

A Comprehensive Linear Speedup Analysis for Asynchronous Stochastic Parallel Optimization from Zeroth-Order to First-Order

Xiangru Lian[†], Huan Zhang[‡], Cho-Jui Hsieh[‡], Yijun Huang[†], and Ji Liu^{†‡b}

[†]Department of Computer Science, University of Rochester, USA

[‡]Department of Computer Science, University of California, Davis, USA

[‡]Department of Electrical and Computer Engineering, University of Rochester, USA

^bGoergen Institute for Data Science, University of Rochester, USA

xiangru@yandex.com, victzhang@gmail.com, chohsieh@ucdavis.edu,
huangyj0@gmail.com, ji.liu.uwisc@gmail.com

May 20, 2017

Abstract

Asynchronous parallel optimization received substantial successes and extensive attention recently. One of core theoretical questions is how much speedup (or benefit) the asynchronous parallelization can bring us. This paper provides a comprehensive and generic analysis to study the speedup property for a broad range of asynchronous parallel stochastic algorithms from the zeroth order to the first order methods. Our result recovers or improves existing analysis on special cases, provides more insights for understanding the asynchronous parallel behaviors, and suggests a novel asynchronous parallel zeroth order method for the first time. Our experiments provide novel applications including model blending problems using the proposed asynchronous parallel zeroth order method.

1 Introduction

Asynchronous parallel optimization received substantial successes and extensive attention recently, for example, [5, 6, 14, 19, 22, 23, 24, 26, 33, 38, 44, 46, 47, 48, 49, 50, 51, 53, 54]. It has been used to solve various machine learning problems, such as deep learning [4, 9, 39, 52], matrix completion [38, 41, 49], SVM [20], linear systems [3, 30], PCA [13], and linear programming [45]. Its main advantage over the synchronous parallel optimization is avoiding the synchronization cost, so it minimizes the system overheads and maximizes the efficiency of all computation workers.

A core theoretical question is how much speedup (or benefit) the asynchronous parallelization can bring us, that is, how much time can we save by employing more computation resources? More precisely, the **running time speedup** (RTS) with T workers:

$$\text{RTS}(T) = \frac{\text{running time using a single worker}}{\text{running time using } T \text{ workers}}.$$

Since in the asynchronous parallelism all workers keep busy, RTS can be measured roughly by the **computational complexity speedup** (CCS) with T workers¹

$$\text{CCS}(T) = \frac{\text{total computational complexity using a single worker}}{\text{total computational complexity using } T \text{ workers}} \times T.$$

In this paper, we are mainly interested in the conditions to ensure the linear speedup property. More specifically, what is the upper bound on T to ensure $\text{CCS}(T) = \Theta(T)$?

Existing studies on special cases, such as asynchronous stochastic gradient descent (ASGD) and asynchronous stochastic coordinate descent (ASCD), have revealed some clues for what factors can affect the upper bound of T . For example, Agarwal and Duchi [1] showed the upper bound depends on the variance of the stochastic gradient in ASGD; Niu et al. [38] showed that the upper bound depends on the data sparsity and the dimension of the problem in ASGD; and Avron et al. [3], Liu and Wright [28] found that the upper bound depends on the problem dimension as well as the diagonal dominance of the Hessian matrix of the objective. However, we still do not have a comprehensive analysis to comprehend all pieces and show how these factors jointly affect the speedup property.

This paper provides a comprehensive and generic analysis to study the speedup property for a broad range of asynchronous parallel stochastic algorithms from the zeroth order to the first order methods.

To avoid unnecessary complication and cover practical problems and algorithms, we consider the following nonconvex stochastic optimization problem:

$$\min_{x \in \mathbb{R}^N} f(x) := \mathbb{E}_{\xi} (F(x; \xi)), \tag{1}$$

where $\xi \in \Xi$ is a random variable, and both $F(\cdot; \xi) : \mathbb{R}^N \rightarrow \mathbb{R}$ and $f(\cdot) : \mathbb{R}^N \rightarrow \mathbb{R}$ are smooth but not necessarily convex functions. This objective function covers a large scope of machine learning problems including deep learning – an extremely hot topic in machine learning. $F(\cdot; \xi)$'s are called *component functions* in this paper. The most common specification is that Ξ is an index set of all training samples $\Xi = \{1, 2, \dots, n\}$ and $F(x; \xi)$ is the loss function with respect to the training sample indexed by ξ .

We highlight the main contributions of this paper in the following:

- We provide a generic analysis for convergence and speedup, which covers many existing algorithms including ASCD, ASGD (implementation on parameter server), ASGD (implementation on multicore systems), and others as its special cases.

¹For simplicity, we assume that the communication cost is not dominant throughout this paper.

Table 1: Asynchronous parallel algorithms. “I” and “C” in “model” stand for inconsistent and consistent read model respectively, which will be explained later. “Base alg.” is short for base algorithm.

Asyn. alg.	base alg.	problem type	upper bound of T	model
ASGD [38]	SGD	smooth, strongly convex	$O(N^{1/4})$	C
ASGD [1]	SGD	smooth, convex	$O(K^{1/4} \min\{\sigma^{3/2}, \sigma^{1/2}\})$	C
ASGD[15]	SGD	composite, convex	$O(K^{1/4}\sigma^{1/2})$	C
ASGD [27]	SGD	smooth, nonconvex	$O(N^{1/4}K^{1/2}\sigma)$	I
ASGD [27]	SGD	smooth, nonconvex	$O(K^{1/2}\sigma)$	C
ARK [30]	SGD	$Ax = b$	$O(N)$	C
ASCD [29]	SCD	smooth, convex, unconstrained	$O(N^{1/2})$	C
ASCD [29]	SCD	smooth, convex, constrained	$O(N^{1/4})$	C
ASCD [28]	SCD	composite, convex	$O(N^{1/4})$	I
ASCD [3]	SCD	$\frac{1}{2}x^T Ax - b^T x$	$O(N)$	C
ASCD [3]	SCD	$\frac{1}{2}x^T Ax - b^T x$	$O(N^{1/2})$	I
ASCD [20]	SCD	$\frac{1}{2}x^T Ax - b^T x$, constrained	$O(N^{1/2})$	I
ASZD	zeroth order SGD & SCD	smooth, nonconvex	$O(\sqrt{N^{3/2} + KN^{1/2}\sigma^2})$	I
ASGD	SGD	smooth, nonconvex	$O(\sqrt{N^{3/2} + KN^{1/2}\sigma^2})$	I
ASGD	SGD	smooth, nonconvex	$O(\sqrt{K\sigma^2 + 1})$	C
ASCD	SCD	smooth, nonconvex	$O(N^{3/4})$	I

- Our generic analysis can recover or improve the existing results on special cases.
- Our generic analysis suggests a novel asynchronous stochastic zeroth-order gradient descent (ASZD) algorithm and provides the analysis for its convergence rate and speedup property. To the best of our knowledge, this is the first asynchronous parallel *zeroth* order algorithm.
- The experiment includes a novel application of the proposed ASZD method on model blending. Note that the algorithm can also be used in many other important applications such as hyper-parameter tuning.

1.1 Related Work

In this section we review the stochastic algorithms for optimization problems. According to the type of the information that the oracle provides in the algorithm, we can the stochastic algorithms into two categories: *zeroth-order* methods and *first-order* methods. Both can be extended to asynchronous parallel versions. We use N to denote the dimension of the problem, K to denote the iteration number, and σ to the variance of stochastic gradients.

The first order methods acquire gradient information of the objective at a given point. These methods include the popular *stochastic gradient “descent” method* (SGD) and *stochastic coordinate descent method* (SCD).

The *stochastic gradient method* is a very powerful and popular approach for solving large scale optimization problems. The well known convergence rate for stochastic gradient is $O(1/\sqrt{K})$ for general convex problems and $O(1/K)$ for strongly convex problems, for example, see [34, 35]. Most studies for stochastic gradient focus on convex optimization. For nonconvex optimization, Ghadimi and Lan [16] proved an ergodic convergence rate of $O(1/\sqrt{K})$, which is consistent with the convergence rate for convex problems.

Stochastic coordinate descent is another powerful first order algorithm for solving large scale optimization problems. For SCD Nesterov [36] proved a linear convergence rate for strongly convex objectives and a sublinear $O(1/K)$ rate for convex objectives. Richtárik and Takáč [43], Lu and Xiao [31] extended the stochastic coordinate descent algorithm to optimize composite function objectives.

