f^*(y) \tag{A.5}$$

– **Projection lemma:**

$$\langle y - \Pi_{\mathcal{X}}(y),\, x - \Pi_{\mathcal{X}}(y)\rangle \le 0, \qquad \forall\, x \in \mathcal{X}. \tag{A.6}$$

– **Descent lemma:**

$$f(y) \le f(x) + \langle \nabla f(x),\, y - x\rangle + \tfrac{L}{2}\|y - x\|^2. \tag{A.7}$$

– **Four-point identity:** Bregman divergences satisfy the following *four point identity*:

$$\langle \nabla h(a) - \nabla h(b),\, c - d\rangle = D_h(d, a) - D_h(d, b) - D_h(c, a) + D_h(c, b). \tag{A.8}$$

A special case of (A.8) is the "three-point" identity

$$\langle \nabla h(a) - \nabla h(b),\, b - c\rangle = D_h(c, a) - D_h(c, b) - D_h(b, a). \tag{A.9}$$

A.1 Bounding the change $f(x_{t+1}) - f(x^*)$

We start the analysis by bounding the gap $f(x_{t+1}) - f(x^*)$. The lemma below is just a combination of several results of [1]. We present the details below in one place for easy reference. The impact of our delay sensitive step sizes shows up in subsequent lemmas, where we bound the individual terms that arise from Lemma A.2.

Lemma A.2. *At any time-point t, let the gradient error due to delays be*

$$e_t := \nabla f(x_t) - g(t - \tau_t). \tag{A.10}$$

Then, we have the following (deterministic) bound:

$$\begin{aligned}
&f(x_{t+1}) - f(x^*) \\
&= \frac{1}{2\alpha(t, \tau_t)}\left[\|x^* - x_t\|^2 - \|x^* - x_{t+1}\|^2\right] + \langle e_t,\, x_{t+1} - x^*\rangle + \tfrac{L - 1/\alpha(t,\tau_t)}{2}\|x_t - x_{t+1}\|^2, \\
&\le \frac{1}{2\alpha(t, \tau_t)}\left[\|x^* - x_t\|^2 - \|x^* - x_{t+1}\|^2\right] + \langle \nabla f(x_t) - \nabla f(x(t - \tau_t)),\, x_{t+1} - x^*\rangle \\
&\qquad + \langle \nabla f(x(t - \tau_t)) - g(t - \tau_t),\, x_t - x^*\rangle + \tfrac{1}{2\eta(t,\tau_t)}\|\nabla f(x(t - \tau_t)) - g(t - \tau_t)\|^2.
\end{aligned} \tag{A.11}$$

Proof. Using convexity of f we have

$$f(x_t) - f(x^*) \leq \langle \nabla f(x_t),\, x_{t+1} - x^* \rangle + \langle \nabla f(x_t),\, x_t - x_{t+1} \rangle. \tag{A.12}$$

Now apply Lipschitz continuity of ∇f to the second term to obtain

$$\begin{aligned} & f(x_t) - f(x^*) \leq \langle \nabla f(x_t),\, x_{t+1} - x^* \rangle + f(x_t) - f(x_{t+1}) + \tfrac{L}{2}\|x_t - x_{t+1}\|^2, \\ \implies\ & f(x_{t+1}) - f(x^*) \leq \langle \nabla f(x_t),\, x_{t+1} - x^* \rangle + \tfrac{L}{2}\|x_t - x_{t+1}\|^2. \end{aligned} \tag{A.13}$$

Using the definition (A.10) of the gradient error e_t, we can rewrite (A.13) as

$$f(x_{t+1}) - f(x^*) \leq \underbrace{\langle g(t-\tau_t),\, x_{t+1} - x^* \rangle}_{T1} + \underbrace{\langle e_t,\, x_{t+1} - x^* \rangle}_{T2} + \tfrac{L}{2}\|x_t - x_{t+1}\|^2.$$

To complete the proof, we bound the terms $T1$ and $T2$ separately below.

