Distributed Asynchronous Optimization with Unbounded Delays: How Slow Can You Go?

Zhengyuan Zhou, Panayotis Mertikopoulos, Nicholas Bambos, Peter W. Glynn, Yinyu Ye, Li-Jia Li, Fei-Fei Li

Distributed Asynchronous Optimization with Unbounded Delays: How Slow Can You Go?

Zhengyuan Zhou[1] **Panayotis Mertikopoulos**[2] **Nicholas Bambos**[1] **Peter Glynn**[1] **Yinyu Ye**[1] **Li-Jia Li**[3] **Li Fei-Fei**[1 3]

Abstract

One of the most widely used training methods for large-scale machine learning problems is distributed asynchronous stochastic gradient descent (DASGD). However, a key issue in its implementation is that of delays: when a "worker" node asynchronously contributes a gradient update to the "master", the global model parameter may have changed, rendering this information stale. In massively parallel computing grids, these delays can quickly add up if a node is saturated, so the convergence of DASGD is uncertain under these conditions. Nevertheless, by using a judiciously chosen quasilinear step-size sequence, we show that it is possible to amortize these delays and achieve global convergence with probability 1, even under polynomially growing delays, reaffirming in this way the successful application of DASGD to large-scale optimization problems.

1. Introduction

With the advent of high-performance computing infrastructures that are capable of handling massive amounts of data, distributed asychronous optimization has become the predominant paradigm in a broad range of large-scale machine learning and data science applications – ranging from statistical inference (Meshi & Schwing, 2017; Smyth et al., 2009; Grover & Ermon, 2016; Anand et al., 2016) and matrix completion (Recht et al., 2011; Yun et al., 2014; Petroni & Querzoni, 2014), to training deep neural networks (Dean et al., 2012a; Zhang et al., 2013; Paine et al., 2013; Zhang et al., 2015; Grover et al., 2015; Jiang et al., 2017) and representation learning (Grover & Leskovec, 2016; Grover et al., 2018a;b). As a result, recent years have witnessed a commensurate surge of interest in the asynchronous parallelization of first-order methods such as (stochastic) gradient descent (Agarwal & Duchi, 2011; Recht et al., 2011; Paine et al., 2013; Chaturapruek et al., 2015; Lian et al., 2015; Feyzmahdavian et al., 2016; Mania et al., 2017), coordinate descent (Liu et al., 2014; Avron et al., 2015; Liu & Wright, 2015; Tappenden et al., 2017; Fercoq & Richtárik, 2015), dual coordinate descent (Tran et al., 2015), block coordinate descent (Wang et al., 2016; Marecek et al., 2015; Wright, 2015), ADMM (Zhang & Kwok, 2014; Hong, 2017), etc.

This popularity is a direct consequence of Moore's law of silicon integration and the commensurately increased distribution of computing power. For instance, in a typical supercomputer cluster, up to several thousands of "workers" perform independent computations with little to no synchronization between them (as the cost of such coordination quickly becomes prohibitive in terms of overhead and energy spillage). Similarly, massively parallel computing grids and data centers may house up to several million computing nodes and/or servers, all working asynchronously to execute a variety of different tasks. Finally, taking the concept of distributed computing to its logical extreme, volunteer computing grids (such as Berkeley's BOINC infrastructure or Stanford's folding@home project) essentially span the entire globe and harness the computing power of a vast, heterogeneous network of non-clustered nodes that receive and process computational requests in a non-concurrent fashion, rendering syncrhonization impossible. In this way, by eliminating the required coordination overhead, asynchronous operations become simultaneously more appealing (in physically clustered systems) and more scalable (in massively parallel and volunteer computing grids).

In this broad landscape, distributed asynchronous stochastic gradient descent (DASGD) is one of the most widely deployed methods for training large-scale machine learning models, particularly when a stochastic gradient of the underlying learning objective is easily computable (Dean et al., 2012a;b; Krizhevsky et al., 2012; Zhang et al., 2013; Paine et al., 2013; Zhang et al., 2015). More concretely, there are two types of distributed computing architectures that are common in practice: The first is a shared-memory multi-core/multi-GPU cluster where different processors independently compute stochastic gradients and update a global

[1]Stanford University, Stanford, USA [2]Univ. Grenoble Alpes, CNRS, Inria, LIG, 38000 Grenoble, France. [3]Google, Mountain View, USA. Correspondence to: Zhengyuan Zhou <zyzhou@stanford.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

model parameter using a shared memory (see e.g., Recht et al., 2011; Chaturapruek et al., 2015; Feyzmahdavian et al., 2016). The second is a "master-slave" architecture where each worker independently computes a stochastic gradient of the objective; these gradients are sent asynchronously to the master as they become available, and the master sends out new computation requests after updating the model's global parameter (for instance, as in Agarwal & Duchi, 2011 and Lian et al., 2015).

In both cases, DASGD is inherently susceptible to *delays*, a key impediment that is absent in *centralized* stochastic optimization settings. For instance, in a master-slave system, when a worker sends its gradient update to the master, the master may have already updated the model parameters several times (using updates from other workers), so the received gradient is already stale by the time it is received. In fact, even in the perfectly synchronized setting where all workers have the same speed and send input to the master in an exact round-robin fashion, there is still a constant delay that grows roughly proportinally to the number of workers in the system (Agarwal & Duchi, 2011). This situation is exacerbated further in volunteer computing grids: here, workers typically volunteer their time and resources following a highly erratic and inconstant update/work schedule, often being turned off and/or being used for different tasks for hours (or even days) on end. In such cases, there is no lower bound on the fraction of resources used by a worker to compute an update at any given time (this is especially true in heterogeneous computing grids such as BOINC and SimGrid), meaning in turn that there is no upper bound on the induced delays (this can also happen in parallel computing environments where many tasks with different priorities are executed at the same time across different machines).