Zeroth order methods only acquire the objective (or component) function value at any point. One intuitive way to use these information is to approximate the gradient of a function along a direction vector \hat{n} at some point x in the first order algorithms by calculating the difference of the function on two points $x + \epsilon\hat{n}$ and $x - \epsilon\hat{n}$ near enough (i.e. ϵ is small enough) and divide the difference by the distance between the two points (i.e. $2\epsilon\|\hat{n}\|$). There are theories to justify whether the approximation is “good enough”. Both stochastic gradient method and stochastic coordinate descent have their corresponding zeroth-order counterparts.

Nesterov and Spokoiny [37] proved a convergence rate of $O(N/\sqrt{K})$ for a zeroth-order stochastic gradient method applied to nonsmooth convex problems. Based on [37], Ghadimi and Lan [16] proved a convergence rate of $O(\sqrt{N/K})$ for a zeroth-order stochastic gradient method on nonconvex smooth problems. Jamieson et al. [21] proved that for any zeroth-order method with inaccurate evaluation of the objective, a lower bound of the optimization error is $O(1/\sqrt{K})$. Duchi et al. [11] proved a $O(\frac{N^{1/4}}{K} + \frac{1}{\sqrt{K}})$ rate for a zeroth order stochastic gradient method on convex objectives with some very different assumptions compared with our paper. They also proved the convergence on strongly convex objectives. Agarwal et al. [2] proved a $O(\text{poly}(N)\sqrt{K})$ regret with zeroth-order bandit algorithm on convex objectives. Please refer to Conn et al. [8] for more related work about zeroth-order methods.

Finally we review *first-order asynchronous parallel stochastic algorithms*. Avron et al. [3], Liu et al. [30] employed asynchronous algorithms to solve linear systems. Agarwal and Duchi [1] proved the asynchronous stochastic gradient method for *convex smooth* problems has a convergence rate of $O(\frac{1}{K} + \frac{\sigma}{\sqrt{K}} + \frac{T^2+T/\sigma^2}{K})$, which implies that linear speedup is achieved when $T \leq O(K^{1/4})$. Feyzmahdavian et al. [15] extended the analysis in [1] to minimize functions in the form “smooth convex loss + nonsmooth convex regularization” and obtained similar results. Lian et al. [27] proved a similar $O(1/\sqrt{K})$ convergence rate for asynchronous SGD with $O(\sqrt{K})$ linear speedup for nonconvex problems. Niu et al. [38] proposed a lock free asynchronous parallel implementation of SGD on the shared memory system and described this implementation as HOGWILD! algorithm. They proved a sublinear convergence rate $O(1/K)$ for *strongly* convex smooth objectives. Duchi et al. [12] proved a $O(1/K)$ convergence rate for asynchronous stochastic gradient method for convex objectives under the assumption that the second moment of the stochastic gradients is

upper bounded. Mania et al. [32] provided analysis for several asynchronous parallel optimization algorithms, including HOGWILD!, asynchronous stochastic coordinate descent and asynchronous sparse SVRG. Reddi et al. [42] proved the convergence of the asynchronous variance reduced stochastic gradient method and its speedup in sparse setting. Gurbuzbalaban et al. [18] proved a linear convergence rate for the incremental aggregated gradient algorithm, which uses stale gradient information. Table 1 summarizes known asynchronous parallel algorithms and their corresponding speedup properties.

To the best of our knowledge, this paper provides the first analysis for *zeroth-order asynchronous parallel stochastic algorithms*.

1.2 Notation

- $e_i \in \mathbb{R}^N$ denotes the i^{th} natural unit basis vector.
- $\mathbb{E}(\cdot)$ means taking the expectation with respect to all random variables, while $\mathbb{E}_a(\cdot)$ denotes the expectation with respect to a random variable a .
- $\nabla f(x) \in \mathbb{R}^N$ is the gradient of $f(x)$ with respect to x . Let S be a subset of $\{1, \dots, N\}$. $\nabla_S f(x) \in \mathbb{R}^N$ is the projection of $\nabla f(x)$ onto the index set S , that is, setting components of $\nabla f(x)$ outside of S to be zero. We use $\nabla_i f(x) \in \mathbb{R}^N$ to denote $\nabla_{\{i\}} f(x)$ for short.
- f^* denotes the optimal objective value in (1).

2 Algorithm

Algorithm 1 Generic Asynchronous Stochastic Algorithm (GASA)

Require: $x_0, K, Y, (\mu_1, \mu_2, \dots, \mu_N), \{\gamma_k\}_{k=0, \dots, K-1}$ $\triangleright \gamma_k$ is the step length for k^{th} iteration
Ensure: $\{x_k\}_{k=0}^K$
1: **for** $k = 0, \dots, K - 1$ **do**
2: Randomly select a component function index ζ_k and a set of coordinate indices S_k , where $|S_k| = Y$;
3: $x_{k+1} = x_k - \gamma_k G_{S_k}(\hat{x}_k; \zeta_k)$;
4: **end for**

We illustrate the asynchronous parallelism by assuming a centralized network: a central node and multiple child nodes (workers). The central node maintains the optimization variable x . It could be a parameter server if implemented on a computer cluster [25]; it could be a shared memory if implemented on a multicore machine. Given a base algorithm \mathcal{A} , all child nodes run algorithm \mathcal{A} independently and concurrently: read the value of x from the central node (we call the result of

this read \hat{x}), make local calculation using the \hat{x} , and modify the x on the central node. There is no need to synchronize child nodes or enforce them to communicate with each other for coordination. Therefore, all child nodes stay busy, and consequently their efficiency gets maximized. In other words, we have $\text{CCS}(T) \approx \text{RTS}(T)$. Note that due to the asynchronous parallel mechanism, the variable x in the central node is not updated exactly following the protocol of Algorithm \mathcal{A} , since when a child node returns its computation result, the x in the central node might have been changed by other child nodes. Thus, a new analysis is required. A fundamental question would be under what conditions a linear speedup can be guaranteed. In other words, under what conditions $\text{CCS}(T) = \Theta(T)$ or equivalently $\text{RTS}(T) = \Theta(T)$.

To provide a comprehensive analysis, we consider the following algorithm \mathcal{A} – a zeroth order hybrid of SCD and SGD: iteratively sample a component function² indexed by ξ and a coordinate block $S \subseteq \{1, 2, \dots, N\}$, where $|S| = Y$ for some constant Y and update x with

$$x \leftarrow x - \gamma G_S(x; \xi) \quad (2)$$

where $G_S(x; \xi)$ is an approximation to the block coordinate stochastic gradient $NY^{-1}\nabla_S f(x; \xi)$:

$$G_S(x; \xi) := \sum_{i \in S} \frac{N}{2Y\mu_i} (F(x + \mu_i e_i; \xi) - F(x - \mu_i e_i; \xi)) e_i, \quad S \subset \{1, 2, \dots, N\}. \quad (3)$$

In the definition of $G_S(x; \xi)$, μ_i is the finite difference step size for the i^{th} coordinate. $(\mu_1, \mu_2, \dots, \mu_N)$ is predefined in practice. We only use the function value (the zeroth order information) to estimate $G_S(x; \xi)$. It is easy to see that the closer to 0 the μ_i 's are, the closer $G_S(x; \xi)$ and $NY^{-1}\nabla_S f(x; \xi)$ will be. In particular, $\lim_{\mu_i \rightarrow 0, \forall i} G_S(x; \xi) = NY^{-1}\nabla_S f(x; \xi)$.

Applying the asynchronous parallelism, we propose a generic asynchronous stochastic algorithm in Algorithm 1. This algorithm essentially characterizes how the value of x is updated in the central node. γ_k is the predefined steplength (or learning rate). K is the total number of iterations (note that this iteration number is counted by the the central node, that is, any update on x no matter from which child node will increase this counter.)

One may notice that the key difference from the protocol of Algorithm \mathcal{A} in Eq. (2) is that \hat{x}_k (which is the value of x in the server read by a child node, and is mathematically defined later in (4)) is not equal to x_k in general, and this difference is caused by the asynchronous parallelism. In consequence, existing analysis for zeroth order optimization cannot be directly applied here, and a new analysis is required. In asynchronous parallelism, there are two different ways to model the value of \hat{x}_k :

- **Consistent read:** \hat{x}_k is some early *existed* state of x in the central node, that is, $\hat{x}_k = x_{k-\tau_k}$ for some $\tau_k \geq 0$. This happens if reading x and writing x on the central node by any child node are atomic operations, for instance, the implementation on a parameter server [25].

²The algorithm and theoretical analysis followed can be easily extended to the minibatch version.

- **Inconsistent read:** \hat{x}_k could be more complicated when the atomic read on x cannot be guaranteed. For example, for the implementation on the multi-core system, the central node is the shared memory which can be accessed by all child nodes at any time without locking.³ It means that while one child is reading x in the central node, other child nodes may be performing modifications on x at the same time. Therefore, different coordinates of x read by any child node may have different ages. In other words, \hat{x}_k may not be any *existed* state of x in the central node.