Bounding T1: Since x_{t+1} is a minimizer in (2.1), from the projection inequality (A.6) we have

$$\langle x_t - \alpha(t,\tau_t) g(t-\tau_t) - x_{t+1},\, x - x_{t+1} \rangle \leq 0, \quad \forall x \in \mathcal{X}.$$

Choose $x = x^*$; then rewrite the above inequality and identity (A.9) with $h(x) = \frac{1}{2}\|x\|^2$ to get

$$\begin{aligned} \alpha(t,\tau_t)\langle g(t-\tau_t),\, x_{t+1} - x^* \rangle &\leq \langle x_t - x_{t+1},\, x_{t+1} - x^* \rangle \\ &= \tfrac{1}{2}\|x^* - x_t\|^2 - \tfrac{1}{2}\|x^* - x_{t+1}\|^2 - \tfrac{1}{2}\|x_{t+1} - x_t\|^2; \end{aligned}$$

Plugging in this bound for $T1$ and collecting the $\|x_{t+1} - x_t\|^2$ terms we obtain

$$\begin{aligned} & f(x_{t+1}) - f(x^*) \\ & \leq \tfrac{1}{2\alpha(t,\tau_t)}\left[\|x^* - x_t\|^2 - \|x^* - x_{t+1}\|^2 - \|x_{t+1} - x_t\|^2\right] + \langle e_t,\, x_{t+1} - x^* \rangle + \tfrac{L}{2}\|x_t - x_{t+1}\|^2 \\ & = \tfrac{1}{2\alpha(t,\tau_t)}\left[\|x^* - x_t\|^2 - \|x^* - x_{t+1}\|^2\right] + \langle e_t,\, x_{t+1} - x^* \rangle + \tfrac{L - 1/\alpha(t,\tau_t)}{2}\|x_t - x_{t+1}\|^2. \end{aligned} \tag{A.14}$$

Bounding T2: Adding and subtracting $\nabla f(x(t-\tau_t))$ we obtain

$$\begin{aligned} & \langle e_t,\, x_{t+1} - x^* \rangle = \langle \nabla f(x_t) - g(t-\tau_t),\, x_{t+1} - x^* \rangle \\ & = \langle \nabla f(x_t) - \nabla f(x(t-\tau_t)),\, x_{t+1} - x^* \rangle + \langle \nabla f(x(t-\tau_t)) - g(t-\tau_t),\, x_{t+1} - x^* \rangle \\ & = \langle \nabla f(x_t) - \nabla f(x(t-\tau_t)),\, x_{t+1} - x^* \rangle + \langle \nabla f(x(t-\tau_t)) - g(t-\tau_t),\, x_t - x^* \rangle \\ & \quad + \langle \nabla f(x(t-\tau_t)) - g(t-\tau_t),\, x_{t+1} - x_t \rangle \\ & \leq \langle \nabla f(x_t) - \nabla f(x(t-\tau_t)),\, x_{t+1} - x^* \rangle + \langle \nabla f(x(t-\tau_t)) - g(t-\tau_t),\, x_t - x^* \rangle \\ & \quad + \tfrac{1}{2\eta(t,\tau_t)}\|\nabla f(x(t-\tau_t)) - g(t-\tau_t)\|^2 + \tfrac{\eta(t,\tau_t)}{2}\|x_{t+1} - x_t\|^2, \end{aligned}$$

where the last inequality is an application of (A.5). Adding this inequality to (A.14) and using $1/\alpha(t,\tau_t) = L + \eta(t,\tau_t)$, we obtain (A.11). □

The next step is to take expectations over (A.11) and then further bound the resulting terms separately. Note that $\nabla f(x(t-\tau_t)) - g(t-\tau_t)$ is independent of x_t given $g(1), \ldots, g(t-\tau_t-1)$ (since x_t is a function of gradients up to time $t-\tau_t-1$). Thus, the third term in (A.11) has zero expectation. It remains to consider expectations over the following three quantities:

$$\Delta(t) := \frac{1}{2\alpha(t,\tau_t)}\left[\|x^* - x_t\|^2 - \|x^* - x_{t+1}\|^2\right]; \tag{A.15}$$

$$\Gamma(t) := \langle \nabla f(x_t) - \nabla f(x(t-\tau_t)),\, x_{t+1} - x^* \rangle; \tag{A.16}$$

$$\Sigma(t) := \tfrac{1}{2\eta(t,\tau_t)}\|\nabla f(x(t-\tau_t)) - g(t-\tau_t)\|^2. \tag{A.17}$$

Lemma A.3 bounds (A.15) under Assumption 2.5(A), while Lemma A.4 provides a bound under the Assumption 2.5(B). Similarly, Lemmas A.5 and A.6 bound (A.16), while Lemmas A.7 bounds (A.17). Combining these bounds we obtain the theorem.

