Delay-Adaptive Distributed Stochastic Optimization

Zhaolin Ren,[1] **Zhengyuan Zhou,**[*2,4] **Linhai Qiu,**[3] **Ajay Deshpande,**[4] **Jayant Kalagnanam**[4]
[1]Harvard University, [2]New York University, [3]Google Inc., [4]IBM Research

Abstract

In large-scale optimization problems, *distributed asynchronous stochastic gradient descent* (DASGD) is a commonly used algorithm. In most applications, there are often a large number of computing nodes asynchronously computing gradient information. As such, the gradient information received at a given iteration is often stale. In the presence of such delays, which can be unbounded, the convergence of DASGD is uncertain. The contribution of this paper is twofold. First, we propose a delay-adaptive variant of DASGD where we adjust each iteration's step-size based on the size of the delay, and prove asymptotic convergence of the algorithm on variationally coherent stochastic problems, a class of functions which properly includes convex, quasi-convex and star-convex functions. Second, we extend the convergence results of standard DASGD, used usually for problems with bounded domains, to problems with unbounded domains. In this way, we extend the frontier of theoretical guarantees for distributed asynchronous optimization, and provide new insights for practitioners working on large-scale optimization problems.

1 Introduction

In recent years, rapid advances in computing infrastructures have led to a significant increase in the use of distributed stochastic optimization. There has correspondingly been intensive work studying distributed stochastic optimization, such as (Chaturapruek, Duchi, and Ré 2015), (Karakus et al. 2017), (Damaskinos et al. 2018), (Wu et al. 2018),(Assran et al. 2018), (Cutkosky and Busa-Fekete 2018), (Koloskova, Stich, and Jaggi 2019), (Yu and Jin 2019), (Xie, Koyejo, and Gupta 2019).

Many optimization algorithms in distributed settings are first-order methods involving multiple computing nodes working asynchronously to perform stochastic gradient descent. For brevity, we will refer to such algorithms as *distributed asynchronous stochastic gradient descent* (DASGD) in this paper. Two common architectures on which DASGD are deployed are 1) a "master-slave" architecture where each worker independently computes a noisy gradient for the master node, and 2) a multiprocessor shared-memory architecture, where there is no master node and workers asynchronously update a parameter in some shared memory. Both architectures are vulnerable to delays in the gradient computation. Such delays are of particular concern in volunteer computing grids, where there is no upper bound on the workers' delay.

Related Work and Our Contributions

Our paper addresses the robustness of DASGD-type algorithms in the presence of large, unbounded delays resulting from either the master-slave or shared memory parallel computing architectures. In distributed deterministic optimization, asynchronous gradient descent has been shown to solve convex problems, even if delays scale sublinearly over time (Bertsekas and Tsitsiklis 2003). For stochastic convex problems, recent results by ((Agarwal and Duchi 2011), (Recht et al. 2011), (Chaturapruek, Duchi, and Ré 2015)) have derived convergence rates for DASGD when the delay is bounded. A delay-adaptive DASGD variant for convex problems was proposed in (Sra et al. 2015), where the algorithm's step-size depends on the actual delay received. However, the authors of (Sra et al. 2015) assumed that the delays are either bounded or have finite first and second moments, whereas in many applications the delays might not be bounded. An algorithm for problems with unbounded delays for both convex and nonconvex problems was introduced in (Peng et al. 2019). However, the proposed step-size in (Peng et al. 2019) is not delay-adaptive, and the authors there assumed finite-mean i.i.d delays, while we make no distributional assumption. Beyond convexity, the class of *variationally coherent* (VC) problems, which properly includes convex, quasi-convex, and star-convex objectives, was introduced in ((Zhou et al. 2017a), (Zhou et al. 2020a)), and subsequently generalized to multi-player game settings ((Zhou et al. 2017c), (Zhou et al. 2017b), (Zhou et al. 2018a), (Mertikopoulos and Zhou 2019)). For VC problems on a bounded domain, ((Zhou et al. 2018b), (Zhou et al. 2020b)) proved that DASGD converges almost surely to a global minimizer, even when the delays between gradient updates and requests grow at a polynomial rate. For synchronous stochastic optimization of variationally-coherent-like problems with unbounded domains, (Lelong 2008) proposed a random truncation approach that achieves convergence.

Our main contributions are twofold: first, we propose a delay-adaptive variant of DASGD for VC problems whose convergence is robust to any delay sequence from either the master-slave or shared-memory architecture, going be-

[*]This author gratefully acknowledges the support of the IBM Goldstine Fellowship.

yond the polynomially growing delay allowed in (Zhou et al. 2020b). Developing such a theoretical guarantee is crucial given the arbitrary nature of delays in real-life computing grids. Next, we develop a modified version of DASGD that converges for VC problems with unbounded domains, reposing on the truncation technique pioneered in (Lelong 2008). Both contributions are to the best of the authors' knowledge novel in the literature.

2 Problem Setup

We will consider the following problem:

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to} \quad x \in \mathcal{X}, \end{aligned} \tag{Opt}$$

where the objective $f : \mathcal{X} \to \mathbb{R}$ is of the form

$$f(x) = \mathbb{E}[F(x;\omega)]$$

for some random function $F : \mathcal{X} \times \Omega \to \mathbb{R}$, where we let $(\Omega, \mathcal{F}, \mathbb{P})$ be some underlying probability space, and $\mathcal{X}$ is a compact, convex subset of $\mathbb{R}^d$.