Our Contributions and Related Work. In this paper, we focus on the impact of outdated information on the convergence of DASGD. To begin with, in distributed *deterministic* optimization, it is well-known that convex problems can be solved by asynchronous descent with perfect gradient computations, even if the *delays grow sublinearly over time* (Bertsekas & Tsitsiklis, 1997). In the context of stochastic convex programming (where the objective is to minimize a function of the form $f(x) = \mathbb{E}[F(x,\omega)]$ for some random variable ω), recent works by Agarwal & Duchi (2011), Recht et al. (2011), Chaturapruek et al. (2015), Lian et al. (2015), Feyzmahdavian et al. (2016) and Mania et al. (2017) have derived convergence rates for DASGD under *bounded* delays, typically in the sense of mean ergodic averages. Extending these results to a non-convex setting, Lian et al. (2015) showed that the global state parameter X_n of DASGD enjoys the guarantee $n^{-1}\sum_{k=1}^{n}\mathbb{E}[\|\nabla f(X_k)\|_2^2] \to 0$ as $n \to \infty$, again under bounded delays.[1] Albeit powerful, this convergence result does not imply that DASGD converges to a stationary point, even in expectation; however, without further assumptions, it does not seem possible to refine this result further.

Our aim in this paper is to obtain sharper global convergence guarantees for DASGD with *unbounded delays* in a wide class of unimodal non-convex objectives known as *variationally coherent*. First introduced by Zhou et al. (2017a), this functional class properly includes, among others, all star- and quasi-convex stochastic programs (and hence all stochastic convex programs as well). Our main result may then be stated as follows: *in stochastic variationally coherent problems, the global state parameter X_n of DASGD converges to a global minimizer with probability* 1*, even when the delays between gradient updates and requests grow at a polynomial rate.*

This analysis extends the works mentioned above in several directions: it shows that *a*) convexity is not required to obtain almost sure global convergence; and *b*) bounded delays are not necessary to ensure the robustness of DASGD (in particular, even in the determinstic case, our analysis improves the sublinear requirement of Bertsekas & Tsitsiklis (1997) to polynomial). From this point of view, our results can be viewed as a contribution to the following question: *is vanilla DASGD robust to delays, and if so, to what extent?* We provide an affirmative answer to this question which, together with the existing rich literature on the topic, helps explain and reaffirm the prolific empirical success of DASGD in large-scale machine learning problems.

Our analysis relies on several novel ideas from the theory of stochastic approximation that were leveraged in a series of recent papers to examine descent schemes in a game-theoretic/continuous-time setting (Zhou et al., 2017b; Mertikopoulos & Staudigl, 2018a;b; Mertikopoulos & Zhou, 2018). Specifically, instead of focusing on the discrete-time algorithm directly, we first establish the convergence of an underlying, deterministic dynamical system, and then connect the continuous- and discrete-time systems via the stochastic approximation machinery of *asymptotic pseudotrajectories* (APTs), as pioneered by Benaïm & Hirsch (1996) and Benaïm (1999).

2. Problem Setup

Let $\mathcal{X}$ be a convex and compact subset of $\mathbb{R}^d$ and let $(\Omega, \mathcal{F}, \mathbb{P})$ be some underlying (complete) probability space. Throughout this paper, we will focus on the stochastic opti-

[1] Lian et al. (2015) and Mania et al. (2017) also examined both master-slave and multi-processor with shared memory, allowing for inconsistent reads and writes in the latter case.

mization problem:

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

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

$$f(x) = \mathbb{E}[F(x;\omega)] \tag{2.1}$$

for some random function $F\colon \mathcal{X} \times \Omega \to \mathbb{R}$ (not necessarily convex on $\mathcal{X}$). As is well-known in the distributed optimization literature (Agarwal & Duchi, 2011; Krizhevsky et al., 2012; Zhang et al., 2013; Paine et al., 2013; Lian et al., 2015), the stochastic expectation in Eq. (2.1) contains as a special case the common machine learning objectives of the form $N^{-1}\sum_{i=1}^{N} f_i(x)$, where each $f_i(x)$ is the loss associated with the i-th training sample.[2] Objectives of the form $N^{-1}\sum_{i=1}^{N} f_i(x)$ in fact encompasses a wide variety of machine learning tasks. For instance, even when f_i is assumed to be convex, least squares, logistic regression, SVM, matrix completion with trace norm (see (Bubeck et al., 2015) for more examples) can be seamlessly cast in this form.

In terms of regularity, we also make the following standard assumptions in the rest of our paper:

Assumption 1. F satisfies the following:

1. $F(x;\omega)$ is differentiable in x for $\mathbb{P}$-almost all $\omega \in \Omega$.
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: $\mathbb{E}[\nabla F(x;\omega)]$ is Lipschitz on $\mathcal{X}$.

Assumptions 1 and 2 together imply that f is differentiable because, by the dominated convergence theorem, $\nabla f(x) = \nabla\,\mathbb{E}[F(x;\omega)] = \mathbb{E}[\nabla F(x;\omega)]$.[3] Assumption 3 then implies that ∇f is Lipschitz continuous. Since f is continuous and $\mathcal{X}$ is compact, the solution set $\mathcal{X}^* \equiv \arg\min f$ of (Opt) is closed and nonempty.

2.1. Gradient Descent in Master-Slave Systems

Our main goal in this paper is to solve the optimization problem (Opt) in master-slave architectures, a widely used distributed computing framework for data-centers and parallel computing grids. The standard way of deploying stochastic gradient descent in such systems – and that which we adopt here – is for the workers to asychronously compute stochastic gradients and then send them to the master,[4] while the master updates the global state of the system and pushes the update back to the workers (Agarwal & Duchi, 2011; Lian et al., 2015). This process is presented in Algorithm 1:

Algorithm 1 Running SGD on a Master-Slave Architecture

Require: 1 Master and K workers, $k = 1, \ldots, K$

1: **repeat**
2: **Master**:
(a) Receive a stochastic gradient from worker k
(b) Update current iterate
(c) Send updated iterate to worker k
3: **Workers**:
(a) Receive iterate
(b) Compute an i.i.d. stochastic gradient
(c) Send gradient update to master
4: **until** end

As discussed in the introduction, since workers asychronously compute the gradients and then send updates to the master, a gradient received by the master on any given iteration can be stale (i.e., from a long-ago iteration). Since the master is the node that updates the global state of the system (the current solution candidate), we can get a clearer representation of this scheme by taking the master's point of view. The resulting process, aptly called *distributed asynchronous stochastic gradient descent* (DASGD) is encoded in pseudocode form in Algorithm 2 below.