A.2 Bounding Δ, Γ, and Σ

Lemma A.3. *Let $\Delta(t)$ be given by (A.15), and let Assumption 2.5 (A) hold. Then,*

$$\sum_{t=1}^{T}\mathbb{E}[\Delta(t)] = \frac{1}{2}\sum_{t=1}^{T}\mathbb{E}\Big[\frac{1}{\alpha(t,\tau_t)}\left(\|x^*-x_t\|^2-\|x^*-x_{t+1}\|^2\right)\Big] \leq \tfrac{1}{2}(L+c)R^2+\sqrt{2}cR^2\bar{\tau}\sqrt{T}.$$

Proof. Unlike the delay independent step sizes treated in [1], bounding $\Delta(t)$ requires some more work because $\alpha(t,\tau_t)$ depends on τ_t, which in turn breaks the monotonically decreasing nature of $\alpha(t,\tau_t)$ (we wish to avoid using a fixed worst case bound on the steps, to gain more precise insight into the impacts of being sensitive to delays), necessitating a more intricate analysis.

Let $r_t = \|x_t - x^*\|^2$. Observe that although $r_t \perp\!\!\!\perp \tau_t$, it is *not* independent of $\tau(t-1)$. Thus, with

$$z_t = \frac{1}{\alpha(t,\tau_t)} - \frac{1}{\alpha(t-1,\tau_{t-1})} = c(\sqrt{t+\tau_t}-\sqrt{t-1+\tau_{t-1}}),$$

we have

$$\sum_{t=1}^{T}\mathbb{E}[\Delta(t)] = \frac{1}{2}\mathbb{E}\Big[\frac{r_1}{\alpha(1,\tau(1))}+\sum_{t=2}^{T}z_tr_t\Big] \leq \frac{1}{2}(L+c)R^2+\frac{1}{2}\mathbb{E}\Big[\sum_{t=2}^{T}z_tr_t\Big]. \tag{A.18}$$

Since $\alpha(t,\tau_t)$ is *not* monotonically decreasing with t, while upper-bounding $\mathbb{E}[\Delta(t)]$ we cannot simply discard the final term in (A.18).

When $\tau(t-1)\sim U(\{0,2\bar{\tau}\})$, r_t uniformly takes on at most $2\bar{\tau}+1$ values

$$r_{t,s} := \|x_{t,s}-x^*\|^2, \qquad s\in[2\bar{\tau}],$$

where $x_{t,s}=\Pi_{\mathcal{X}}[x_{t-1}-\alpha(t-1,\tau(t-1)=s)g(t-1,\tau(t-1))]$. Given a delay $\tau(t-1)=s$, r_t is just $r_{t,s}$. Using $z_t=\alpha(t)^{-1}-\alpha(t-1)^{-1}=c\sqrt{t+\tau_t}-c\sqrt{t-1+\tau_{t-1}}$, we have

$$z_{t,s}=c\left(\sqrt{t+\tau_t}-\sqrt{t-1+s}\right), \qquad s\in[2\bar{\tau}].$$

Using nested expectations $\mathbb{E}[z_tr_t]=\mathbb{E}_{\tau_t}[\mathbb{E}[z_tr_t|\tau_t]]$ we then see that

$$\begin{aligned}\mathbb{E}[z_tr_t] &= \frac{1}{2\bar{\tau}+1}\sum_{l=0}^{2\bar{\tau}}\left(\sum_{s=0}^{2\bar{\tau}}(2\bar{\tau}+1)^{-1}r_{t,s}c\left(\sqrt{t+l}-\sqrt{t-1+s}\right)\right)\\ &\leq \frac{1}{2\bar{\tau}+1}\sum_{l=0}^{2\bar{\tau}}\left(\sum_{s=0}^{l-1}(2\bar{\tau}+1)^{-1}r_{t,s}c\left(\sqrt{t+l}-\sqrt{t-1+s}\right)\right),\end{aligned}$$

where we dropped the terms with $s\geq l$ as they are non-positive.