Assumptions

We make the following assumptions:

Assumption 1. F satisfies the following:

1. $F(x;\omega)$ is differentiable in x for almost every $\omega \in \Omega$.
2. $\nabla F(x;\omega)$ has bounded second moments, that is, $\mathbb{E}[\|\nabla F(x;\omega)\|_2^2] < \infty$ for all $x \in \mathcal{X}$.
3. $\nabla F(x;\omega)$ is Lipschitz continuous in the mean on any compact set $\mathcal{X}$: for any $\mathcal{X}$, there exists some $L_{\mathcal{X}}$ such that $\mathbb{E}[\nabla F(x;\omega)]$ is $L_{\mathcal{X}}$-Lipschitz on $\mathcal{X}$.

Assumption 2. The optimization problem (Opt) is *variationally coherent in the mean*, which means that

$$\mathbb{E}[\langle F(x;\omega), x - x^*\rangle] \geq 0, \tag{VC}$$

for all $x \in \mathcal{X}$ and $x^* \in \mathcal{X}^*$, with equality if and only if $x = x^*$.

Variational coherence is a wide class of optimization problems that properly includes convex, quasi-convex, and star-convex functions. We note that (VC) is a significantly weaker condition than convexity, as it has no information on generic point pairs. For examples of such functions, see (Bottou 1998), (Zhou et al. 2020a).

DASGD: A Framework for Asynchronous Optimization Our goal in this paper is to deal with delays that occur either in 1) master-slave or 2) multi-processor systems with shared memory.

1. In distributed master-slave gradient descent, workers asynchronously compute stochastic gradients and then send them to the master. Meanwhile, the master updates the global state of the system and pushes the updated state back to the workers.
2. In a multi-processor system with shared memory, all processors access a global, shared memory, which contains both the data needed to compute a stochastic gradient as well as the current iterate. Each processor independently and asynchronously reads the current global iterate, computes a stochastic gradient, and then updated the shared global iterate.

Both systems are common asynchronous computations architectures used in practice, and are described with more detail in (Zhou et al. 2020b).

We can unify both these architectures under the DASGD framework in Algorithm 1.

Algorithm 1: Distributed Asynchronous Stochastic Gradient Descent

Require: $Y_0 \in \mathbb{R}^d$
$n \leftarrow 0$
repeat
  $X_n = \mathrm{Proj}_{\mathcal{X}}(Y_n)$
  $Y_{n+1} = Y_n - \alpha_{n+1}(\nabla F(X_{s(n)}; \omega_{n+1}))$
  $n \leftarrow n + 1$
until *end*;
return *solution candidate* X_n

In DASGD, note that $s(n)$ is the iteration of the gradient used at iteration n. Let $d_n = n - s(n)$ represent the delay. We say that the delay is (i) linear, (ii) polynomial, (iii) exponential respectively if d_n is (i) linear, (ii) polynomial, (iii) exponential as a function of $s(n)$. For instance, if $s(n) = \sqrt{n}$, so that $d_n = n - \sqrt{n}$, we say that the delay received at iteration n is polynomial.

3 Adapting Step-size to Delay

Intuitively, using a fixed step-size sequence whilst ignoring the delay information might jeopardize convergence, given the potential for unbounded delay. Therefore, it is important to develop a delay-adaptive step-size sequence that is robust to bad delay. In this section, we first state and prove two elementary results on the delay sequence resulting from either a master-slave or shared-memory architecture, before providing intuition for our proposed delay-adaptive stepsize.

Results on Delay Sequence

We start by better understanding the delay generated from either (i) the master-slave architecture, or (ii) the multi-processor system with shared memory. We show in fact that the delays generated are for the most part not too stale.

To see this, assume that there are K workers computing the gradients, where $K \in \mathbb{Z}^+$. In a master-slave system, x_t from any particular time t is used in the computation of future gradients precisely once. Meanwhile, in a shared-memory system, multiple workers can access the same x_t. Then, while the K gradients computed by the K workers are different (due to different stochasticity), they could all stem from the same global iterate x_t. Hence, x_t from any particular time t can be used in the computation of gradients for

future updates at most K times. This observation gives rise to the following result.

Lemma 3.1. *Suppose there are K workers in a master-slave or shared-memory system. Then, for any $N > 0$, there exists at least $N/2$ indices $n \in [N]$ such that $s(n) \geq n/(2K)$.*

Proof. To see this, note that the first $N/(2K)$ iterates can each be used at most K times, so in total at most $N/2$ iterations $n \in [N]$ can use gradients computed from the first $N/(2K)$ iterates. Hence, for the remaining indices $n \in [N]$ (and there are at least $N/2$ such indices),

$$s(n) \geq N/(2K) \geq n/(2K).$$

□

This shows that more often than not, the delay is not worse than linear. This is an interesting observation in its own right. It also implies that there are infinitely many n such that $s(n) \geq n/(2K)$, which we will use later to prove convergence of DASGD along a subsequence.