Readers who want to learn more details and examples about consistent read and inconsistent read can refer to [3, 27, 28]. To generalize the analysis for the consistent read mode and the inconsistent read mode, we note that \hat{x}_k can be represented in the following generic form:

$$\hat{x}_k = x_k - \sum_{j \in J(k)} (x_{j+1} - x_j), \quad (4)$$

where $J(k) \subset \{k-1, k-2, \dots, k-T\}$ is a subset of the indices of early iterations, and T is the upper bound for staleness. This expression is also considered in [3, 27, 28, 40]. *Note that the practical value of T is usually proportional to the number of involved nodes (or workers).* Therefore, the total number of workers and the upper bound of the staleness are treated as the same in the following discussion and this notation T is abused for simplicity.

3 Theoretical Analysis

Before we show the main results of this paper, let us first make some global assumptions commonly used for the analysis of stochastic algorithms.⁴

Bounded Variance The variance of stochastic gradients is bounded:

$$\mathbb{E}_{\xi}(\|\nabla F(x; \xi) - \nabla f(x)\|^2) \leq \sigma^2, \quad \forall x. \quad (5)$$

Lipschitzian Gradient The gradient of both the objective and its component functions are Lipschitzian:⁵

$$\max\{\|\nabla f(x) - \nabla f(y)\|, \|\nabla F(x; \xi) - \nabla F(y; \xi)\|\} \leq L\|x - y\| \quad \forall x, \forall y, \forall \xi. \quad (6)$$

Under the Lipschitzian gradient assumption, define two more constants L_s and L_{\max} . Let s be any positive integer bounded by N . Define L_s to be the minimal constant satisfying the

³Of course, one can always use “lock” to restrict the access from child nodes, but it largely degrades the performance [38]. Therefore, it is rarely used to implement asynchronous parallel algorithms in practice.

⁴Some underlying assumptions such as reading and writing a float number are omitted here. As pointed in [38], these behaviors are guaranteed by most modern architectures.

⁵Note that the Lipschitz assumption on the component function $F(x; \xi)$'s can be eliminated when it comes to first order methods (i.e., $\omega \rightarrow 0$) in our following theorems.

following inequality: $\forall \xi, \forall x, \forall S \subset \{1, 2, \dots, N\}$ with $|S| \leq s$:

$$\max \left\{ \left\| \nabla f(x) - \nabla f \left(x + \sum_{i \in S} \alpha_i e_i \right) \right\|, \left\| \nabla F(x; \xi) - \nabla F \left(x + \sum_{i \in S} \alpha_i e_i; \xi \right) \right\| \right\} \leq L_s \left\| \sum_{i \in S} \alpha_i e_i \right\|.$$

Note that L_Y and L_T in the following text are L_s defined here with $s = Y$ and $s = T$ respectively.

Define $L_{(i)}$ for $i \in \{1, 2, \dots, N\}$ as the minimum constant that satisfies:

$$\max \{ \|\nabla_i f(x) - \nabla_i f(x + \alpha e_i)\|, \|\nabla_i F(x; \xi) - \nabla_i F(x + \alpha e_i; \xi)\| \} \leq L_{(i)} |\alpha|. \quad \forall \xi, \forall x. \quad (7)$$

Define $L_{\max} := \max_{i \in \{1, \dots, N\}} L_{(i)}$. It can be seen that $L_{\max} \leq L_s \leq L$.

Independence All random variables ξ_k, S_k for $k = 0, 1, \dots, K$ in Algorithm 1 are independent to each other.

Bounded Age Let T be the global bound for delay: $J(k) \subset \{k-1, \dots, k-T\}, \forall k$, so $|J(k)| \leq T$.

We define the following global quantities for short notations:

$$\begin{aligned} \omega &:= \left(\sum_{i=1}^N L_{(i)}^2 \mu_i^2 \right) / N, \quad \alpha_1 := 4 + 4 \left(TY + Y^{3/2} T^2 / \sqrt{N} \right) L_T^2 / (L_Y^2 N), \\ \alpha_2 &:= Y / ((f(x_0) - f^*) L_Y N), \quad \alpha_3 := (K(N\omega + \sigma^2) \alpha_2 + 4) L_Y^2 / L_T^2. \end{aligned} \quad (8)$$

Next we show our main result in the following theorem:

Theorem 1 (Generic Convergence Rate for GASA). *Choose the steplength γ_k to be a constant γ in Algorithm 1*

$$\gamma_k^{-1} = \gamma^{-1} = 2L_Y N Y^{-1} \left(\sqrt{\alpha_1^2 / (K(N\omega + \sigma^2) \alpha_2 + \alpha_1)} + \sqrt{K(N\omega + \sigma^2) \alpha_2} \right), \forall k$$

and suppose the age T is bounded by $T \leq \frac{\sqrt{N}}{2Y^{1/2}} \left(\sqrt{1 + 4Y^{-1/2} N^{1/2} \alpha_3} - 1 \right)$. We have the following convergence rate:

$$\frac{\sum_{k=0}^K \mathbb{E} \|\nabla f(x_k)\|^2}{K} \leq \frac{20}{K \alpha_2} + \frac{1}{K \alpha_2} \left(\frac{L_T^2}{L_Y^2} \frac{\sqrt{1 + 4Y^{-1/2} N^{1/2} \alpha_3} - 1}{\sqrt{N Y^{-1}}} + 11 \sqrt{N\omega + \sigma^2} \sqrt{K \alpha_2} \right) + N\omega. \quad (9)$$

Roughly speaking, the first term on the RHS of (9) is related to SCD; the second term is related to “stochastic” gradient descent; and the last term is due to the zeroth-order approximation.

Although this result looks complicated (or may be less elegant), it is capable to capture many important subtle structures, which can be seen by the subsequent discussion. We will show how to recover and improve existing results as well as prove the convergence for new algorithms using Theorem 1. To make the results more interpretable, we use the big-O notation to avoid explicitly writing down all the constant factors, including all L 's, $f(x_0)$, and f^* in the following corollaries.

3.1 Asynchronous Stochastic Coordinate Descent (ASCD)

We apply Theorem 1 to study the asynchronous SCD algorithm by taking $Y = 1$ and $\sigma = 0$. $S_k = \{i_k\}$ only contains a single randomly sampled coordinate, and $\omega = 0$ (or equivalently $\mu_i = 0, \forall i$). The essential updating rule on x is

$$x_{k+1} = x_k - \gamma_k \nabla_{i_k} f(\hat{x}_k).$$

Corollary 2 (ASCD). *Let $\omega = 0, \sigma = 0$, and $Y = 1$ in Algorithm 1 and Theorem 1. If*

$$T \leq O(N^{3/4}), \tag{10}$$

the following convergence rate holds:

$$\left(\sum_{k=0}^K \mathbb{E} \|\nabla f(x_k)\|^2 \right) / K \leq O(N/K). \tag{11}$$

The proved convergence rate $O(N/K)$ is consistent with the existing analysis of SCD [43] or ASCD for smooth optimization [29]. However, our requirement in (10) to ensure the linear speedup property is better than the one in [29], by improving it from $T \leq O(N^{1/2})$ to $T \leq O(N^{3/4})$. Mania et al. [32] analyzed ASCD for strongly convex objectives and proved a linear speedup smaller than $O(N^{1/6})$, which is also more restrictive than ours.

3.2 Asynchronous Stochastic Gradient Descent (ASGD)

ASGD has been widely used to solve deep learning [9, 39, 52], NLP [4, 17], and many other important machine learning problems [38]. There are two typical implementations of ASGD. The first type is to implement on the computer cluster with a parameter server [1, 25]. The parameter server serves as the central node. It can ensure the atomic read or write of the whole vector x and leads to the following updating rule for x (setting $Y = N$ and $\mu_i = 0, \forall i$ in Algorithm 1):

$$x_{k+1} = x_k - \gamma_k \nabla F(\hat{x}_k; \xi_k). \tag{12}$$

Note that a single iteration is defined as modifying the whole vector. The other type is to implement on a single computer with multiple cores. In this case, the central node corresponds to the shared memory. Multiple cores (or threads) can access it simultaneously. However, in this model atomic read and write of x cannot be guaranteed. Therefore, for the purpose of analysis, each update on a single coordinate accounts for an iteration. It turns out to be the following updating rule (setting $S_k = \{i_k\}$, that is, $Y = 1$, and $\mu_i = 0, \forall i$ in Algorithm 1):

$$x_{k+1} = x_k - \gamma_k \nabla_{i_k} F(\hat{x}_k; \xi_k). \tag{13}$$

Readers can refer to [3, 27, 38] for more details and illustrations for these two implementations.