Consider now the inner summation above. We have

$$\begin{aligned}&\frac{c}{2\bar{\tau}+1}\sum_{s=0}^{l-1}r_{t,s}\left(\sqrt{t+l}-\sqrt{t-1+s}\right)\\ &\leq \frac{cR^2}{2\bar{\tau}+1}\sum_{s=0}^{l-1}\left(\sqrt{t+l}-\sqrt{t-1+s}\right)\\ &= \frac{cR^2}{2\bar{\tau}+1}\sum_{s=0}^{l-1}\frac{l-s+1}{\sqrt{t+l}+\sqrt{t-1+s}}\\ &\leq \frac{cR^2}{2\bar{\tau}+1}\frac{1}{\sqrt{2t-1}}\sum_{s=0}^{l-1}(l-s+1)\\ &= \frac{cR^2}{2\bar{\tau}+1}\frac{1}{\sqrt{2t-1}}\frac{3l+l^2}{2}.\end{aligned}$$

Thus, we now consider

$$\begin{aligned}\mathbb{E}[z_t r_t] &\le \frac{1}{2\bar\tau+1}\sum_{l=0}^{2\bar\tau}\frac{cR^2}{2\bar\tau+1}\frac{1}{\sqrt{2t-1}}\frac{3l+l^2}{2}\\ &= \frac{cR^2}{(2\bar\tau+1)^2\sqrt{2t-1}}(2\bar\tau+1)(4\bar\tau+2.5)\bar\tau\\ &< \frac{2cR^2\bar\tau}{\sqrt{2t-1}}.\end{aligned}$$

Summing over $t = 2$ to T, we finally obtain the upper bound

$$\sum_{t=2}^{T}\mathbb{E}[z_t r_t] \le cR^2\bar\tau\sum_{t=2}^{T}\frac{1}{\sqrt{2t-1}} \le 2cR^2\bar\tau\sqrt{2T}. \qquad \square$$

Lemma A.4. *Let Assumption (2.5) (B) hold. Then*

$$\sum_{t=1}^{T}\mathbb{E}[\Delta(t)] \le \frac{1}{2}R^2(L+c) + \frac{1}{2}cR^2\sum_{t=2}^{T}\frac{\bar\tau_t+1}{\sqrt{2t-1}}.$$

Proof. Proceeding as for Lemma A.3, according to (A.18), the task reduces to bounding $\mathbb{E}[z_t r_t]$. Consider thus,

$$\mathbb{E}[z_t r_t] \le \mathbb{E}[z_t^+ r_t] \le R^2\mathbb{E}[z_t^+],$$

where we use z_t^+ to denote $\max(z_t, 0)$. Let us now control the last expectation. Let $P_t(l) = \mathbb{P}(\tau(t) = l)$, then

$$\begin{aligned}\mathbb{E}[z_t^+] &= \sum_{\tau_t,\tau_{t-1}} P(\tau_t,\tau_{t-1})\max(0, z_t)\\ &= c\sum_{l=0}^{t-1}\sum_{s=0}^{t-2}P_t(l)P_{t-1}(s)[\sqrt{t+l}-\sqrt{t-1+s}]^+\\ &= c\sum_{l=0}^{t-1}\sum_{s=0}^{l}P_t(l)P_{t-1}(s)\frac{l+1-s}{\sqrt{t+l}+\sqrt{t-1+s}}\\ &\le c\sum_{l=0}^{t-1}\sum_{s=0}^{l}P_t(l)P_{t-1}(s)\frac{l+1}{\sqrt{2t+l-1}}\\ &\le c\sum_{l=0}^{t-1}P_t(l)\frac{l+1}{\sqrt{2t+l-1}}\\ &\le c\sum_{l=0}^{t-1}P_t(l)\frac{l+1}{\sqrt{2t-1}} = c\frac{\bar\tau_t+1}{\sqrt{2t-1}}.\end{aligned}$$

So

$$\sum_{t=2}^{T}R^2\mathbb{E}[z_t^+] \le cR^2\sum_{t=2}^{T}\frac{\bar\tau_t+1}{\sqrt{2t-1}}. \qquad \square$$

Lemma A.5.