Lemma 3.2. *As n goes to infinity, so does $s(n)$.*

Proof. To see this, suppose otherwise. Then, there exists some $C > 0$ such that $s(n) \leq C$ for all n. Since there are only K workers, it follows that only $C \cdot K$ iterates will ever be used in the computation of gradients, a contradiction. □

A Delay-adaptive Step-size Proposal

In developing a new delay-adaptive step-size, we leverage on insight from (Zhou et al. 2020b). There, to deal with polynomially growing delay, the authors used the step-size sequence

$$\alpha_{n+1} = \frac{1}{n \log n \log\log n}.$$

This suggests that we might want to build our step-size on top of that, and incorporate a delay-adaptive adjustment term. Suppose $0 < c < 1$. Consider the step-size

$$\alpha_{n+1} = \frac{1}{n \log n \log\log n + n^{c(\log(n)/\log s(n))}} \quad (1)$$

We can view $n^{c(\log(n)/\log s(n))}$ as a delay-adjustment term that varies with the delay. For concreteness we pick $c = 2/3$ in our analysis during the rest of the paper.

To see why we picked this step-size, observe the size of α_n when $s(n)$ is small, i.e. the delay is large. Suppose $s(n) = O(\sqrt{n})$. Then,

$$n^{(2/3)(\log(n)/\log s(n))} = o(n^{4/3}).$$

So when the delay is unbounded and grows faster than polynomially, α_n is on the order $O(1/n^{4/3})$, which is summable.

Next, note that when the delay is linear, i.e. $s(n)$ is on the order $o(n)$, $n^{(2/3)(\log(n)/\log s(n))} = O(n^{2/3})$, so the step-size α_n is on the order o$(1/(n \log n \log\log n))$, which is not summable

We formalize the above in the following result generalized for any $0 < c < 1$. Details of the proof can be found in the supplementary material.

Lemma 3.3. *Let*

$$\alpha_{n+1} = \frac{1}{n \log n \log\log n + n^{c(\log(n)/\log s(n))}}.$$

Then,

1. *On the subsequence whose indices satisfy $s(n) = O(n^\ell)$, where $\ell < c$, the step-sizes are summable:*
$$\sum_{n=0, s(n)=O(n^\ell)}^{\infty} \alpha_{n+1} < \infty$$
2. *On the subsequence whose indices satisfy $s(n) \geq n/(2K)$, where the step-sizes are not summable:*
$$\sum_{n=0, s(n)\geq n/(2K)}^{\infty} \alpha_{n+1} = \infty$$

4 Convergence Analysis

For clarity, we will write down the delay-adaptive variant of DASGD we are proposing.

Algorithm 2: Delay-Adaptive DASGD

Require: $Y_0 \in \mathbb{R}^d$
$n \leftarrow 0, 0 < c < 1$
repeat
  $X_n = \mathrm{Proj}_{\mathcal{X}}(Y_n)$
  $\alpha_{n+1} = \frac{1}{n \log n \log\log n + n^{c(\log(n)/\log s(n))}}$
  $Y_{n+1} = Y_n - \alpha_{n+1}(\nabla F(X_{s(n)}; \omega_{n+1}))$
  $n \leftarrow n + 1$
until *end*;
return *solution candidate* X_n

Energy Function

Consider the energy function

$$E(y) = \inf_{x^* \in \mathcal{X}^*} E_{x^*}(y),$$

where $E_{x^*}(y) = \|x^*\|_2^2 - \|\mathrm{Proj}_{\mathcal{X}}(y)\|_2^2 + 2\langle y, \mathrm{Proj}_{\mathcal{X}}(y) - x^*\rangle$.

The energy function satisfies the following properties:

Lemma 4.1.

1. $E(y) \geq 0$ *with equality if and only if* $\mathrm{Proj}_{\mathcal{X}}(y) \in \mathcal{X}^*$
2. *Let* $\{y_n\}_{n=1}^{\infty}$ *be a sequence. Then,* $\lim_{n\to\infty} E(y_n) = 0$ *if and only if* $\mathrm{Proj}_{\mathcal{X}}(y_n) \to \mathcal{X}^*$ *as* $n \to \infty$

Observe calling $E(y)$ an energy function is justified since $E(y)$ is always nonnegative. The keypoint here is this: since $E(Y_n) \to 0$ if and only if $X_n \to \mathcal{X}^*$, the energy function $E(y)$ provides us a convenient way to characterize and prove convergence.

This next result allows us to bound the change across an iteration of the energy function, and will prove essential in the technical analysis.

Lemma 4.2. *Fix any $x^* \in \mathcal{X}^*$*

1. $\|\mathrm{Proj}_{\mathcal{X}}(y) - \hat{y}\|_2^2 - \|\mathrm{Proj}_{\mathcal{X}}(\hat{y}) - \hat{y}\|_2^2 \leq \|y - \hat{y}\|_2^2$, *for any $y, \hat{y} \in \mathbb{R}^d$*
2. $E_{x^*}(y+\Delta y) - E_{x^*}(y) \leq 2\langle \Delta y, \mathrm{Proj}_{\mathcal{X}}(y) - x^*\rangle + \|\Delta y\|_2^2$, *for any $y, \Delta y \in \mathbb{R}^d$.*

Convergence Analysis

We can write the DASGD updates as follows:

$$X_n = \text{Proj}_{\mathcal{X}}(Y_n)$$
$$Y_{n+1} = Y_n - \alpha_{n+1}(\nabla f(X_n) + B_n + U_{n+1})$$

where $B_n = \nabla f(X_{s(n)}) - \nabla f(X_n)$ is the delay error term and $U_{n+1} = \nabla F(X_{s(n)}; \omega_{n+1}) - \nabla f(X_{s(n)})$ is the stochastic term in the gradient computation. In addition, we use the adaptive step-size from equation (1).