Corollary 3 (ASGD in (12)). Let $\omega = 0$ (or $\mu_i = 0, \forall i$ equivalently) and $Y = N$ in Algorithm 1 and Theorem 1. If

$$T \leq O\left(\sqrt{K\sigma^2 + 1}\right), \quad (14)$$

then the following convergence rate holds:

$$\left(\sum_{k=0}^K \mathbb{E} \|\nabla f(x_k)\|^2\right) / K \leq O\left(\sigma / \sqrt{K} + 1/K\right). \quad (15)$$

First note that the convergence rate in (15) is tight since it is consistent with the serial (nonparallel) version of SGD [35]. We compare this linear speedup property indicated by (14) with results in [1], [15], and [27]. To ensure such rate, Agarwal and Duchi [1] need T to be bounded by $T \leq O(K^{1/4} \min\{\sigma^{3/2}, \sqrt{\sigma}\})$, which is inferior to our result in (14). Feyzmahdavian et al. [15] need T to be bounded by $\sigma^{1/2} K^{1/4}$ to achieve the same rate, which is also inferior to our result. Our requirement is consistent with the one in [27]. To the best of our knowledge, it is the best result so far.

Corollary 4 (ASGD in (13)). Let $\omega = 0$ (or equivalently, $\mu_i = 0, \forall i$) and $Y = 1$ in Algorithm 1 and Theorem 1. If

$$T \leq O\left(\sqrt{N^{3/2} + KN^{1/2}\sigma^2}\right), \quad (16)$$

then the following convergence rate holds

$$\left(\sum_{k=0}^K \mathbb{E} \|\nabla f(x_k)\|^2\right) / K \leq O\left(\sqrt{N/K}\sigma + N/K\right). \quad (17)$$

The additional factor N in (17) (comparing to (15)) arises from the different way of counting the iteration. This additional factor also appears in [38] and [27]. We first compare our result with [27], which requires T to be bounded by $O(\sqrt{KN^{1/2}\sigma^2})$. We can see that our requirement in (17) allows a larger value for T , especially when σ is small such that $N^{3/2}$ dominates $KN^{1/2}\sigma^2$. Next we compare with [38], which assumes that the objective function is strongly convex. Although this is sort of comparing “apple” with “orange”, it is still meaningful if one believes that the strong convexity would not affect the linear speedup property, which is implied by [32]. In [38], the linear speedup is guaranteed if $T \leq O(N^{1/4})$ under the assumption that the sparsity of the stochastic gradient is bounded by $O(1)$. In comparison, we do not require the assumption of sparsity for stochastic gradient and have a better dependence on N . Moreover, beyond the improvement over existing analysis in [32] and [27], our analysis provides some interesting insights for asynchronous parallelism. Niu et al. [38] essentially suggests a large problem dimension N is beneficial to the linear speedup, while Lian et al. [27] and many others (for example, Agarwal and Duchi [1], Feyzmahdavian et al. [15]) suggest that a large stochastic variance σ (this often implies the number of samples is large) is beneficial to the linear speedup. Our analysis shows the combo effect of N and σ and shows how they improve the linear speedup jointly.

3.3 Asynchronous Stochastic Zeroth-order Descent (ASZD)

We end this section by applying Theorem 1 to generate a novel asynchronous zeroth-order stochastic descent algorithm, by setting the block size $Y = 1$ (or equivalently $S_k = \{i_k\}$) in $G_{S_k}(\hat{x}_k; \zeta_k)$

$$G_{S_k}(\hat{x}_k; \zeta_k) = G_{\{i_k\}}(\hat{x}_k; \zeta_k) = \frac{F(\hat{x}_k + \mu_{i_k} e_{i_k}; \zeta_k) - F(\hat{x}_k - \mu_{i_k} e_{i_k}; \zeta_k)}{2\mu_{i_k}} e_{i_k}. \quad (18)$$

To the best of our knowledge, this is the first asynchronous algorithm for zeroth-order optimization.

Corollary 5 (ASZD). *Set $Y = 1$ and all μ_i 's to be a constant μ in Algorithm 1. Suppose that μ satisfies*

$$\mu \leq O\left(1/\sqrt{K} + \min\left\{\sqrt{\sigma}(NK)^{-1/4}, \sigma/\sqrt{N}\right\}\right), \quad (19)$$

and T satisfies

$$T \leq O\left(\sqrt{N^{3/2} + KN^{1/2}\sigma^2}\right). \quad (20)$$

We have the following convergence rate

$$\left(\sum_{k=0}^K \mathbb{E}\|\nabla f(x_k)\|^2\right) / K \leq O\left(N/K + \sqrt{N/K}\sigma\right). \quad (21)$$

We firstly note that the convergence rate in (21) is consistent with the rate for the serial (non-parallel) zeroth-order stochastic gradient method in [16]. Then we evaluate this result from two perspectives.

First, we consider $T = 1$, which leads to the serial (non-parallel) zeroth-order stochastic descent. Our result implies a better dependence on μ , comparing with [16].⁶ To obtain such convergence rate in (21), Ghadimi and Lan [16] require $\mu \leq O\left(1/(N\sqrt{K})\right)$, while our requirement in (19) is much less restrictive. An important insight in our requirement is to suggest the dependence on the variance σ : if the variance σ is large, μ is allowed to be a much larger value. This insight meets the common sense: a large variance means that the stochastic gradient may largely deviate from the true gradient, so we are allowed to choose a large μ to obtain a less exact estimation for the stochastic gradient without affecting the convergence rate. From the practical view of point, it always tends to choose a large value for μ . Recall the zeroth-order method uses the function difference at two different points (e.g., $x + \mu e_i$ and $x - \mu e_i$) to estimate the differential. In a practical system (e.g., a concrete control system), there usually exists some system noise while querying the function values. If two points are too close (in other words μ is too small), the obtained function difference is dominated by noise and does not really reflect the function differential.

⁶Acute readers may notice that our way in (18) to estimate the stochastic gradient is different from the one used in [16]. Our method only estimates a single coordinate gradient of a sampled component function, while Ghadimi and Lan [16] estimate the whole gradient of the sampled component function. Our estimation is more accurate but less aggressive. The proved convergence rate actually improves a small constant in [16].

Second, we consider the case $T \geq 1$, which leads to the asynchronous zeroth-order stochastic descent. To the best of our knowledge, this is the first such algorithm. The upper bound for T in (20) essentially indicates the requirement for the linear speedup property. The linear speedup property here also shows that even if $K\sigma^2$ is much smaller than 1, we still have $O(N^{3/4})$ linear speedup, which shows a fundamental understanding of asynchronous stochastic algorithms that N and σ can improve the linear speedup jointly.

4 Experiment

Since the ASCD and various ASGDs have been extensively validated in recent papers, we conduct two experiments to validate the proposed ASZD on in this section. The first part applies ASZD to estimate the parameters for a synthetic black box system. The second part applies ASZD to the model combination problem in Yahoo Music Recommendation Competition.

4.1 Parameter Optimization for A Black Box

We use a deep neural network to simulate a black box system. The optimization variables are the weights associated with a neural network. We choose 5 layers (400/100/50/20/10 nodes) for the neural network with 46380 weights (or parameters) totally. The weights are randomly generated from i.i.d. Gaussian distribution. The output vector is constructed by applying the network to the input vector plus some Gaussian random noise. We use this network to generate 463800 samples. These synthetic samples are used to optimize the weights for the black box. (We pretend not to know the structure and weights of this neural network because it is a black box.) To optimize (estimate) the parameters for this black box, we apply the proposed ASZD method.

The experiment is conducted on a machine (Intel Xeon architecture), which has 4 sockets and 10 cores for each socket. We run Algorithm 1 on various numbers of cores from 1 to 32 and the steplength is chosen as $\gamma = 0.1$, which is based on the best performance of Algorithm 1 running on 1 core to achieve the precision 10^{-1} for the objective value.

The speedup is reported in Table 2. We observe that the iteration speedup is almost linear while the running time speedup is slightly worse than the iteration speedup. We also draw Figure 1 to show the curve of the objective value against the number of iterations and running time respectively.