$$\begin{aligned}\sum_{t=1}^{T}\mathbb{E}[\Gamma(t)] &= \sum_{t=1}^{T}\mathbb{E}\left[\langle\nabla f(x_t)-\nabla f(x(t-\tau_t)),\, x_{t+1}-x^*\rangle\right]\\ &\le \bar\tau GR + \frac{LC_1}{2} + \frac{LC_2}{2}\log T\end{aligned}$$

where

$$C_1 = \frac{G^2\bar\tau(\bar\tau+1)(2\bar\tau+1)^2}{3(L^2+c^2)} \quad \text{and} \quad C_2 = \frac{G^2(4\bar\tau+3)(\bar\tau+1)}{3c^2}$$

Proof. This proof is an adaptation of Lemma 4 and Corollary 1 of Agarwal and Duchi [1]. First, we exploit convexity of f to help analyze the gradient differences using the four-point identity (A.8):

$$\begin{aligned}&\langle \nabla f(x_t) - \nabla f(x(t-\tau_t)),\, x_{t+1} - x^* \rangle \\ &= D_f(x^*, x_t) - D_f(x^*, x(t-\tau_t)) - D_f(x_{t+1}, x_t) + D_f(x_{t+1}, x(t-\tau_t)).\end{aligned} \tag{A.19}$$

Since ∇f is L-Lipschitz, we further have

$$f(x_{t+1}) \le f(x(t-\tau_t)) + \langle \nabla f(x(t-\tau_t)),\, x_{t+1} - x(t-\tau_t) \rangle + \tfrac{L}{2}\|x(t-\tau_t) - x_{t+1}\|^2.$$

By definition of a Bregman divergence, we also have

$$D_f(x_{t+1}, x(t-\tau_t)) = f(x_{t+1}) - f(x(t-\tau_t)) - \langle \nabla f(x(t-\tau_t)),\, x_{t+1} - x(t-\tau_t) \rangle,$$

which, upon using using A.7, immediately yields the bound

$$D_f(x_{t+1}, x(t-\tau_t)) \le \tfrac{L}{2}\|x(t-\tau_t) - x_{t+1}\|^2.$$

Dropping the negative term $D_f(x_{t+1}, x_t)$ from (A.19) and summing over t, we then obtain

$$\begin{aligned}&\sum_{t=1}^{T} \langle \nabla f(x_t) - \nabla f(x(t-\tau_t)),\, x_{t+1} - x^* \rangle \\ &\le \sum_{t=1}^{T} [D_f(x^*, x_t) - D_f(x^*, x(t-\tau_t))] + \frac{L}{2}\sum_{t=1}^{T} \|x_{t+1} - x(t-\tau_t)\|^2.\end{aligned}$$

Notice that the first sum partially telescopes, leaving only the terms not received by the server within the first T iterations. Thus, we obtain the bound

$$\sum_{t: t+\tau_t > T} D_f(x^*, x_t) + \frac{L}{2}\sum_{t=1}^{T} \|x_{t+1} - x(t-\tau_t)\|^2. \tag{A.20}$$

We bound both each of the terms in (A.20) in turn below.

To bound the contribution of the first term in expectation, compute the expected cardinality

$$\mathbb{E}[|\{t : t+\tau_t > T\}|] = \sum_{t=1}^{T} \Pr(\tau_t > T - t), \tag{A.21}$$

Assuming delays uniform on $\{0, 2\bar{\tau}\}$ bounding this cardinality is easy, since

$$\Pr(\tau_t > T - t) = \begin{cases} 0 & T - t > 2\bar{\tau}, \\ \frac{2\bar{\tau} - T + t}{2\bar{\tau} + 1} & \text{otherwise}. \end{cases}$$

Assuming that $2\bar{\tau} + 1 < T$, (A.21) becomes (unsurprisingly)

$$\sum_{s=1}^{2\bar{\tau}} \frac{2\bar{\tau} - s}{2\bar{\tau} + 1} = \frac{(4\bar{\tau} - 2\bar{\tau})(2\bar{\tau} + 1)}{2(2\bar{\tau} + 1)} = \bar{\tau}.$$

From definition of a Bregman divergence we immediately see that

$$0 \le D_f(x^*, x_t) \le -\langle \nabla f(x_t),\, x^* - x_t \rangle \le \|\nabla f(x_t)\|\|x^* - x_t\| \le GR.$$

Thus, the contribution of the first term in (A.20) is bounded in expectation by by $\bar{\tau}GR$.