In our proof (see supplement), we first prove almost sure (a.s.) convergence along a subsequence before extending to full a.s. convergence, a standard technique for convergence proofs of discrete iterates.

Convergence Along a Subsequence

Proposition 4.2.1. *Under assumptions 1 - 2, and lemmas 3.1 and 3.2, delay-adaptive DASGD admits a subsequence X_{n_k} that converges to $\mathcal{X}^*$ almost surely as $k \to \infty$.*

The proof is technical, and provided in full in the supplement. We highlight that the key here is to show a bound on the delay term B_n, which is almost surely bounded above by $O(\log\log\log n)$ when the delay is allowed to be exponential. A telescoping argument then allows us to show that $E(Y_{n+1}) - E(Y_0) \to -\infty$, a contradiction.

Extending to Convergence The main result of delay-adaptivity in the stochastic context is the following:

Theorem 4.3. *Under Assumptions 1 to 2, and lemmas 3.1 and 3.2, the global state variable X_n of delay-adaptive DASGD converges a.s. to the solution set $\mathcal{X}^*$ of Opt.*

The stochastic noise makes extending subsequential convergence to full convergence tricky. To get around this, we will write down an ODE that approximates the DASGD iterates trajectory and show convergence of the ODE flow to the optima set. Then, we will relate the DASGD iterates to the ODE trajectory to prove convergence.

We note that we can view the DASGD updates as a discretization of the mean-flow ODE

$$\dot{y} = -\nabla f(\text{Proj}_{\mathcal{X}}(y)) \tag{2}$$

Compactness of $\mathcal{X}$ and Lipschitz-continuity of ∇f ensure a unique trajectory for y. We can define $P : \mathbb{R}^+ \times \mathbb{R}^d \to \mathbb{R}^d$ as follows, where $P(t, y_0)$ denotes the state of the system when starting from y_0 after running for a time of t.

The idea is that asymptotically, the (random) iterates Y_1, Y_2, $\ldots$, Y_n will hug close to the mean-field ODE specified above in (2). Using the VC condition allows us to prove that $P(t, y_0) \to x^*$ as $t \to \infty$. This then allows us to prove convergence of the iterates.

To make this argument more precise, we next introduce the following three objects

1. The DASGD iterates $Y_1, Y_2, \ldots, Y_n$
2. An affine curve $A(t)$ interpolating the iterates $Y_1, Y_2, \ldots, Y_n$ where Y_r is placed at $\sum_{s=0}^{r} \alpha_s$ (recall that here the step-size α_r is still delay-adaptive)
3. the curve given by the ODE flow $P(t, y)$

In order to show that the interpolated curve $A(t)$ is close to the mean-field ODE asymptotically, we consider the following notion of an *asymptotic pseudotrajectory* (APT), introduced in (Benaïm 1999):

Definition 1. *A continuous function $s : \mathbb{R}^+ \to \mathbb{R}^d$ is an APT for P if every $T > 0$,*

$$\lim_{t\to\infty} \sup_{h \geq T} d(s(t+h), P(h, s(t))) = 0$$

In the absence of delay-adaptivity, Theorem 4.12 in (Zhou et al. 2020b) showed that as long $A(t)$ can be shown to be an APT for P, assuming there exists a convergent subsequence, we can prove convergence of the DASGD iterates. Since we have already shown a convergent subsequence in the delay-adaptive setting, we just need to show that $A(t)$ is still an APT for P under the delay-adaptive setting. We can prove this by relying on the fact that when $s(n) = O(\sqrt{n})$, i.e. when there is a possibly bad delay, the delay-adaptive stepsize is summable. The precise details, using ideas from (Benaïm 1999), are in the supplement.

5 Extending DASGD to Unbounded Domain

While DASGD guarantees are often provided for problems with bounded domains, to date, there has been no result in the literature proving convergence of DASGD for problems with unbounded domains. In this section, we propose a novel variant of DASGD that achieves such guarantees for variationally coherent problems with unbounded domains.

We will require the following assumption, as the existing proof technique only works for sublinearly growing decay.

Assumption 3. We assume that the delay sequence $\{d_n\}_n$ satisfies

$$d_n \leq \beta n^c \text{ for some } \beta > 0, 0 \leq c < 1.$$

For clarity, we will also assume that the minimizer x^* of f is unique.

In an unconstrained setting, ensuring boundedness of the iterates is problematic. To deal with this issue, we force the algorithm to remain in an increasing sequence of compact sets $(\mathcal{K}_j)_j$, where the containment $\mathcal{K}_j \subset \mathcal{K}_{j+1}$ is strict. Each time the iterate leaves the current compact set $\mathcal{K}_{\tau_n}$, we increase τ_n by 1, and perform a *truncation* by resetting the iterate to a fixed point in $\mathcal{K}_{\tau_0}$. To differentiate it from vanilla DASGD, we call this new algorithm DASGD-T, where T stands for truncation. Algorithm (3) provides one implementation of the algorithm DASGD-T.