Algorithm 2 Distributed asynchronous stochastic gradient descent

Require: Initial state $Y_0 \in \mathbb{R}^d$, step-size sequence α_n

$n \leftarrow 0$;
repeat
$X_n = \mathbf{proj}_{\mathcal{X}}(Y_n)$;
$Y_{n+1} = Y_n - \alpha_{n+1}\nabla F(X_{s(n)}, \omega_{s(n)+1})$;
$n \leftarrow n + 1$;
until end
return solution candidate X_n

In more detail, the master keeps track of a global counter n and increments it every time it updates the current solution candidate X_n. In what follows, we will write $s(n)$ for the iteration from which the gradient received at time n originated. In other words, the delay associated with iteration $s(n)$ is $n - s(n)$, since it took $n - s(n)$ iterations for the gradient computed on iteration $s(n)$ to be received by the master at stage n. Notation-wise, we will write d_n for the delay required to compute a gradient requested at iteration n. This gradient is received at stage $n + d_n$, whereas the delay for a gradient received at n is $d_{s(n)} = n - s(n)$.

[2] This setup corresponds to empirical risk minimization with $N^{-1}\sum_{i=1}^{N} f_i(x)$ representing an average with uniform weights.

[3] Note that finite second moments automatically imply finite first moments, which in turns guarantees that the expectation of the gradient exists.

[4] In machine learning applications, this is done by sampling a subset of the training data, computing the gradient for each data point and averaging over all points in the sample.

2.2. Mean Variational Coherence

Our goal in this paper is to establish the convergence of DASGD in as wide a class of problems as possible. Of course, global convergence[5] will not hold for all non-convex stochastic optimization problems (even without delays). As such, following Zhou et al. (2017a), we will focus on a class of non-convex functions called *coherent*:

Assumption 2. The optimization problem (Opt) is *variationally coherent in the mean* if

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

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

By Assumption 1, we can interchange expectation and differentiation in (VC) to obtain $\langle \nabla f(x), x - x^* \rangle > 0$, for all $x \notin \mathcal{X}^*$, $x^* \in \mathcal{X}^*$. As a result, mean variational coherence can be interpreted as an averaged coherence condition for the deterministic optimization problem with objective $f(x)$. A simple case of this, where $\mathcal{X} = \mathbb{R}^d$ and a unique minimizer x^* exists, is discussed in (Bottou, 1998), where many examples are given. Among others, this class of functions properly contains all star- and pseudo-convex functions; see Zhou et al. (2017a) for a detailed presentation of (VC).

Remark 2.1. It should be noted that (VC) is a significantly weaker requirement than the monotonicity condition $\langle \nabla f(x') - \nabla f(x), x' - x \rangle \geq 0$ for all $x, x' \in \mathcal{X}$, that characterizes convex functions. (VC) only concerns the minimum set of f and gives no information on generic point pairs (thus allowing for highly non-convex profiles).

3. Deterministic analysis

To streamline our presentation and build intuition along the way, we will begin with the deterministic case, where there is no randomness in the calculation of a gradient update. In this case,DASGD boils down to a *distributed asynchronous gradient descent* (DAGD), as illustrated in Algorithm 3:

Algorithm 3 Master's DAGD Update

Require: Initial state $y_0 \in \mathbb{R}^d$, step-size sequence α_n
1: $n \leftarrow 0$
2: **repeat**
3: $x_n = \mathbf{proj}_{\mathcal{X}}(y_n)$;
4: $y_{n+1} = y_n - \alpha_{n+1} \nabla f(x_{s(n)})$;
5: $n \leftarrow n + 1$;
6: **until** end
7: **return** solution candidate x_n

[5] Our focus in this paper is on global convergence. Other types of non-convex functions exist for which variants of SGD would converge to local optima (Zhang et al., 2017). Further, there can exist specific structured class of non-convex problems, that neither contain nor belong to VC, for which SGD converges to global optima (Chen et al., 2018).

3.1. Energy Function

A key role in the analysis of Algorithm 3 is played by the *energy function*

$$E(y) = \|p\|_2^2 - \|\mathbf{proj}_{\mathcal{X}}(y)\|_2^2 + 2\langle y, \mathbf{proj}_{\mathcal{X}}(y) - p \rangle, \tag{3.1}$$

where p is a fixed base point in $\mathcal{X}$ (typically a global minimizer $x^* \in \arg\min f$) and y is a gradient variable (typically a gradient iterate of Algorithm 2 or Algorithm 3, depending on the context).[6]

Lemma 3.1. *For all $p \in \mathcal{X}$, $y \in \mathbb{R}^d$, we have:*

1. *$E(y) \geq 0$ with equality if and only if $\mathbf{proj}_{\mathcal{X}}(y) = x$.*
2. *The sequence $\{y_n\}_{n=1}^{\infty}$ has $\lim_{n\to\infty} \mathbf{proj}_{\mathcal{X}}(y_n) = p$ if and only if $\lim_{n\to\infty} E(y_n) = p$.*

The proof of Lemma 3.1 is given in the appendix, but it is helpful to make a few quick remarks. The first statement justifies the terminology of "energy", as $E(y)$ is always non-negative. This energy function will also be the tool we use to establish an important component of the global convergence result. Further, it should also be clear that if $\mathbf{proj}_{\mathcal{X}}(y) = x$, then $E(y) = 0$; it is the "only if" part that is less obvious. For the second part of the lemma, it is again clear that if $\mathbf{proj}_{\mathcal{X}}(y_n) \to x$, then $E(y_n) \to 0$; the "only if" part is less obvious, but it is also what provides us with a way to establish convergence to optimal solutions. If we can show that $E(y_n) \to 0$, Lemma 3.1 would guarantee that $x_n = \mathbf{proj}_{\mathcal{X}}(y_n) \to x^*$.

3.2. Main Convergence Result

Going back to Algorithm 3, and with a fair bit of hindsight, we will make the following assumption relating the delays and the algorithm's step-size sequence:

Assumption 3. The gradient delay process d_n and the step-size sequence α_n of Algorithms 2 and 3 satisfy one of the following conditions:

1. *Bounded delays:* $\sup_n d_n < \infty$ and $\sum_{n=1}^{\infty} \alpha_n^2 < \infty$, $\sum_{n=1}^{\infty} \alpha_n = \infty$.
2. *Linearly growing delays:* $d_n = \mathcal{O}(n)$ and $\alpha_n \propto 1/(n \log n)$ for large n.
3. *Polynomially growing delays:* $d_n = \mathcal{O}(n^q)$ for some $q \geq 1$ and $\alpha_n \propto 1/(n \log n \log\log n)$ for large n.