To bound the contribution of the second term, use convexity of $\|\cdot\|^2$ to obtain

$$\begin{aligned}
&\|x_{t+1} - x(t-\tau_t)\| \\
=&\|x_{t+1} - x_t + x_t - x(t-1) + \cdots + x(t-\tau_t+1) - x(t-\tau_t)\|^2 \\
\leq&(\tau_t+1)^2 \sum_{s=0}^{\tau_t} \tfrac{1}{\tau_t+1} \|x_{t+1-s} - x_{t-s}\|^2 \\
=&(\tau_t+1) \sum_{s=0}^{\tau_t} \|\Pi_{\mathcal{X}}\big(x(t-s) - \alpha(t-s,\tau_{t-s})g(t-s,\tau_{t-s})\big) - \Pi_{\mathcal{X}}(x(t-s))\|^2 \\
\leq&(\tau_t+1)G^2 \sum_{s=0}^{\tau_t} \alpha(t-s,\tau_{t-s})^2.
\end{aligned}$$

Conditioned on the delay τ_t we have

$$\mathbb{E}[\|x_{t+1} - x(t-\tau_t)\|^2 | \tau_t] \leq (\tau_t+1)G^2 \sum\nolimits_{s=0}^{\tau_t} \mathbb{E}[\alpha(t-s,\tau_{t-s})^2].$$

Under the uniform or scaled assumptions on delays, we obtain similar bounds on the above quantity.

Consider now the expectation

$$\begin{aligned}
\mathbb{E}[\alpha(t-s,\tau(t-s))^2] &= \mathbb{E}[\frac{1}{L^2 + c^2((t-s)+\tau(t-s)) + 2Lc\sqrt{t-s+\tau(t-s)}}] \leq \frac{1}{L^2 + c^2(t-s)} \\
\implies \text{ if } \tau_t = l,\ &\sum_{s=0}^{\tau_t} \mathbb{E}[\alpha(t-s,\tau_{t-s})^2] \leq \sum_{s=0}^{l} \frac{1}{L^2 + c^2(t-l)} = \frac{l+1}{L^2 + c^2(t-l)}.
\end{aligned}$$

Thus, for $t > 2\bar{\tau}$, we have the following bound

$$\begin{aligned}
\mathbb{E}[\|x_{t+1} - x(t-\tau_t)\|^2] &\leq G^2 \sum_{l=0}^{2\bar{\tau}} \frac{1}{2\bar{\tau}+1} \frac{(l+1)^2}{L^2 + c^2(t-l)} \\
&\leq \frac{G^2}{(2\bar{\tau}+1)(L^2 + c^2(t-2\bar{\tau}))} \sum_{l=0}^{2\bar{\tau}} (l+1)^2 \\
&= \frac{G^2(4\bar{\tau}+3)(\bar{\tau}+1)}{3(L^2 + c^2(t-2\bar{\tau}))}.
\end{aligned}$$

and for $t \leq 2\bar{\tau}$, we have

$$\begin{aligned}
\mathbb{E}[\|x_{t+1} - x(t-\tau_t)\|^2] &\leq G^2 \sum_{l=0}^{t-1} P_t(l) \frac{(l+1)^2}{L^2 + c^2(t-l)} \\
&\leq G^2 \sum_{l=0}^{t-1} \frac{(l+1)^2}{L^2 + c^2} \\
&= \frac{G^2 t(t+1)(2t+1)}{6(L^2 + c^2)}.
\end{aligned}$$

Now adding up over $t = 1$ to T, we have

$$\sum_{t=1}^{T} \mathbb{E}[\|x_{t+1} - x(t-\tau_t)\|^2] \leq C_1 + C_2 \log T$$ □

Lemma A.6. *Assuming scaled delays, we have the bound*

$$\begin{aligned}
\sum_{t=1}^{T} \mathbb{E}[\Gamma(t)] &= \sum_{t=1}^{T} \mathbb{E}\left[\langle \nabla f(x_t) - \nabla f(x(t-\tau_t)),\, x_{t+1} - x^* \rangle\right] \\
&\leq GR\left(1 + \sum_{t=1}^{T-1} \frac{B_t^2}{(T-t)^2}\right) + LG^2 \sum_{t=1}^{T} \frac{B_t^2 + 1 + \bar{\tau}_t}{L^2 + c^2(1-\theta_t)t}.
\end{aligned}$$