As with DASGD, the master keeps track of a global counter n and increments it every time it updates the current solution candidate X_n. When an iterate exceeds the current compact set $\mathcal{K}_{\tau_n}$, we return the algorithm to its initial iterate X_0. This ensures that the iterates return to some compact set infinitely often, which will prove essential in ensuring boundedness of the iterates. On the other hand, if the tentative iterate stays within $\mathcal{K}_{\tau_n}$, we update Y_n to be our new candidate solution X_{n+1}. We say that the algorithm is in its t-th cycle at time n if $\tau_n = t$.

Algorithm 3: DASGD-T

Require: Initial state $X_0 \in \mathbb{R}^d$, step-size sequence α_n, initial truncation parameter τ_0

$n \leftarrow 0$;
repeat
  if $\tau_{s(n)} < \tau_n$; /* only use gradient from current cycle */ **then**
    $X_{n+1} \leftarrow X_n$;
  else
    $Y_n \leftarrow X_n - \alpha_{n+1} \nabla F(X_{s(n)}, \omega_{s(n)+1})$;
    if $Y_n \in \mathcal{K}_{\tau_n}$ **then**
      $X_{n+1} \leftarrow Y_n$;
      $\tau_{n+1} \leftarrow \tau_n$;
    else
      $X_{n+1} \leftarrow X_0$; /* truncate */
      $\tau_{n+1} \leftarrow \tau_n + 1$; /* increase exploration limit */
    end
  end
  $n \leftarrow n + 1$
until *end*;
return X_n

Note also that if the gradient received at time n came from an iteration $s(n)$ for which $\tau_{s(n)} < s(n)$, we keep the existing iterate and do not use the gradient evaluated at time $\tau_{s(n)}$ to produce a new iterate. The idea behind this is to prevent a stale gradient from an earlier cycle $\tau_{s(n)}$ to affect the algorithm, as the reason why we exited the compact set $\mathcal{K}_{\tau_{s(n)}}$ in the first place could have been because the algorithm was producing iterates that started to diverge. This choice of only using delayed gradient from the current cycle will also help in proving boundedness of the iterates later.

Boundedness of Iterates

The crux of this analysis is showing that the iterates remain a.s. bounded. To do so, we will first show that the delay and noise terms are asymptotically negligible. Using that, we will then show that the iterates have to remain a.s. bounded.

For the purpose of analysis, it is convenient to rewrite the algorithm update in the following way. We have

$$X_{n+1} = X_n - \alpha_{n+1} \nabla f(X_n) - \alpha_{n+1} \delta M_{n+1} + \alpha_{n+1} p_{n+1} \tag{3}$$

where

$$\begin{aligned} \delta M_{n+1} &= \nabla f(X_{s(n)}; \omega) - \nabla f(X_n) \\ &= \underbrace{\left(\nabla f(X_{s(n)}; \omega) - \nabla f(X_{s(n)})\right)}_{\xi_{s(n)}} + \underbrace{\left(\nabla f(X_{s(n)} - \nabla f(X_n)\right)}_{b_n} \end{aligned} \tag{4}$$

and

$$p_{n+1} = \begin{cases} \nabla f(X_n) + \delta M_{n+1} + \dfrac{1}{\alpha_{n+1}}(X_0 - X_n) & \text{if } Y_n \notin \mathcal{K}_{\tau_n} \\ 0 & \text{if otherwise.} \end{cases} \tag{5}$$

We can interpret δM_{n+1} as representing the delay and noise terms, whilst p_{n+1} is the deviation from the noisy gradient descent dynamics resulting from a truncation.

We also introduce a new notation v_n, which we define as follows,

$$v(n) \triangleq \inf \{k : \tau_k = \tau_n, k + d_k \geq s(n)\}.$$

To parse the above, $X_{v(n)}$ is the earliest iterate in the current cycle whose gradient was used in producing any iterate from time $s(n)$ onwards. The particular choice of definition is to make sure that $(v(n))_n$ is an increasing sequence, which is important for the technical analysis.

We now characterize the notion that the delay and noise terms do not matter asymptotically.

Lemma 5.1. *Suppose assumptions (1), (2), and (3) hold, and $\alpha_{n+1} = 1/n$. Then, for all $q > 0$, the series $\sum_n \alpha_{n+1} \delta M_{n+1} \mathbb{I}_{\|X_{v(n):n} - x^*\| \leq q}$ converges a.s.*

We prove this in the supplement. Note that this is where we used the assumption that the delays scale sublinearly.

Next, we show that given that the delay and noise terms do not matter asymptotically, the iterates must remain a.s. bounded.

Proposition 5.1.1. *Suppose assumptions (1), (2), and (3) hold, and $\alpha_{n+1} = 1/n$. If for all $q > 0$, the series $\sum_{n>0} \alpha_{n+1} \delta M_{n+1} \mathbb{I}_{\{\|X_{v(n):n} - x^*\| < q\}}$ converges a.s., then the sequence $(X_n)_n$ remains a.s. in a compact set.*

The proof is provided in the supplement. At a high-level, the proof works by showing that no matter the convergence of the truncation terms involving p_{n+1}, as long as Lemma 5.1 is true, the iterates must remain a.s. bounded.

Convergence of Iterates

Having ascertained boundedness of the iterates, we can now prove convergence.

Theorem 5.2. *Suppose assumptions (1), (2), and (3) hold. Suppose also that the step-size sequence is the following,*

$$\alpha_n = 1/n, \quad \forall n \in \mathbb{N}.$$

Then, a.s., the sequence of iterates the algorithm DASGD-T produces, $(X_n)_n$ converges to x^*.