Table 2: CCR and RTS of ASZD for different # of threads (synthetic data).

thr-#	1	4	8	12	16	20	24	28	32
CCS	1	3.87	7.91	9.97	14.74	17.86	21.76	26.44	30.86
RTS	1	3.32	6.74	8.48	12.49	15.08	18.52	22.49	26.12

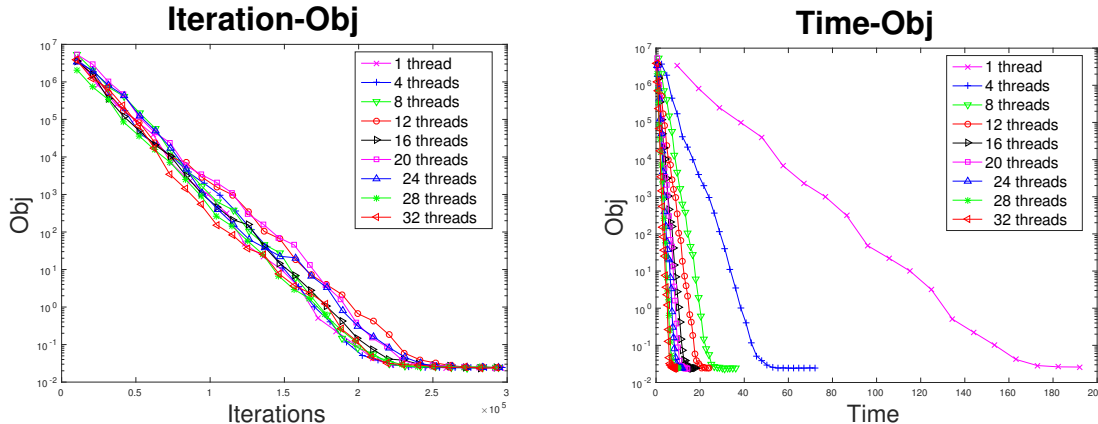


Figure 1: Deep neural network for synthetic data using Algorithm 1. The Algorithm 1 algorithm is run on various numbers of machines from 1 to 32. The curves of the objective loss against the number of iteration and the running time are drawn in the left and the right graphs respectively.

4.2 Asynchronous Parallel Model Combination for Yahoo Music Recommendation Competition

In KDD-Cup 2011, teams were challenged to predict user ratings in music given the Yahoo! Music data set [10]. The evaluation criterion is the Root Mean Squared Error (RMSE) of the test data set:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{T}_1|} \sum_{(u,i) \in \mathcal{T}_1} (r_{ui} - \hat{r}_{ui})^2}, \quad (22)$$

where $(u, i) \in \mathcal{T}_1$ are all user ratings in Track 1 test data set (6,005,940 ratings), r_{ui} is the true rating for user u and item i , and \hat{r}_{ui} is the predicted rating. The winning team from NTU created more than 200 models using different machine learning algorithms [7], including Matrix Factorization, k-NN, Restricted Boltzmann Machines, etc. They blend these models using Neural Network and Binned Linear Regression on the validation data set (4,003,960 ratings) to create a model ensemble to achieve better RMSE.

We were able to obtain the predicted ratings of $N = 237$ individual models on the KDD-Cup test data set from the NTU KDD-Cup team, which is a matrix X with 6,005,940 rows (corresponding to the 6,005,940 test data set samples) and 237 columns. Each element X_{ij} indicates the j -th model's predicted rating on the i -th Yahoo! Music test data sample. In our experiments, we try to linearly

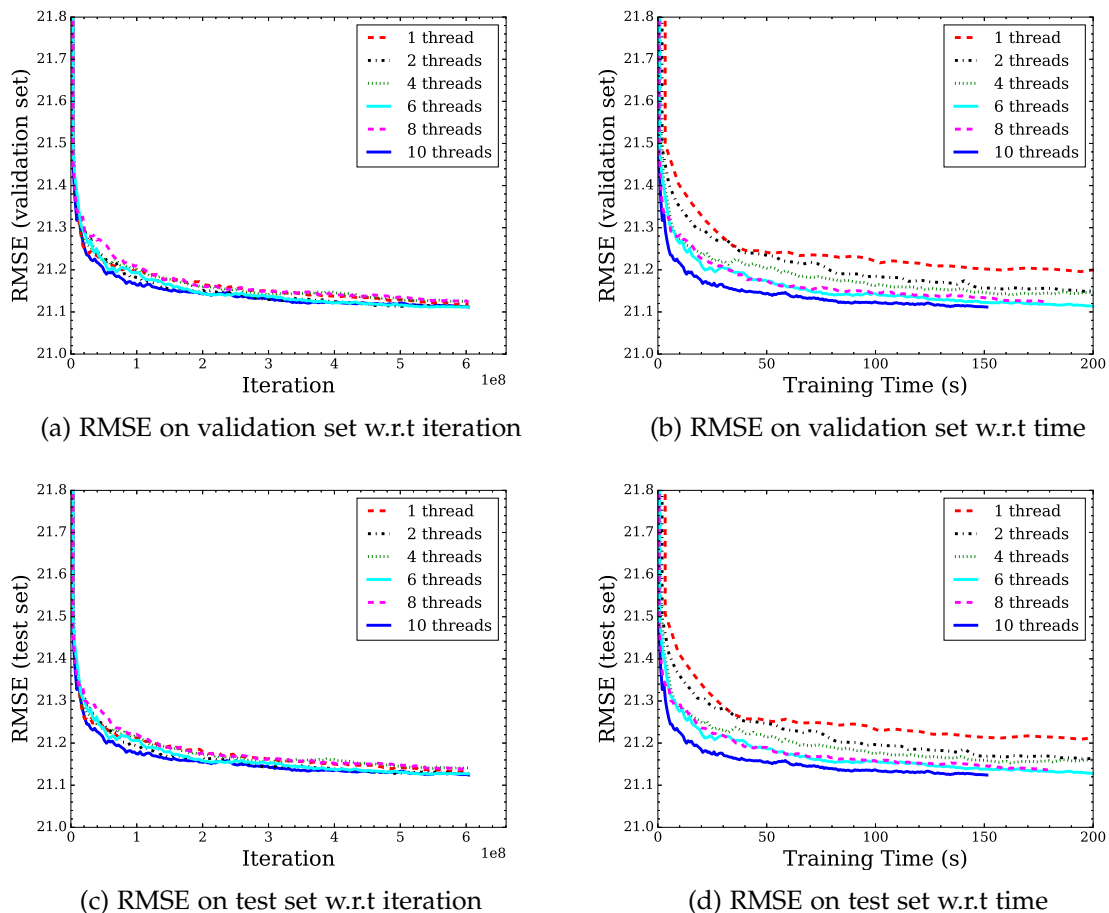


Figure 2: Model blending with Yahoo! Music test data set using ASZD.

blend the 237 models using information from the test data set. Thus, our variable to optimize is a vector $x \in \mathbb{R}^N$ as coefficients of the predicted ratings for each model. To ensure that our linear blending does not over-fit, we further split X randomly into two equal parts, calling them the “validation” set (denoted as $A \in \mathbb{R}^{n \times N}$) for model blending and the true test set.

We define our objective function as RMSE^2 of the blended output on the validation set: $f(x) = \|Ax - r\|^2/n$ where r is the corresponding true r ratings in the validation set and Ax is the predicted ratings after blending.

We assume that we cannot see the entries of r directly, and thus cannot compute the gradient of $f(x)$. In our experiment, we treat $f(x)$ as a blackbox, and the only information we can get from it is its value given a model blending coefficients x . This is similar to submitting a model for KDD-Cup and obtain a leader-board RMSE of the test set; we do not know the actual values of the test set. Then, we apply our ASZD algorithm to minimize $f(x)$ with zero-order information only.

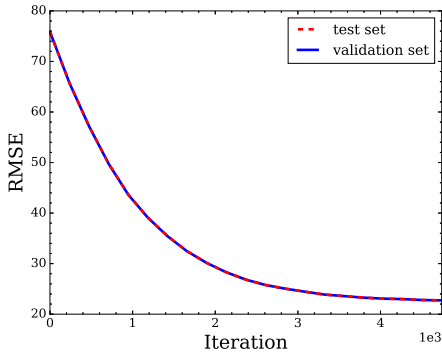


Figure 3: We zoom in for the first a few thousands iterations in Figure 2. Our algorithm quickly approaches to a reasonable RMSE.

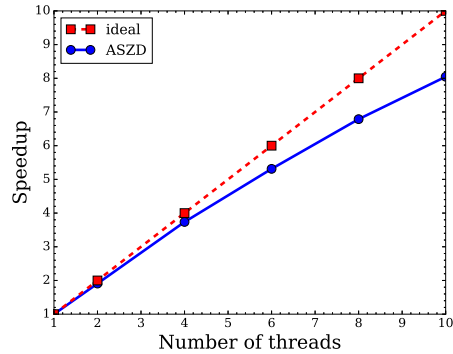


Figure 4: Running time speedup of Algorithm 1 is almost linear. On a 10-core machine we can achieve a 8x speedup.