Proof. We build on Corollary 1 of [1], and proceed as in Lemma A.5 to bound the terms in (A.20) separately. For the first term, we bound the expected cardinality using Chebyshev's inequality and Assumption 2.5 (B):

$$\mathbb{E}[|\{t : t+\tau_t > T\}|] = \sum_{t=1}^{T} \Pr(\tau_t > T-t) \leq 1 + \sum_{t=1}^{T-1} \frac{\mathbb{E}[\tau_t^2]}{(T-t)^2} = 1 + \sum_{t=1}^{T-1} \frac{B_t^2}{(T-t)^2}$$

To bound the second term, we again follow Lemma A.5 to obtain

$$\mathbb{E}[\|x_{t+1} - x(t-\tau_t)\|^2|\tau_t] \leq (\tau_t+1)G^2 \sum\nolimits_{s=0}^{\tau_t} \mathbb{E}[\alpha(t-s,\tau_{t-s})^2].$$

$$\begin{aligned}\mathbb{E}[\alpha(t-s,\tau(t-s))^2] &= \mathbb{E}[\frac{1}{L^2 + c^2((t-s)+\tau(t-s)) + 2Lc\sqrt{t-s+\tau(t-s)}}] \\ &\leq \frac{1}{L^2+c^2(t-s)},\end{aligned}$$

which yields the bound (since $0 \leq s \leq \tau_t$)

$$\mathbb{E}[\|x_{t+1} - x(t-\tau_t)\|^2|\tau_t] \leq \frac{G^2(\tau_t+1)^2}{L^2+c^2(t-\tau_t)}$$

Now adding up over $t=1$ to T consider

$$G^2 \sum\nolimits_{t=1}^{T} \frac{(\tau_t+1)^2}{L^2+c^2(t-\tau_t)},$$

so that taking expectation (over τ_t) we then obtain

$$\sum_{t=1}^{T} \mathbb{E}[\|x_{t+1} - x(t-\tau_t)\|^2] \leq G^2 \sum_{t=1}^{T} \mathbb{E}\left[\frac{(\tau_t+1)^2}{L^2+c^2(t-\tau_t)}\right].$$

Using our assumption that $\tau_t < \theta_t t$ for $\theta_t \in (0,1)$, we have in particular that

$$\begin{aligned}&G^2 \sum_{t=1}^{T} \mathbb{E}\left[\frac{(\tau_t+1)^2}{L^2+c^2(t-\tau_t)}\right] \\ \leq &G^2 \sum_{t=1}^{T} \frac{1}{L^2+c^2(1-\theta_t)t} \mathbb{E}[(\tau_t+1)^2] \\ \leq &G^2 \sum_{t=1}^{T} \frac{B_t^2+1+\bar{\tau}_t}{L^2+c^2(1-\theta_t)t}\end{aligned}$$

□

Lemma A.7. *Let the step-offsets be $\eta(t,\tau_t) = c\sqrt{t+\tau_t}$. For any delay distribution we have*

$$\sum_{t=1}^{T} \mathbb{E}[\Sigma(t)] \leq \frac{\sigma^2}{c}\sqrt{T}.$$

Proof. From Assumption 2.2 on the variance of stochastic gradients, it follows that

$$\mathbb{E}[\Sigma(t)] = \mathbb{E}\left[\tfrac{1}{2\eta(t,\tau_t)}\|\nabla f(x(t-\tau_t)) - g(t-\tau_t)\|^2\right] \leq \frac{\sigma^2}{2}\mathbb{E}\left[\eta(t,\tau_t)^{-1}\right].$$

Plugging in $\eta(t,\tau_t) = c\sqrt{t+\tau_t}$, clearly the bound

$$\frac{1}{c}\mathbb{E}[(t+\tau_t)^{-1/2}] = \frac{1}{c}\sum_{s=0}^{t-1} P(s)\frac{1}{\sqrt{t+s}} \leq \frac{1}{c\sqrt{t}}, \tag{A.22}$$

holds for any delay distribution. Summing up over t, we then obtain

$$\sum_{t=1}^{T} \mathbb{E}[\Sigma(t)] \leq \frac{\sigma^2}{2c}\sum_{t=1}^{T}\frac{1}{\sqrt{t}} \leq \frac{\sigma^2}{c}\sqrt{T}.$$