Remark 5.2.1. *We note that our algorithm is delay-adaptive in the sense that we only use delayed gradients computed in the current cycle of truncation. We can further consider a delay-adaptive step-size of the form*

$$\alpha_n = \frac{1}{n \log n \log\log n + n^{c \log n / \log s(n)}},$$

similar to what we studied earlier. However, in order to highlight the truncation argument, we simplify the analysis by considering a delay-agnostic stepsize $\alpha_n = 1/n$.

The key to showing this is Proposition 5.1.1 (boundedness of iterates) and the VC condition, and we leave details to the supplement.

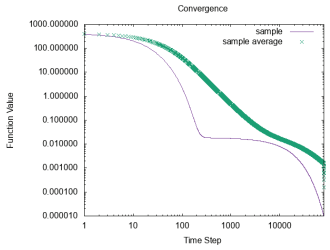

Figure 1: Convergence for f_{Ros}, with no delay (baseline)

6 Numerical Results

Delay-adaptive DASGD

Rosenbrock test We supplement our theoretical analysis with numerical results. We first verify convergence of delay-adaptive DASGD on a standard Rosenbrock test function with $d = 101$ degrees of freedom,

$$f_{\text{Ros}}(x) = \sum_{i=1}^{100} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2], \qquad (6)$$

where $x \in [0, 2]^{101}$. The global minimum of f_{Ros} is located at (1,...,1). While the profile of f_{Ros} is highly nonconvex, its variational coherence can be checked over the constraint set we specified.

In all the cases with different delay functions, we use the same initial condition that was generated randomly but fixed throughout all the runs. For all the cases, we drew noise from a standard multivariate Gaussian distribution when computing the gradient of each time step. We ran 10 trials and averaged the results for each case. We plotted the Rosenbrock function value as well as an average of the function values as the function of the time step.

Figure 1 shows the result with no delay with a fixed step size of $1e^{-4}$ showing smooth convergence, which we use as the baseline for the evaluation. In Figures (2, 3, 4, 5), we simulate DASGD with 1000 workers and compare the results with v.s. without delay-adaptive step sizes. In the sublinear delay case, we used the delay function of unif$[0, 10 \cdot \log(n)]$, where unif[.] is a uniform distribution random number generator. In the linear delay cases, we used the delay function of unif$[0, n/2000]$. In the polynomial delay cases, we used the delay function of unif$[0, n^2/2000000 + n/5000]$. In the exponential delay case, we used the delay function of exp(unif$[0, n/5000]$). In the non-delay-adaptive cases, we used the fixed step size of $1e^{-4}$ as the baseline. In the delay-adaptive cases, we use the step size $1e^{-4}/(n \log n \log\log n + n^{c(\log(n)/\log s(n))})$, where $c = 1/2$, but when the delay is less than 10, we use the non-adaptive step size of $1e^{-4}$ in order to allow faster convergence when the delay is small. In all the test cases with different delay functions, it was observed that the function values either diverged or struggled in converging when using the non-delay-adaptive step size, while using the delay-adaptive step sizes always led to convergence.

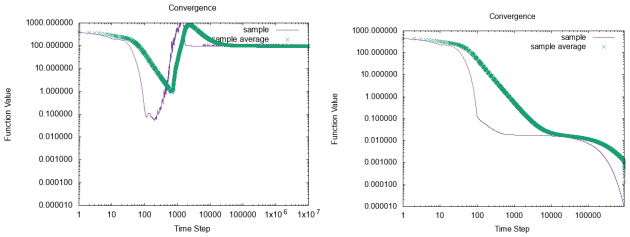

Figure 2: Non-adaptive DASGD (left) vs. delay-adaptive DASGD (right), for f_{Ros}, with sublinear delay

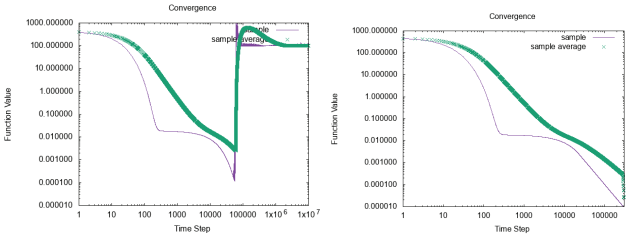

Figure 3: Non-adaptive DASGD (left) vs. delay-adaptive DASGD (right), for f_{Ros}, with linear delay

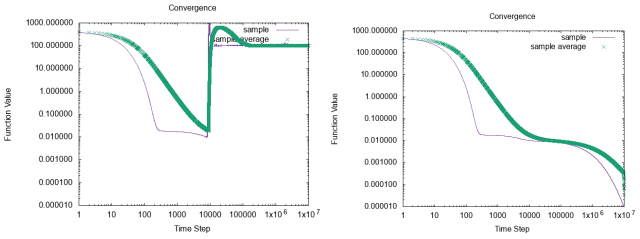

Figure 4: Non-adaptive DASGD (left) vs. delay-adaptive DASGD (right), for f_{Ros}, with polynomial delay