The heavy lifting for our converence analysis is then provided by the following technical result:

Proposition 3.2. *Under Assumptions 1–3, DAGD admits a subsequence x_{n_k} that converges to $\mathcal{X}^*$ as $k \to \infty$.*

[6] In the above, we drop the dependence of E on p to stress the fact that the dynamic variable is y.

We highlight the main steps below and refer the reader to the appendix for the details:

1. Letting $b_n = \nabla f(x_{s(n)}) - \nabla f(x_n)$, we can rewrite the gradient update in DAGD as:

$$\begin{aligned} y_{n+1} &= y_n - \alpha_{n+1} \nabla f(x_{s(n)}) \\ &= y_n - \alpha_{n+1} \nabla f(x_n) \\ &\quad - \alpha_{n+1}\{\nabla f(x_{s(n)}) - \nabla f(x_n)\} \\ &= y_n - \alpha_{n+1}(\nabla f(x_n) + b_n). \end{aligned} \tag{3.2}$$

   Recall here that $s(n)$ denotes the previous iteration count whose gradient becomes available only at the current iteration n. By bounding the magnitude of b_n using the delay sequence through a careful analysis, we establish that under any of the three conditions in Assumption 3, $\lim_{n\to\infty} \|b_n\|_2 = 0$. The analysis here, particularly the one for the last two conditions, reveals the following pattern: as the magnitude of the delays gets larger and larger in the order of growth, one needs to use a more conservative step-size sequence in order to mitigate the damage done by the stale gradient information. Intuitively, smaller step-sizes are more helpful in larger delays because they carry a better "amortization" effect that makes DAGD more tolerant to delays.

2. With the defintion of b_n, DAGD can be written as:

$$\begin{aligned} x_n &= \mathbf{proj}_{\mathcal{X}}(y_n), \\ y_{n+1} &= y_n - \alpha_{n+1}(\nabla f(x_n) + b_n). \end{aligned} \tag{3.3}$$

   We then use the energy function to study the behavior of y_n and x_n. More specifically, we look at the quantity $E(y_{n+1}) - E(y_n)$ and bound this one-step change using the step size α_n, the b_n sequence and the defining quantity $\langle \nabla f(x_n), x_n - x^* \rangle$ of a variationally coherent function (as well as another term that will prove inconsequential). We then telescope on $E(y_{n+1}) - E(y_n)$ to obtain an upper bound for $E(y_{n+1}) - E(y_0)$. Since the energy function is always non-negative (by Lemma 3.1), $E(y_{n+1}) - E(y_0)$ is at least $-E(y_0)$ for every n. However, utilizing the fact that b_n converges to 0 and that $\langle \nabla f(x_n), x_n - x^* \rangle$ is always positive (unless the iterate is exactly an optimal solution), we show that the upper bound will approach $-\infty$ if X_n only enters $\mathcal{N}(\mathcal{X}^*, \epsilon)$, an open ϵ-neighborhood of $\mathcal{X}^*$, a finite number of times (for an arbitrary $\epsilon > 0$). This generates an immediate contradiction, and thereby establishes that X_n will get arbitrarily close to $\mathcal{X}^*$ for an infinite number of times. This then implies that there exists a subsequence of DAGD iterates that converges to the solution set of (Opt), i.e, $x_{n_k} \to \mathcal{X}^*$ as $k \to \infty$.

Theorem 3.3. *Under Assumptions 1–3, the global state variable x_n of DAGD (Algorithm 3) converges to the solution set $\mathcal{X}^*$ of (Opt).*

We give an outline of the proof below, referring to the appendix for the details. Also, for notational simplicity, we assume below that $\mathcal{X}^*$ is a singleton; the general case is no more difficult to prove.

Sketch of proof. Fix a $\delta > 0$. Since $x_{n_k} \to x^*$, as $k \to \infty$, it must be $E(y_{n_k}) \to 0$ as $k \to \infty$ per Lemma 3.1. So we can pick an n that is sufficiently large and $E(y_n) < \delta$. Now, there are two cases:

1. Case 1: $E(y_n) < \delta/2$.
2. Case 2: $\delta/2 \leq E(y_n) < \delta$.

For Case 1, we show in the appendix that

$$E(y_{n+1}) - E(y_n) \leq 2BC_4\alpha_{n+1} + 2\alpha_{n+1}^2(C_2 + B^2), \tag{3.4}$$

for suitable constants B and C. Now for n sufficiently large, we can make the right-hand arbitrarily small, and in particular, smaller than $\delta/2$. This means $E(y_{n+1}) < \delta$.

For Case 2, we show in the appendix that

$$E(y_{n+1}) - E(y_n) \leq -2\alpha_{n+1}\left[\frac{a}{2} - \alpha_{n+1}(C_2 + B^2)\right], \tag{3.5}$$

where a is a positive constant that depends only on δ. Again, since n is sufficiently large, we can make $\frac{a}{2} - \alpha_{n+1}(C_2 + B^2)$ positive, thereby making the right-hand side negative. Consequently, $E(y_{n+1}) < E(y_n) < \delta$.

The key conclusion from the above is that, for large enough n, once $E(y_n)$ is less than δ, $E(y_{n+1})$ is less than δ as well and so are all the iterates afterwards. Convergence of x_n to x^* is then immediate by the second part of Lemma 3.1. □

4. Stochastic analysis

The main ideas behind our deterministic analysis are relatively simple and intuitive. On the other hand, the stochastic case is much more involved because randomness can lead to very volatile behavior in the presence of delays. To streamline our presentation, we break the theoretical development into several sections, each comprising an important component of the overall analysis.

4.1. Recurrence of DASGD

We begin with the stochastic equivalent of Proposition 3.2:

Proposition 4.1. *Under Assumptions 1–3, DAGD admits a subsequence X_{n_k} that converges to $\mathcal{X}^*$ almost surely; concretely, $X_{n_k} \to \mathcal{X}^*$ with probability 1 as $k \to \infty$.*

Sketch of proof. We outline the two main steps of the proof below, referring the reader to the appendix for the details.