□

B More general step-sizes

If we use the offsets $\eta_t = c(t+\tau_t)^\beta$, where $0 < \beta < 1$, we obtain slightly more general step sizes that fit within our framework. The *only* benefit of considering stepsizes other than $\beta = 1/2$ is because they allow us to tradeoff the contributions of the various terms in the bounds, and for a larger value of β for instance, we will obtain smaller step sizes, which can be beneficial in high noise regimes, at least in the initial iterations. The theoretical sweet-spot (in terms of dependence on T), is, however $\beta = 1/2$, the choice analyzed above. We summarize below the impact of these steps sizes for non-uniform scaled delays; the uniform case is even simpler. For simplicity, we do not bound the terms as tightly as for the special case $\beta = 1/2$.

Lemma B.1. *Assume that τ_t satisfies Assumption 2.5 (B) and $\eta_t = c(t+\tau_t)^\beta$ and $0 < \beta < 1$. Then,*

$$\mathbb{E}[z_t^+] \leq \frac{cR^2\beta(\bar{\tau}_t + 1)}{(t-1)^{1-\beta}} \tag{B.1}$$

$$\mathbb{E}[\|x_{t+1} - x(t-\tau_t)\|^2] \leq \frac{G^2(\tau_t+1)^2}{L^2 + c^2(t-\tau_t)^{2\beta}} \tag{B.2}$$

$$\mathbb{E}[\eta(t,\tau_t)^{-1}] \leq \frac{1}{ct^\beta}. \tag{B.3}$$

Proof. Proceeding as in Lemma A.4 we bound

$$\begin{aligned}
\mathbb{E}[z_t^+] &= c\sum_{l=0}^{t-1}\sum_{s=0}^{l} P_t(l)P_{t-1}(s)\left((t+l)^\beta - (t-1+s)^\beta\right) \\
&\leq c\sum_{l=0}^{t-1}\sum_{s=0}^{l} P_t(l)P_{t-1}(s)\beta\frac{l+1-s}{(t-1+s)^{1-\beta}} \\
&\leq c\beta\sum_{l=0}^{t-1}\sum_{s=0}^{l} P_t(l)P_{t-1}(s)\frac{l+1}{(t-1)^{1-\beta}} \\
&\leq c\beta\sum_{l=0}^{t-1} P_t(l)\frac{l+1}{(t-1)^{1-\beta}} = \frac{c\beta(\bar{\tau}_t+1)}{(t-1)^{1-\beta}}.
\end{aligned}$$

where the first inequality follows from concavity if t^β, the second one since $\frac{l+1-s}{(t-1+s)^{1-\beta}}$ is decreasing in s, while the third is clear as P_{t-1} is a probability.

Next, we bound (B.2). Proceeding as in Lemma A.6, we obtain the bounds

$$\begin{aligned}
\mathbb{E}[\alpha(t-s,\tau_{t-s})^2] &\leq \frac{1}{L^2 + c^2(t-s)^{2\beta}} \\
\implies \mathbb{E}[\|x_{t+1} - x(t-\tau_t)\|^2|\tau_t] &\leq \frac{G^2(\tau_t+1)^2}{L^2 + c^2(t-\tau_t)^{2\beta}}
\end{aligned}$$

Finally, the bound on (B.3) is trivial; since $\eta_t^{-1} = c^{-1}(t+\tau_t)^{-\beta}$, we have

$$\frac{1}{c}\mathbb{E}[(t+\tau_t)^{-\beta}] = \frac{1}{c}\sum_{s=0}^{t-1} P_t(s)\frac{1}{(t+s)^\beta} \leq \frac{1}{ct^\beta}. \qquad \square$$

Using these key bounds, we can defined full versions of Lemmas A.4, A.6, and A.7, where we finally we will need a bound of the form

$$\sum_{t=1}^{T}\frac{1}{t^\beta} \leq 1 + \int_0^T t^{-\beta}dt = 1 + \frac{(T^{1-\beta}-1)}{1-\beta} \leq \frac{1}{1-\beta}T^{1-\beta}.$$