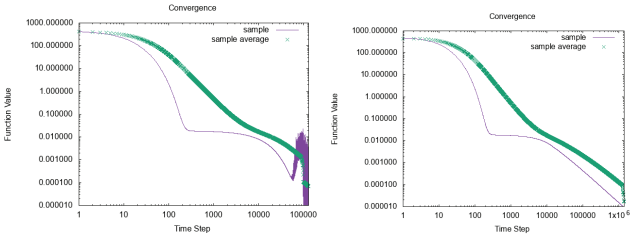

Figure 5: Non-adaptive DASGD (left) vs. delay-adaptive DASGD (right), for f_{Ros}, with exponential delay

	Mean delay-adaptive test accuracy	Mean non-adaptive test accuracy	Mean difference in test accuracy ($\pm$ 1 stdev)
exponential delay	0.892	0.575	0.316 ($\pm$ 0.080)
polynomial delay	0.851	0.622	0.230 ($\pm$ 0.067)
linear delay	0.855	0.743	0.112 ($\pm$ 0.035)

Table 1: MNIST test accuracy, delay-adaptive vs non-delay-adaptive [**baseline test accuracy (no delay)**: 0.907 ± 0.001]

MNIST test We also compared the accuracies of a logistic regression model learned using the delay-adaptive vs non-delay-adaptive DASGD algorithms on the MNIST dataset (LeCun 1998), a standard benchmark for machine learning tasks.

For this comparison, we experimented with linear, polynomial and exponential delay and simulated DASGD on a logistic regression model with K = 10 workers. We also tested the model on a baseline case with no delay, where we used the fixed step size of $1e^{-3}$. For linear delay, we used the delay function of unif$[0, n/50]$. For polynomial delay, we used the delay function of unif$[0, n^2/5000000 + n/50]$. For exponential delay, we used the delay function of exp(unif$[0, n/5000]$). In each case, we used a mini-batchsize of 1, and ran the algorithm for 100000 steps. In the non-delay-adaptive cases, we used the fixed step size of $1e^{-3}$ as the baseline. In the delay-adaptive cases, we use the step size $1e^{-3}/(n \log n \log\log n + n^{c(\log(n)/\log s(n))})$, where $c = 1/2$, but when the delay is less than 100, we use the non-adaptive step size of $1e^{-3}$ to allow faster convergence when the delay is small. For each delay setting, we ran 10 trials, where we ran the delay-adaptive and non-delay-adaptive algorithms on the same (random) delay sequence each trial. In Table 1, the first column shows the average delay-adaptive test accuracy, the second column shows the average non-delay-adaptive test accuracy, whilst the third column shows the average difference between the delay-adaptive test accuracy and the non-delay-adaptive test accuracy across the trials (this is possible since in each trial both delay-adaptive and non-delay-adaptive models use the same delay-sequence). For our baseline model with no delay, we achieved a test accuracy of 0.907 (± 0.001), where 0.001 is the standard deviation. In the three delay settings we considered, we found that the non-delay-adaptive step-size models' test performances deteriorate significantly (between 0.575 to 0.743), while the delay-adaptive step-sizes achieve a level of performance (between 0.855 to 0.892) that is close to the baseline with no delay (0.907). This suggests the empirical effectiveness of our delay-adaptive step-size.

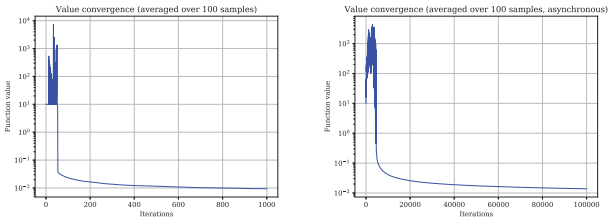

Figure 6: DASGD-T with no delay (left) vs. DASGD-T with $\Theta(\sqrt{n})$ delay (right)

DASGD-T (Unbounded Domains)

To verify the results for DASGD-T, consider the following two-dimensional function

$$f(x, y) = \frac{1}{2}x^2y^2 + \frac{1}{20}(x^2 + y^2).$$

Note that f has a single global minimizer at $(0, 0)$, and a simple calculation shows that it is also variationally coherent, and locally Lipschitz. The motivation for the above function comes from the knowledge that the d-dimensional function,

$$g(x) = \frac{1}{2}\prod_{i=1}^{d} x_i^2,$$

satisfies a weak variational coherence property, defined in (Zhou et al. 2020a). As an aside, a simple calculation shows that f is not quasi-convex. Suppose $z_1 = (1, 0)$, and $(z_2) = (0, 1)$; then, $f(z_1/2 + z_2/2) = (1/40) + (1/32) > 1/20 = \max(f(z_1), f(z_2))$. This can also be seen by visualizing the sublevel sets of the function f.

To carry out the minimization, an asynchronous master-slave framework was used, with delays that scale as $\Theta(\sqrt{n})$, a step-size sequence of $\alpha_n \propto 1/n$, and gradients with stochastic noise drawn from a standard multivariate Gaussian distribution with zero mean and identity covariance. The choice for the initial point was $(10, 10)$, and the truncation sets $\mathcal{K}_j$ was chosen to be $\left\{x : \|x - X_0\|_2 \leq j^2\right\}$. We show plots of value convergence of the function f, averaged over 100 trials, both in the setting with and without delay. While the no-delay case converged faster, we see that the algorithm eventually stabilizes and converges in both cases. This confirms our theoretical analysis that DASGD-T can achieve convergence for a variationally coherent problem with unbounded domain even when there is (sublinear) delay.