1. We begin by rewriting the gradient update step in DASGD as:

$$\begin{aligned}\frac{Y_{n+1}-Y_n}{\alpha_{n+1}} &= -\nabla F(X_{s(n)},\omega_{s(n)+1})\\ &= -\nabla f(X_n)\\ &\quad -[\nabla f(X_{s(n)})-\nabla f(X_n)]\\ &\quad -[\nabla F(X_{s(n)},\omega_{s(n)+1})-\nabla f(X_{s(n)})].\end{aligned} \tag{4.1}$$

   Letting $B_n = \nabla f(X_{s(n)}) - \nabla f(X_n)$ and $U_{n+1} = \nabla F(X_{s(n)},\omega_{s(n)+1}) - \nabla f(X_{s(n)})$, we may then rewrite the DASGD update as

$$Y_{n+1} = Y_n - \alpha_{n+1}\{\nabla f(X_n) + B_n + U_{n+1}\}. \tag{4.2}$$

   We then establish the following two facts in this step. First, we verify that $\sum_{r=0}^{n} U_{n+1}$ is a martingale adapted to $Y_1, Y_2 \ldots, Y_{n+1}$. Second, we show that $\lim_{n\to\infty} \|B_n\|_2 = 0, a.s.$.

   The second claim is done by first giving an upper bound on $\|B_n\|_2$ by writing $\nabla f(X_{s(n)}) - \nabla f(X_n)$ as a sum of one-step changes $(\nabla f(X_{s(n)}) - \nabla f(X_{s(n)+1}) + \nabla f(X_{s(n)+1}) - \cdots + \nabla f(X_{n-1}) - \nabla f(X_n))$ and analyzing each such successive change. We then break that upper bound into two parts, one deterministic and one stochastic. For the deterministic part, the same analysis in the proof of Proposition 3.2 yields convergence to 0.

   The stochastic part turns out to be the tail of a martingale. By leveraging the property of the step-size and a crucial property of martingale differences (two martingale differences at different time steps are uncorrelated), we establish that said martingale is L_2-bounded. Then, by applying a version of Doob's martingale convergence theorem, it follows that said martingale converges almost surely to a limit random variable with finite second moment (and hence almost surely finite). Consequently, writing the tail as a difference between two terms (each of which converges to the same limit variablewith probability 1), we conclude that the tail converges to 0 (a.s.).

2. The full DASGD update may then be written as

$$\begin{aligned}X_n &= \mathbf{proj}_{\mathcal{X}}(Y_n)\\ Y_{n+1} &= Y_n - \alpha_{n+1}[\nabla f(X_n) + B_n + U_{n+1}].\end{aligned} \tag{4.3}$$

   As in Step 2 of the proof of Proposition 3.2, we again bound the one-step change of the energy function $E(Y_{n+1}) - E(Y_n)$ and then telescope the differences. The two distinctions from the determinstic case are: 1) Everything is now a random variable. 2) We have three terms: in addition to B_n, we also have a martingale term U_{n+1}. Since B_n converges to 0 almost surely (as shown in the previous step), its effect is the same as b_n in the deterministic case. Futher, the analysis utilizes law of large numbers for martingale as well as Doob's martingale convergence theorem to bound the effect of the various martingale terms and to establish that the final dominating term is still the same term as in the deterministic case: a term that converges to $-\infty$ (which generates a contradiction since the energy function is always positive) unless a subsequence X_{n_k} converges almost surely to $\mathcal{X}^*$. □

4.2. Mean-Field Approximation of DASGD

With all this at hand, we can rewrite the DASGD update as:

$$\begin{aligned}X_n &= \mathbf{proj}_{\mathcal{X}}(Y_n)\\ Y_{n+1} &= Y_n - \alpha_{n+1}\{\nabla f(X_n) + B_n + U_{n+1}\}.\end{aligned} \tag{4.4}$$

Written in this way, DASGD can be viewed as a discretization of the "mean-field" ODE

$$\begin{aligned}x &= \mathbf{proj}_{\mathcal{X}}(y),\\ \dot{y} &= -\nabla f(x).\end{aligned} \tag{4.5}$$

The intuition is that this ODE provides a "mean" approximation of the DASGD update, because in (4.4), the noise term U_{n+1} has 0 mean, and the term B_n converges to 0 (and therefore has negilible effect in the long run). Thes leaves only the term $Y_{n+1} = Y_n - \alpha_{n+1}\nabla f(X_n)$, which can be seen as a Euler discretization of the ODE.

Next, writing Equation (4.5) solely in terms of y yields $\dot{y} = -\nabla f(\mathbf{proj}_{\mathcal{X}}(y))$. Since ∇f and $\mathbf{proj}_{\mathcal{X}}$ are both Lipschitz continuous and $\mathcal{X}$ is a compact set, the composition $\nabla f \circ \mathbf{proj}_{\mathcal{X}}$ is itself Lipschitz continuous and bounded. Standard results from the theory of dynamical systems then show that(4.5) admits a unique global solution $y(t)$ for any initial condition $y(0)$. On the other hand, since $\mathbf{proj}_{\mathcal{X}}$ is not a one-to-one map, it is not invertible; consequently, there need not exist a unique solution trajectory for $x(t)$. By this token, the rest of our analysis will focus on the trajetory of $y(t)$.

With the guarantee of the existence and uniqueness of the y trajectory, let $P\colon \mathbb{R}_+ \times \mathbb{R}^d \to \mathbb{R}^d$ be the flow of (4.5), i.e., $P(t, y_0)$ denotes the state of (4.5) at time t when the initial condition is y_0. In other words, when viewed as a function of time, $P(\cdot, y_0)$ is the solution trajectory to $\dot{y} = -\nabla f(\mathbf{proj}_{\mathcal{X}}(y))$. It is worth pointing out that writing it in this double-argument form also allows us to interpret P as a function of the initial condition: for a fixed t, $P(t, \cdot)$

gives different states at t when the ODE starts from different initial conditions (in particular, $P(0, y) = y$). Both views will be useful later.

We end this subsection with a "sufficient decrease" property of the mean dynamics (4.5):

Lemma 4.2. *With notation as above, we have:*

1. *If* $\mathbf{proj}_{\mathcal{X}}(P(t, y)) \notin \mathcal{X}^*$, *then* $E(P(t, y))$ *is strictly decreasing for all* $y \in \mathbb{R}^d$.
2. *For all* $\delta > 0$, *there exists some* $T \equiv T(\delta) > 0$ *such that, for all* $t \geq T$, *we have*

$$\sup_y\{E(P(t, y)) - E(y) : E(P(t, y)) > \delta/2\} < \delta/2.$$

4.3. Relating DASGD Iterates to ODE Trajectories

Lemma 4.2 essentially says $E(P(t, y))$ is strictly decreasing at a non-vanishing rate. By some additional analysis, one can then show[7] that Lemma Lemma 4.2 implies $P(t, y) \to \mathcal{X}^*, \forall y$ as $t \to \infty$. Now, if we can somehow show that the trajectory generated by the discrete-time iterates of DASGD is "close" to the continuous-time trajectory $P(t, y)$, then likely convergence of the DASGD iterates can be guaranteed as well.