Table 3: Comparing RMSEs on test data set with KDD-Cup winner teams

	NTU (1st)	Commendo (2nd)	InnerPeace (3rd)	Our result
RMSE	21.0004	21.0545	21.2335	21.1241

We implement our algorithm using Julia on a 10-core Xeon E7-4680 machine and run our algorithm for the same number of iterations, with different number of threads, and measured the running time speedup (RTS) in Figure 4. Similar to our experiment on neural network blackbox, our algorithm has an almost linear speedup. For completeness, Figure 2 shows the square root of objective function value (RMSE) against the number of iterations and running time. We initialize $x = 0$ before training starts. After about 150 seconds, our algorithm running with 10 threads achieves a RMSE of 21.1241 on our test set. Our results are comparable to KDD-Cup winners, as shown in Table 3. Since our goal is to show the performance of our algorithm, we assume we can “submit” our solution x for unlimited times, which is unreal in a real contest like KDD-Cup. However, even with very few iterations, our algorithm does converge fast to a reasonable small RMSE, as shown in Figure 3. It is worth noting that Figure 3 shows only the first a few thousands iterations (Figure 2 shows 6×10^8 iterations), and Figure 3 has a large y-axis range; thus the differences between test and validation RMSE are not visible in this figure.

5 Conclusion

In this paper, we provide a generic linear speedup analysis for the zeroth-order and first-order asynchronous parallel algorithms. Our generic analysis can recover or improve the existing results on special cases, such as ASCD, ASGD (parameter implementation), ASGD (multicore

implementation). Our generic analysis also suggests a novel ASZD algorithm with guaranteed convergence rate and speedup property. To the best of our knowledge, this is the first asynchronous parallel *zeroth* order algorithm. The experiment includes a novel application of the proposed ASZD method on model blending.

6 Acknowledgements

This project is in part supported by the NSF grant CNS-1548078. We especially thank Chen-Tse Tsai for providing the code and data for the Yahoo Music Competition.

References

- [1] A. Agarwal and J. C. Duchi. Distributed delayed stochastic optimization. *NIPS*, 2011.
- [2] A. Agarwal, D. P. Foster, D. J. Hsu, S. M. Kakade, and A. Rakhlin. Stochastic convex optimization with bandit feedback. In *NIPS*, pages 1035–1043, 2011.
- [3] H. Avron, A. Druinsky, and A. Gupta. Revisiting asynchronous linear solvers: Provable convergence rate through randomization. *Journal of the ACM (JACM)*, 62(6):51, 2015.
- [4] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.
- [5] N. Cesa-Bianchi, C. Gentile, Y. Mansour, and A. Minora. Delay and cooperation in non-stochastic bandits. *CoRR*, abs/1602.04741, 2016.
- [6] S. Chaturapruek, J. C. Duchi, and C. Ré. Asynchronous stochastic convex optimization: the noise is in the noise and SGD don’t care. In *NIPS*, pages 1531–1539, 2015.
- [7] P.-L. Chen, C.-T. Tsai, Y.-N. Chen, K.-C. Chou, C.-L. Li, C.-H. Tsai, K.-W. Wu, Y.-C. Chou, C.-Y. Li, W.-S. Lin, et al. A linear ensemble of individual and blended models for music rating prediction. In *KDDCup*, 2012.
- [8] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to derivative-free optimization*, volume 8. Siam, 2009.
- [9] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al. Large scale distributed deep networks. *NIPS*, 2012.
- [10] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer. The Yahoo! Music dataset and KDD-Cup’11. In *KDDCup*, pages 8–18, 2012.
- [11] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012.

- [12] J. C. Duchi, S. Chaturapruek, and C. Ré. Asynchronous stochastic convex optimization. *arXiv:1508.00882*, 2015.
- [13] J. Fellus, D. Picard, and P. H. Gosselin. Asynchronous gossip principal components analysis. *Neurocomputing*, 2015. doi: 10.1016/j.neucom.2014.11.076.
- [14] H. R. Feyzmahdavian. Performance analysis of positive systems and optimization algorithms with time-delays. 2016.
- [15] H. R. Feyzmahdavian, A. Aytakin, and M. Johansson. An asynchronous mini-batch algorithm for regularized stochastic optimization. *arXiv*, 2015.
- [16] S. Ghadimi and G. Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.
- [17] K. Gimpel, D. Das, and N. A. Smith. Distributed asynchronous online learning for natural language processing. In *CoNLL*, pages 213–222, 2010.
- [18] M. Gurbuzbalaban, A. Ozdaglar, and P. Parrilo. On the convergence rate of incremental aggregated gradient algorithms. *arXiv:1506.02081*, 2015.
- [19] M. Hong, M. Razaviyayn, Z. Luo, and J. Pang. A unified algorithmic framework for block-structured optimization involving big data: With applications in machine learning and signal processing. *IEEE Signal Process. Mag.*, 33(1):57–77, 2016. doi: 10.1109/MSP.2015.2481563.
- [20] C. Hsieh, H. Yu, and I. S. Dhillon. PASSCoDe: Parallel ASynchronous Stochastic dual Co-ordinate Descent. In *ICML*, pages 2370–2379, 2015.
- [21] K. G. Jamieson, R. Nowak, and B. Recht. Query complexity of derivative-free optimization. In *NIPS*, 2012.
- [22] S. Kumar, R. Jain, and K. Rajawat. Asynchronous optimization over heterogeneous networks via consensus admm. *arXiv:1605.00076*, 2016.
- [23] K. Lee and R. Bhattacharya. On the convergence analysis of asynchronous distributed quadratic programming via dual decomposition. *arXiv:1506.05485*, abs/1506.05485, 2015.
- [24] M. Li, L. Zhou, Z. Yang, A. Li, F. Xia, D. G. Andersen, and A. Smola. Parameter server for distributed machine learning. *Big Learning NIPS Workshop*, 2013.
- [25] 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. *OSDI*, 2014.
- [26] M. Li, D. G. Andersen, A. J. Smola, and K. Yu. Communication efficient distributed machine learning with the parameter server. *NIPS*, 2014.
- [27] X. Lian, Y. Huang, Y. Li, and J. Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *NIPS*, pages 2719–2727, 2015.

- [28] J. Liu and S. J. Wright. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *arXiv:1403.3862*, 2014.
- [29] J. Liu, S. J. Wright, C. Ré, V. Bittorf, and S. Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. *ICML*, 2014.
- [30] J. Liu, S. J. Wright, and S. Sridhar. An asynchronous parallel randomized kaczmarz algorithm. *arXiv*, 2014.
- [31] Z. Lu and L. Xiao. On the complexity analysis of randomized block-coordinate descent methods. *Mathematical Programming*, 152(1-2):615–642, 2015.
- [32] H. Mania, X. Pan, D. Papailiopoulos, B. Recht, K. Ramchandran, and M. I. Jordan. Perturbed iterate analysis for asynchronous stochastic optimization. *arXiv:1507.06970*, 2015.
- [33] J. Marecek, P. Richtárik, and M. Takáč. Distributed block coordinate descent for minimizing partially separable functions. *arXiv:1406.0238*, 2014.
- [34] E. Moulines and F. R. Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. *NIPS*, 2011.
- [35] 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.
- [36] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.
- [37] Y. Nesterov and V. Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, pages 1–40, 2011.
- [38] F. Niu, B. Recht, C. Re, and S. Wright. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. *NIPS*, 2011.
- [39] T. Paine, H. Jin, J. Yang, Z. Lin, and T. Huang. Gpu asynchronous stochastic gradient descent to speed up neural network training. *NIPS*, 2013.
- [40] Z. Peng, Y. Xu, M. Yan, and W. Yin. Arock: an algorithmic framework for asynchronous parallel coordinate updates. *arXiv*, 2015.
- [41] F. Petroni and L. Querzoni. Gasgd: stochastic gradient descent for distributed asynchronous matrix completion via graph partitioning. *ACM Conference on Recommender systems*, 2014.
- [42] S. J. Reddi, A. Hefny, S. Sra, B. Póczos, and A. J. Smola. On variance reduction in stochastic gradient descent and its asynchronous variants. In *NIPS*, pages 2629–2637, 2015.
- [43] P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.
- [44] C. D. Sa, C. Zhang, K. Olukotun, C. Ré, and C. Ré. Taming the wild: A unified analysis of hogwild-style algorithms. In *NIPS*, pages 2674–2682, 2015.