7 Conclusion

To conclude, our contributions are twofold. First, we show that a delay-adaptive algorithm can achieve convergence for VC functions under general arbitrary delay sequences, a novel contribution to the literature. Second, we show that we can extend the convergence results of bounded domains to unbounded domains using a truncation algorithm, with the caveat that our current results hold only in the setting with

sub-linearly growing delay. On the simulation front, we provide numerical results verifying our theoretical findings, and also show that delay-adaptive step-sizes perform better than their non-delay-adaptive counterparts across a range of delay scenarios.

References

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

Assran, M.; Loizou, N.; Ballas, N.; and Rabbat, M. 2018. Stochastic gradient push for distributed deep learning. *arXiv preprint arXiv:1811.10792*.

Benaïm, M. 1999. Dynamics of stochastic approximation algorithms. In *Seminaire de probabilites XXXIII*. Springer. 1–68.

Bertsekas, D. P., and Tsitsiklis, J. N. 2003. Parallel and distributed computation: numerical methods.

Bottou, L. 1998. Online learning and stochastic approximations. *On-line learning in neural networks* 17(9):142.

Chaturapruek, S.; Duchi, J. C.; and Ré, C. 2015. Asynchronous stochastic convex optimization: the noise is in the noise and sgd don't care. In *Advances in Neural Information Processing Systems*, 1531–1539.

Cutkosky, A., and Busa-Fekete, R. 2018. Distributed stochastic optimization via adaptive sgd. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc. 1910–1919.

Damaskinos, G.; El Mhamdi, E. M.; Guerraoui, R.; Patra, R.; and Taziki, M. 2018. Asynchronous Byzantine machine learning (the case of SGD). In Dy, J., and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 1145–1154. Stockholmsmässan, Stockholm Sweden: PMLR.

Karakus, C.; Sun, Y.; Diggavi, S.; and Yin, W. 2017. Straggler mitigation in distributed optimization through data encoding. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 5434–5442.

Koloskova, A.; Stich, S. U.; and Jaggi, M. 2019. Decentralized stochastic optimization and gossip algorithms with compressed communication. *arXiv preprint arXiv:1902.00340*.

LeCun, Y. 1998. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Lelong, J. 2008. Almost sure convergence and of randomly truncated stochastic algorithms under verifiable conditions. *Statistics and Probability Letters* 78:2632–2636.

Mertikopoulos, P., and Zhou, Z. 2019. Learning in games with continuous action sets and unknown payoff functions. *Mathematical Programming* 173(1-2):465–507.

Peng, Z.; Xu, Y.; Yan, M.; and Yin, W. 2019. On the convergence of asynchronous parallel iteration with unbounded delays. *Journal of the Operations Research Society of China* 7(1):5–42.

Recht, B.; Re, C.; Wright, S.; and Niu, F. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, 693–701.

Sra, S.; Yu, A. W.; Li, M.; and Smola, A. J. 2015. Adadelay: Delay adaptive distributed stochastic convex optimization. *arXiv preprint arXiv:1508.05003*.

Wu, J.; Huang, W.; Huang, J.; and Zhang, T. 2018. Error compensated quantized SGD and its applications to large-scale distributed optimization. In Dy, J., and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 5325–5333. Stockholmsmässan, Stockholm Sweden: PMLR.

Xie, C.; Koyejo, S.; and Gupta, I. 2019. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In Chaudhuri, K., and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 6893–6901. Long Beach, California, USA: PMLR.

Yu, H., and Jin, R. 2019. On the computation and communication complexity of parallel sgd with dynamic batch sizes for stochastic non-convex optimization. *arXiv preprint arXiv:1905.04346*.

Zhou, Z.; Mertikopoulos, P.; Bambos, N.; Boyd, S.; and Glynn, P. W. 2017a. Stochastic mirror descent in variationally coherent optimization problems. In *Advances in Neural Information Processing Systems*, 7040–7049.

Zhou, Z.; Mertikopoulos, P.; Bambos, N.; Glynn, P. W.; and Tomlin, C. 2017b. Countering feedback delays in multi-agent learning. In *Advances in Neural Information Processing Systems*, 6171–6181.

Zhou, Z.; Mertikopoulos, P.; Moustakas, A. L.; Bambos, N.; and Glynn, P. 2017c. Mirror descent learning in continuous games. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 5776–5783. IEEE.

Zhou, Z.; Mertikopoulos, P.; Athey, S.; Bambos, N.; Glynn, P. W.; and Ye, Y. 2018a. Learning in games with lossy feedback. In *Advances in Neural Information Processing Systems*, 5134–5144.

Zhou, Z.; Mertikopoulos, P.; Bambos, N.; Glynn, P.; Ye, Y.; Li, L.-J.; and Fei-Fei, L. 2018b. Distributed asynchronous optimization with unbounded delays: How slow can you go? In *International Conference on Machine Learning*, 5965–5974.

Zhou, Z.; Mertikopoulos, P.; Bambos, N.; Boyd, S.; and Glynn, P. 2020a. On the convergence of mirror descent beyond stochastic convex programming. *SIAM Journal On Optimization*.

Zhou, Z.; Mertikopoulos, P.; Bambos, N.; Glynn, P.; and Ye, Y. 2020b. Distributed stochastic optimization with large delays. *Mathematics of Operations Research (under review)*.