To be more specific, there are two things that need to be more precisely defined from the preceding high-level discussion. First, what does it mean to be a trajectory generated by the discrete-time iterates of DASGD? Second, what does it mean to be "close"?

The answer to the first question is rather intuitive: (perhaps) the simplest way to generate a continuous trajectory from a sequence of discrete points is the *affine interpolation*: connect the iterates $Y_0, Y_1, \ldots, Y_n$ at times $0, \alpha_1, \ldots, \sum_{r=1}^{n=1} \alpha_r$. We call this curve the affine interpolation curve of DASGD and denote it by $A(t)$. Note that $A(t)$ is a random curve because the DASGD iterates $Y_0, Y_1, \ldots, Y_n$ are random. To avoid confusion, we summarize the three different objects discussed so far:

1. The DASGD iterates $Y_0, Y_1, \ldots, Y_n$.
2. The affine interpolation curve $A(t)$ of Y_n.
3. The flow $P(t, y)$ of the ODE (4.5).

The answer to the second question lies in the notion of an *asymptotic pseudotrajectory* (APT) , a concept introduced by Benaïm & Hirsch (1996) and Benaïm & Schreiber (2000). Specifically, the affine interpolation curve is considered close to ODE solution $P(t, y)$ if the following holds:

[7] Although this is an interesting conclusion, we do not prove it here because we are mainly concerned with establishing convergence of the DASGD iterates, rather than the ODE solution trajectory.

Definition 4.3. A continuous function $s : \mathbb{R}_+ \to \mathbb{R}^d$ is an APT for P if for every $T > 0$,

$$\lim_{t\to\infty} \sup_{0\leq h\leq T} d(s(t + h), P(h, s(t))) = 0. \tag{4.6}$$

Intuitively, the definition matches exactly the naming: s is an APT for P if, for sufficiently large t, the flow lines of P remain arbitrarily close to $X(t)$ over a time window of any (fixed)length. More precisely, for each fixed $T > 0$, one can find a large enough t_0,such that for all $t > t_0$, the curve $s(t + h)$ approximates the trajectory $P(h, s(t))$ on the interval $h \in [0, T]$ with any predetermined degree of accuracy.

Thanks to (Benaïm, 1999), we can easily verify that in our case $A(t)$ is in fact an APT for P:

Theorem 4.4 (Benaïm, 1999)**.** *The affine interpolation curve of the iterates generated by the difference equation* $Y_{n+1} = Y_n - \alpha_{n+1}\{G(X_n) + B_n + U_{n+1}\}$ *is an APT for the solution to the ODE* $\dot{y} = -G(y)$ *if the following three conditions* ***all*** *hold:*

1. G *is Lipschitz continuous and bounded.*
2. $\lim_{n\to\infty} B_n = 0$ *(a.s.).*
3. U_{n+1} *is a martingale difference sequence with* $\sup_n \mathbb{E}[\|U_{n+1}\|_2^p] < \infty$ *and* $\sum_{n=0}^{\infty} \alpha_{n+1}^{1+\frac{p}{2}} < \infty$ *for some* $p > 2$.

Remark 4.1. The above theorem is a combination of Proposition 4.1, Proposition 4.2 and Remark 4.5 in Benaïm (1999). We emphasize that $A(t)$ is an APT for P almost surely means that almost all realizations of the random paths $A(t)$ are asymptotic pseudotrajectories for P (recall that $A(t)$ is a random trajectory).

Corollary 4.5. $A(t)$ *is an APT of* P.

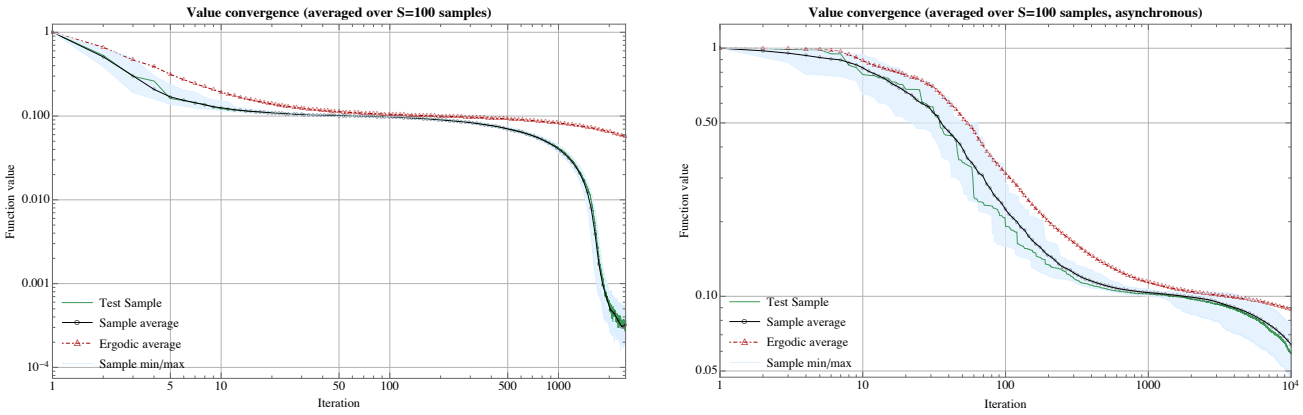
Proof. It suffices to verify the conditions of Theorem 4.4. The first condition is immediate: as argued before $\nabla f(\mathbf{proj}_{\mathcal{X}}(\cdot))$ is Lipschitz continuous. It is also bounded because $\mathbf{proj}_{\mathcal{X}}(\cdot)$'s domain is $\mathcal{X}$, which is compact. The second condition follows from part 1 in Proposition 4.1. The third condition is immediate by setting $p = 2$. □

4.4. Main Convergence Result

We are now in a position to combine all of the previous pieces to obtain our main convergence result:

Theorem 4.6. *Under Assumptions 1–3, the global state variable* X_n *of DASGD (Algorithm 2) converges (a.s.) to the solution set* $\mathcal{X}^*$ *of* (Opt).

Again, we only give an outline of the proof below, referring to the appendix for the more technical details.