- [45] S. Sridhar, S. Wright, C. Re, J. Liu, V. Bittorf, and C. Zhang. An approximate, efficient LP solver for lp rounding. *NIPS*, 2013.
- [46] R. Tappenden, M. Takáč, and P. Richtárik. On the complexity of parallel coordinate descent. *arXiv:1503.03033*, 2015.
- [47] K. Tran, S. Hosseini, L. Xiao, T. Finley, and M. Bilenko. Scaling up stochastic dual coordinate ascent. *ICML*, 2015.
- [48] J. Wei, W. Dai, A. Kumar, X. Zheng, Q. Ho, and E. P. Xing. Consistent bounded-asynchronous parameter servers for distributed ml. *arXiv:1312.7869*, 2013.
- [49] H. Yun, H.-F. Yu, C.-J. Hsieh, S. Vishwanathan, and I. Dhillon. Nomad: Non-locking, stochastic multi-machine algorithm for asynchronous and decentralized matrix completion. *arXiv:1312.0193*, 2013.
- [50] C. Zhang and C. Re. Dimmwitted: A study of main-memory statistical analytics. *PVLDB*, 7(12):1283–1294, 2014.
- [51] R. Zhang, S. Zheng, and J. T. Kwok. Asynchronous distributed semi-stochastic gradient optimization. In *AAAI*, pages 2323–2329, 2016.
- [52] S. Zhang, A. Choromanska, and Y. LeCun. Deep learning with elastic averaging SGD. *arXiv*, 2014.
- [53] Y. Zhang and M. I. Jordan. Splash: User-friendly programming interface for parallelizing stochastic algorithms. *CoRR*, abs/1506.07552, 2015.
- [54] S. Zhao and W. Li. Fast asynchronous parallel stochastic gradient descent: A lock-free approach with convergence guarantee. In *AAAI*, pages 2379–2385, 2016.

A Appendix

We first show some preliminary results about the dependence between the zeroth-order gradient and the true gradient, and then prove the convergence property for GASA (Algorithm 1).

A.1 Preliminary results

Given a function $p(x) : \mathbb{R}^N \rightarrow \mathbb{R}$ and finite difference step sizes vector $(\mu_1, \mu_2, \dots, \mu_N)^\top \in \mathbb{R}_{++}^N$, we define the smoothing function of $p(x)$ with respect to the i th dimension:

$$p_i(x) := \mathbb{E}_{\{v \sim U_{[-\mu_i, \mu_i]}\}}(p(x + ve_i)) = \frac{1}{2\mu_i} \int_{-\mu_i}^{\mu_i} p(x + ve_i) dv = \frac{1}{2} \int_{-1}^1 p(x + v\mu_i e_i) dv. \quad (23)$$

where $v \sim U_{[-\mu_i, \mu_i]}$ means that v follows the uniform distribution over the range $[-\mu_i, \mu_i]$. ∇_x and ∇_v denote taking gradient with respect to variable x and v respectively. ∇_i is short for $e_i e_i^\top \nabla_x$.

Lemma 6 (Zeroth Order Approximation). *Suppose that function $p(x) : x \in \mathbb{R}^N \rightarrow \mathbb{R}$ has Lipschitzian gradient with parameter L and $p(x + ve_i) : v \in \mathbb{R} \rightarrow \mathbb{R}$ has Lipschitzian gradient with parameter $L_{(i)}$ for all x . Given a finite difference step size vector $(\mu_1, \mu_2, \dots, \mu_N)^\top \in \mathbb{R}_{++}^N$, we have:*

1. The smoothing function p_i is differentiable, and also has the Lipschitzian gradient with Lipschitz constant L .
2. For the i^{th} coordinate,

$$\nabla_i p_i(x) = \frac{1}{2\mu_i} (p(x + \mu_i e_i) - p(x - \mu_i e_i)) e_i.$$

3. For any $x \in \mathbb{R}^N, i \in \{1, \dots, N\}$, we have

$$|p_i(x) - p(x)| \leq \frac{L_{(i)} \mu_i^2}{2}, \quad (24)$$

$$\|\nabla_i p_i(x) - \nabla_i p(x)\| \leq \frac{L_{(i)} \mu_i}{2}, \quad (25)$$

$$\mathbb{E}_i \|\nabla_i p_i(x) - \nabla_i p(x)\|^2 \leq \frac{\omega}{4}, \quad (26)$$

where ω is defined in (8).

4. If p is convex, then p_i is also convex.

Proof. To prove the first statement, we note that

$$\nabla p_i(x) = \frac{1}{2} \int_{-1}^1 \nabla_x p(x + \mu_i v e_i) dv.$$

Since function $p(x)$ has Lipschitzian gradient, we have

$$\begin{aligned}\|\nabla p_i(x) - \nabla p_i(y)\| &= \|\mathbb{E}_{\{v \sim U_{[-1,1]}\}}(\nabla p(x + \mu_i v e_i) - \nabla p(y + \mu_i v e_i))\| \\ &\leq \mathbb{E}_{\{v \sim U_{[-1,1]}\}}(L\|x - y\|) \\ &= L\|x - y\|,\end{aligned}$$

which implies the first statement.

To prove the second statement, we have

$$\begin{aligned}\nabla_i p_i(x) &= \frac{1}{2} \left(\int_{-1}^1 \nabla_i p(x + \mu_i v e_i) dv \right) \\ &= \frac{1}{2\mu_i} \left(\int_{-1}^1 \nabla_v p(x + \mu_i v e_i) dv \right) e_i \\ &= \frac{1}{2\mu_i} (p(x + \mu_i e_i) - p(x - \mu_i e_i)) e_i,\end{aligned}$$

where the last step comes from the Stokes' theorem.

Next we prove the third statement. First (24) can be proved by

$$\begin{aligned}|p_i(x) - p(x)| &\leq \frac{1}{2} \left| \int_{-1}^1 p(x + \mu_i v e_i) - p(x + 0\mu_i e_i) dv \right| \\ &= \frac{1}{2} \left| \int_{-1}^1 p(x + \mu_i v e_i) - p(x + 0\mu_i e_i) - \langle \nabla_v p(x), \mu_i v \rangle dv \right| \\ &\leq \frac{1}{2} \int_{-1}^1 |p(x + \mu_i v e_i) - p(x) - \langle \nabla_v p(x), \mu_i v \rangle| dv \\ &\leq \frac{1}{2} \int_{-1}^1 \frac{L_{(i)} \mu_i^2 v^2}{2} dv \\ &\leq \frac{L_{(i)} \mu_i^2}{2},\end{aligned}$$

where the second step is based on the observation

$$\int_{-1}^1 \langle \nabla_v p(x), \mu_i v \rangle dv = 0,$$

and the second last step uses the assumption that $p(x + v e_i)$ has Lipschitzian gradient (with constant $L_{(i)}$) with respect to v . (25) can be proved from

$$\begin{aligned}\|\nabla_i p_i(x) - \nabla_i p(x)\| &= \left\| \frac{1}{2\mu_i} (p(x + \mu_i e_i) - p(x - \mu_i e_i)) e_i - \nabla_i p(x) \right\| \\ &= \left\| \frac{1}{2\mu_i} ((p(x + \mu_i e_i) - p(x - \mu_i e_i)) e_i - 2\mu_i \nabla_i p(x)) \right\| \\ &\leq \left\| \frac{1}{2\mu_i} ((p(x + \mu_i e_i) - p(x)) e_i - \mu_i \nabla_i p(x)) \right\|\end{aligned}$$

$$\begin{aligned}
& + \left\| \frac{1}{2\mu_i} ((p(x) - p(x - \mu_i e_i)) e_i - \mu_i \nabla_i p(x)) \right\| \\
& \leq \frac{1}{2\mu_i} L_{(i)} \mu_i^2 = \frac{L_{(i)} \mu_i}{2}.
\end{aligned}$$

(26) can be proved by

$$\mathbb{E}_i \|\nabla_i p_i(x) - \nabla_i p(x)\|^2 \stackrel{(25)}{\leq} \sum_i \frac{L_{(i)}^2 \mu_i^2}{4N} = \frac{\omega}{4}.$$

For the last statement, given any $\theta \in [0, 1]$, we have

$$\begin{aligned}
\theta p_i(x) + (1 - \theta) p_i(y) &= \frac{1}{2} \int_{-1}^1 \theta p(x + \mu_i v e_i) + (1 - \theta) p(y + \mu_i v e_i) dv \\
&\geq \frac{1}{2} \int_{-1}^1 p(\theta x + (1 - \theta) y + \mu_i v e_i) dv \\
&= p_i(\theta x + (1 - \theta) y).
\end{aligned}$$

It proves the convexity of $p_i(x)$. □

A.2 Proofs to Theorem 1

We first prove the general convergence property for Algorithm 1 and then apply it to prove Theorem 1.