(a) Convergence with no delays between gradient updates

(b) Convergence with linearly growing delays

Figure 1: Value convergence in a non-convex stochastic optimization problem with $d = 101$ degrees of freedom.

Proof. By Proposition 4.1, Y_n gets arbitrarily close to $\mathcal{X}^*$ infinitely often. Thus, it suffices to show that, if Y_n ever gets ϵ-close to $\mathcal{X}^*$, all the ensuing iterates are ϵ-close to $\mathcal{X}^*$ (a.s.).

The way we show this "trapping" property is to use the energy function. Specifically, we consider $E(A(t))$ and show that no matter how small ϵ is, for all sufficiently large t, if $E(A(t_0))$ is less than ϵ for some t_0, then $E(A(t)) < \epsilon, \forall t > t_0$. This would then complete the proof because $A(t)$ actually contains all the DASGD iterates, and hence if $E(A(t)) < \epsilon, \forall t > t_0$, then $E(Y_n) < \epsilon$ for all sufficiently large n. Furthermore, since $A(t)$ contains all the iterates, the hypothesis that " if $E(A(t_0))$ is less than ϵ for some t_0" will be satisfied due to Proposition 4.1.

We expand on one more layer of detail and defer the rest into appendix. Controlling $E(A(t))$, requires control of the energy on the ODE path $E(P(t, y))$ and the discrepancy between $E(P(t, y))$ and $E(A(t))$. The former can be made arbitrarily small as a result of Lemma 4.2; the latter can also be made arbitrarily small as a result of Corollary 4.5: since $A(t)$ is an APT for P, the two paths are close. Therefore, the discrepancy between $E(P)$ and $E(A)$ should also be vanishingly small. Consequently, since $E(A(t)) = E(P(t, y)) + \{E(A(t)) - E(P(t, y))\}$, and both terms on the right can be made arbitrarily small, so can $E(A(t))$ be made arbitrarily small. □

5. Numerical results

To validate our analysis, we test the convergence of Algorithm 2 against a standard Rosenbrock test function with $d = 101$ degrees of freedom, i.e.,

$$f_{\text{Ros}}(x) = \sum_{i=1}^{100} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2], \qquad (5.1)$$

with $x_i \in [0, 2]$, $i = 1, \ldots, 101$. The global minimum of f_{Ros} is located at $(1, \ldots, 1)$, at the end of a very thin and very flat parabolic valley which is notoriously difficult for first-order methods to traverse (Rosenbrock, 1960). Since the minimum of the Rosenbrock function is known, (VC) is easily checked over the problem's feasible region.

For our numerical experiments, we considered *a*) a synchronous update schedule as a baseline; and *b*) an asynchronous master-slave framework with random delays that scale as $d_n = \Theta(n)$. In both cases, Algorithm 2 was run with a decreasing step-size of the form $\alpha_n \propto 1/(n \log n)$ and stochastic gradients drawn from a standard multivariate Gaussian distribution (i.e., zero mean and identity covariance matrix).

Our results are shown in Fig. 1. Starting from a random (but otherwise fixed) initial condition, we ran $S = 100$ realizations of DASGD (with and without delays). We then plotted a randomly chosen trajectory ("test sample" in Fig. 1), the sample average, and the min/max over all samples at every update epoch. For comparison purposes, we also plotted the value of the so-called "ergodic average"

$$\bar{X}_n = \frac{\sum_{k=1}^n \alpha_k X_k}{\sum_{k=1}^n \alpha_k}, \qquad (5.2)$$

which is often used in the analysis of DASGD in the convex case (see e.g., Agarwal & Duchi, 2011). Even though this averaging leads to very robust convergence rate estimates in the convex case, we see here that it performs worse than the worst realization of DASGD. The reason for this is the lack of convexity: due to the ridges and talwegs of the Rosenbrock function, Jensen's inequality fails dramatically to produce an improvement over X_n (and, in fact, causes delays as it causes X_n to deviate from its gradient path).

References

Agarwal, Alekh and Duchi, John C. Distributed delayed stochastic optimization. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 24*, pp. 873–881. Curran Associates, Inc., 2011.

Anand, Ankit, Grover, Aditya, Mausam, Mausam, and Singla, Parag. Contextual symmetries in probabilistic graphical models. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pp. 3560–3568. AAAI Press, 2016. ISBN 978-1-57735-770-4.

Avron, Haim, Druinsky, Alex, and Gupta, Anshul. Revisiting asynchronous linear solvers: Provable convergence rate through randomization. *Journal of the ACM (JACM)*, 62(6):51, 2015.

Benaïm, Michel. Dynamics of stochastic approximation algorithms. In Azéma, Jacques, Émery, Michel, Ledoux, Michel, and Yor, Marc (eds.), *Séminaire de Probabilités XXXIII*, volume 1709 of *Lecture Notes in Mathematics*, pp. 1–68. Springer Berlin Heidelberg, 1999.

Benaïm, Michel and Hirsch, Morris W. Asymptotic pseudotrajectories and chain recurrent flows, with applications. *Journal of Dynamics and Differential Equations*, 8(1):141–176, 1996.

Benaïm, Michel and Schreiber, Sebastian J. Ergodic properties of weak asymptotic pseudotrajectories for semiflows. *Journal of Dynamics and Differential Equations*, 12(3):579–598, 2000.

Bertsekas, Dimitri P. and Tsitsiklis, John N. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997. ISBN 1886529019.

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

Bubeck, Sébastien et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8 (3-4):231–357, 2015.

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

Chen, Yuxin, Chi, Yuejie, Fan, Jianqing, and Ma, Cong. Gradient descent with random initialization: Fast global convergence for nonconvex phase retrieval. *arXiv preprint arXiv:1803.07726*, 2018.

Dean, Jeffrey, Corrado, Greg, Monga, Rajat, Chen, Kai, Devin, Matthieu, Mao, Mark, Senior, Andrew, Tucker, Paul, Yang, Ke, Le, Quoc V, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pp. 1223–1231, 2012a.

Dean, Jeffrey, Corrado, Greg S., Monga, Rajat, Chen, Kai, Devin, Matthieu, Le, Quoc V., Mao, Mark Z., Ranzato, Marc'Aurelio, Senior, Andrew, Tucker, Paul, Yang, Ke, and Ng, Andrew Y. Large scale distributed deep networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 1223–1231, USA, 2012b. Curran Associates Inc.

Fercoq, Olivier and Richtárik, Peter. Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015.

Feyzmahdavian, Hamid Reza, Aytekin, Arda, and Johansson, Mikael. An asynchronous mini-batch algorithm for regularized stochastic optimization. *IEEE Transactions on Automatic Control*, 61(12):3740–3754, 2016.

Grover, Aditya and Ermon, Stefano. Variational bayes on monte carlo steroids. In *Advances in Neural Information Processing Systems*, pp. 3018–3026, 2016.

Grover, Aditya and Leskovec, Jure. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864. ACM, 2016.

Grover, Aditya, Kapoor, Ashish, and Horvitz, Eric. A deep hybrid model for weather forecasting. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 379–386. ACM, 2015.

Grover, Aditya, Al-Shedivat, Maruan, Gupta, Jayesh K., Burda, Yura, and Edwards, Harrison. Learning policy representations in multiagent systems. In *International Conference on Machine Learning*, 2018a.

Grover, Aditya, Zweig, Aaron, and Ermon, Stefano. Graphite: Iterative generative modeling of graphs. *arXiv preprint arXiv:1803.10459*, 2018b.

Hong, Mingyi. A distributed, asynchronous and incremental algorithm for nonconvex optimization: An admm approach. *IEEE Transactions on Control of Network Systems*, 2017.

Jiang, Lu, Zhou, Zhengyuan, Leung, Thomas, Li, Li-Jia, and Fei-Fei, Li. Mentornet: Regularizing very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055*, 2017.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012.

Lian, Xiangru, Huang, Yijun, Li, Yuncheng, and Liu, Ji. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pp. 2737–2745, 2015.

Liu, Ji and Wright, Stephen J. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *SIAM Journal on Optimization*, 25(1):351–376, 2015.

Liu, Ji, Wright, Steve, Re, Christopher, Bittorf, Victor, and Sridhar, Srikrishna. An asynchronous parallel stochastic coordinate descent algorithm. In *International Conference on Machine Learning*, pp. 469–477, 2014.

Mania, Horia, Pan, Xinghao, Papailiopoulos, Dimitris, Recht, Benjamin, Ramchandran, Kannan, and Jordan, Michael I. Perturbed iterate analysis for asynchronous stochastic optimization. *SIAM Journal on Optimization*, 27(4):2202–2229, 2017.

Marecek, Jakub, Richtarik, Peter, and Takac, Martin. Distributed block coordinate descent for minimizing partially separable functions. In *Numerical Analysis and Optimization*, pp. 261–288. Springer, 2015.

Mertikopoulos, Panayotis and Staudigl, Mathias. On the convergence of gradient-like flows with noisy gradient input. *SIAM Journal on Optimization*, 28(1):163–197, January 2018a.

Mertikopoulos, Panayotis and Staudigl, Mathias. Stochastic mirror descent dynamics and their convergence in monotone variational inequalities. *Journal of Optimization Theory and Applications*, 2018b.

Mertikopoulos, Panayotis and Zhou, Zhengyuan. Learning in games with continuous action sets and unknown payoff functions. *Mathematical Programming*, 2018.

Meshi, Ofer and Schwing, Alexander. Asynchronous parallel coordinate minimization for map inference. 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*, pp. 5738–5748. Curran Associates, Inc., 2017.

Paine, Thomas, Jin, Hailin, Yang, Jianchao, Lin, Zhe, and Huang, Thomas. Gpu asynchronous stochastic gradient descent to speed up neural network training. *arXiv preprint arXiv:1312.6186*, 2013.

Petroni, Fabio and Querzoni, Leonardo. Gasgd: stochastic gradient descent for distributed asynchronous matrix completion via graph partitioning. In *Proceedings of the 8th ACM Conference on Recommender systems*, pp. 241–248. ACM, 2014.

Recht, Benjamin, Re, Christopher, Wright, Stephen, and Niu, Feng. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pp. 693–701, 2011.

Rosenbrock, Howard Harry. An automatic method for finding the greatest or least value of a function. *Computer Journal*, 3(3): 175–184, 1960.

Smyth, Padhraic, Welling, Max, and Asuncion, Arthur U. Asynchronous distributed learning of topic models. In *Advances in Neural Information Processing Systems*, pp. 81–88, 2009.

Tappenden, Rachael, Takac, Martin, and Richtarik, Peter. On the complexity of parallel coordinate descent. *Optimization Methods and Software*, pp. 1–24, 2017.

Tran, Kenneth, Hosseini, Saghar, Xiao, Lin, Finley, Thomas, and Bilenko, Mikhail. Scaling up stochastic dual coordinate ascent. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pp. 1185–1194, New York, NY, USA, 2015. ACM.

Wang, Yu-Xiang, Sadhanala, Veeranjaneyulu, Dai, Wei, Neiswanger, Willie, Sra, Suvrit, and Xing, Eric. Parallel and distributed block-coordinate frank-wolfe algorithms. In *International Conference on Machine Learning*, pp. 1548–1557, 2016.

Wright, Stephen J. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

Yun, Hyokun, Yu, Hsiang-Fu, Hsieh, Cho-Jui, Vishwanathan, SVN, and Dhillon, Inderjit. Nomad: Non-locking, stochastic multi-machine algorithm for asynchronous and decentralized matrix completion. *Proceedings of the VLDB Endowment*, 7 (11), 2014.

Zhang, Ruiliang and Kwok, James. Asynchronous distributed admm for consensus optimization. In *International Conference on Machine Learning*, pp. 1701–1709, 2014.

Zhang, Shanshan, Zhang, Ce, You, Zhao, Zheng, Rong, and Xu, Bo. Asynchronous stochastic gradient descent for dnn training. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

Zhang, Sixin, Choromanska, Anna E, and LeCun, Yann. Deep learning with elastic averaging sgd. In *Advances in Neural Information Processing Systems*, pp. 685–693, 2015.

Zhang, Yuchen, Liang, Percy, and Charikar, Moses. A hitting time analysis of stochastic gradient langevin dynamics. *arXiv preprint arXiv:1702.05575*, 2017.

Zhou, Zhengyuan, Mertikopoulos, Panayotis, Bambos, Nicholas, Boyd, Stephen, and Glynn, Peter W. Stochastic mirror descent for variationally coherent optimization problems. In *NIPS '17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017a.

Zhou, Zhengyuan, Mertikopoulos, Panayotis, Bambos, Nicholas, Glynn, Peter W., and Tomlin, Claire. Countering feedback delays in multi-agent learning. In *NIPS '17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017b.