Theorem 7 (Convergence). *If the stepsize γ_k in Algorithm 1 is appropriately chosen to satisfy*

$$\Theta_k := \frac{N\gamma_k}{2} - 2\gamma_k^2 \frac{L_Y}{Y} N^2 - 2L_T^2 \frac{N^2}{Y^2} \gamma_k \sum_{v=1}^T \gamma_{k+v} \left(\gamma_k Y + \frac{Y^{3/2} \sum_{j' \in J(k+v) \setminus \{k\}} \gamma_{j'}}{\sqrt{N}} \right) \geq 0, \forall k, \quad (27)$$

we have

$$\begin{aligned}
\frac{1}{2} \sum_{k=0}^K \gamma_k \mathbb{E} \|\nabla f(x_k)\|^2 &\leq f(x_0) - f^* + 2N \frac{L_T^2}{Y} \left(\frac{3N\omega}{2} + 3\sigma^2 \right) \sum_{k=0}^K \left(\gamma_k \sum_{j \in J(k)} \gamma_j^2 \right) \\
&\quad + \left(\frac{L_Y}{Y} N\sigma^2 + \frac{L_Y}{Y} N^2 \omega \right) \sum_{k=0}^K \gamma_k^2 + \frac{1}{4} N\omega \sum_{k=0}^K \gamma_k.
\end{aligned}$$

Proof. Besides of (3), we introduce one more notation here to abbreviate the notation:

$$G_i(x; \xi) = \frac{N}{2\mu_i} (F(x + \mu_i e_i; \xi) - F(x - \mu_i e_i; \xi)) e_i.$$

Note that $G_i(x; \xi) = N \nabla_i F_i(x; \xi)$ for any x and ξ , where $F_i(\cdot; \xi)$ is defined using the same way in (23) for $p = F(\cdot; \xi)$. We first highlight a frequently used inequality in the subsequent proof:

$$\|a + b\|^2 \leq 2\|a\|^2 + 2\|b\|^2. \quad (28)$$

We then start from the Lipschitzian property of $f(\cdot)$:

$$\begin{aligned}
f(x_{k+1}) - f(x_k) &\leq \langle \nabla_{S_k} f(x_k), x_{k+1} - x_k \rangle + \frac{L_Y}{2} \|x_{k+1} - x_k\|^2 \\
&= -\gamma_k \langle \nabla_{S_k} f(x_k), G_{S_k}(\hat{x}_k; \xi_k) \rangle \\
&\quad + \frac{1}{2} \gamma_k^2 L_Y \|G_{S_k}(\hat{x}_k; \xi_k)\|^2 \\
&\stackrel{(28)}{\leq} -\gamma_k \langle \nabla_{S_k} f(x_k), G_{S_k}(\hat{x}_k; \xi_k) \rangle \\
&\quad + \gamma_k^2 L_Y \left\| \frac{N}{Y} \nabla_{S_k} F(\hat{x}_k; \xi_k) \right\|^2 \\
&\quad + \gamma_k^2 L_Y \left\| G_{S_k}(\hat{x}_k; \xi_k) - \frac{N}{Y} \nabla_{S_k} F(\hat{x}_k; \xi_k) \right\|^2. \tag{29}
\end{aligned}$$

Next we consider the expectation of three items of the right hand side of (29). Let $\{i_k\}_{k=0}^K$ be independent random variables uniformly distributed over $\{1, \dots, N\}$. Note that i_k is just a dummy random variable independent of all S_k 's. Let Ω_k be the set consisting of all possible S_k 's. With this new variable, we can rewrite $\mathbb{E}_{S_k} \langle \nabla_{S_k} f(x_k), G_{S_k}(\hat{x}_k; \xi_k) \rangle$ by

$$\begin{aligned}
&\mathbb{E}_{S_k} \langle \nabla_{S_k} f(x_k), G_{S_k}(\hat{x}_k; \xi_k) \rangle \\
&= \frac{1}{Y} \mathbb{E}_{S_k} \sum_{m \in S_k} \langle \nabla_m f(x_k), G_m(\hat{x}_k; \xi_k) \rangle \\
&= \frac{1}{Y |\Omega_k|} \sum_{S \in \Omega_k} \sum_{m \in S} \langle \nabla_m f(x_k), G_m(\hat{x}_k; \xi_k) \rangle \\
&= \frac{1}{Y |\Omega_k|} \frac{|\Omega_k| Y}{N} \left\langle \nabla f(x_k), \sum_{n \in [N]} G_n(\hat{x}_k; \xi_k) \right\rangle \\
&= \frac{1}{Y} Y \mathbb{E}_{i_k} \langle \nabla_{i_k} f(x_k), G_{i_k}(\hat{x}_k; \xi_k) \rangle \\
&= \mathbb{E}_{i_k} \langle \nabla_{i_k} f(x_k), G_{i_k}(\hat{x}_k; \xi_k) \rangle, \tag{30}
\end{aligned}$$

where the second last step is due to the fact that S_k is selected uniformly randomly. Following similar derivation yields:

$$\begin{aligned}
\mathbb{E}_{S_k} \left\| \frac{N}{Y} \nabla_{S_k} F(\hat{x}_k; \xi_k) \right\|^2 &= \frac{N^2}{Y^2} \mathbb{E}_{S_k} \sum_{m \in S_k} \|\nabla_m F(\hat{x}_k; \xi_k)\|^2 \\
&= \frac{N^2}{Y^2} Y \mathbb{E}_{i_k} \|\nabla_{i_k} F(\hat{x}_k; \xi_k)\|^2 \\
&= \frac{N^2}{Y} \mathbb{E}_{i_k} \|\nabla_{i_k} F(\hat{x}_k; \xi_k)\|^2 \\
&= \frac{1}{Y} \mathbb{E}_{i_k} \|N \nabla_{i_k} F(\hat{x}_k; \xi_k)\|^2, \tag{31}
\end{aligned}$$

and

$$\mathbb{E}_{S_k} \left\| G_{S_k}(\hat{x}_k; \xi_k) - \frac{N}{Y} \nabla_{S_k} F(\hat{x}_k; \xi_k) \right\|^2$$

$$\begin{aligned}
&= \mathbb{E}_{S_k} \sum_{m \in S_k} \left\| \frac{1}{Y} G_m(\hat{x}_k; \zeta_k) - \frac{N}{Y} \nabla_m F(\hat{x}_k; \zeta_k) \right\|^2 \\
&= \frac{1}{Y^2} \mathbb{E}_{S_k} \sum_{m \in S_k} \|G_m(\hat{x}_k; \zeta_k) - N \nabla_m F(\hat{x}_k; \zeta_k)\|^2 \\
&= \frac{1}{Y^2} Y \mathbb{E}_{i_k} \|G_{i_k}(\hat{x}_k; \zeta_k) - N \nabla_{i_k} F(\hat{x}_k; \zeta_k)\|^2 \\
&= \frac{1}{Y} \mathbb{E}_{i_k} \|G_{i_k}(\hat{x}_k; \zeta_k) - N \nabla_{i_k} F(\hat{x}_k; \zeta_k)\|^2. \tag{32}
\end{aligned}$$

We give two more equalities which will be used soon in the following:

$$\begin{aligned}
\mathbb{E}_{\zeta_k} \|\nabla F(\hat{x}_k; \zeta_k)\|^2 &= \mathbb{E}_{\zeta_k} (\|\nabla F(\hat{x}_k; \zeta_k) - \nabla f(\hat{x}_k) + \nabla f(\hat{x}_k)\|^2) \\
&= \mathbb{E}_{\zeta_k} (\|\nabla F(\hat{x}_k; \zeta_k) - \nabla f(\hat{x}_k)\|^2 + \|\nabla f(\hat{x}_k)\|^2) \\
&\quad + 2\mathbb{E}_{\zeta_k} \langle \nabla F(\hat{x}_k; \zeta_k) - \nabla f(\hat{x}_k), \nabla f(\hat{x}_k) \rangle \\
&= \mathbb{E}_{\zeta_k} (\|\nabla F(\hat{x}_k; \zeta_k) - \nabla f(\hat{x}_k)\|^2 + \|\nabla f(\hat{x}_k)\|^2) \\
&\quad + 2\langle \nabla f(\hat{x}_k) - \nabla f(\hat{x}_k), \nabla f(\hat{x}_k) \rangle \\
&= \mathbb{E}_{\zeta_k} (\|\nabla F(\hat{x}_k; \zeta_k) - \nabla f(\hat{x}_k)\|^2 + \|\nabla f(\hat{x}_k)\|^2), \tag{33}
\end{aligned}$$

and

$$\begin{aligned}
&-2\langle \nabla_{i_k} f(x_k), \nabla_{i_k} f_{i_k}(\hat{x}_k) \rangle \\
&= \|\nabla_{i_k} f(x_k) - \nabla_{i_k} f_{i_k}(\hat{x}_k)\|^2 - \|\nabla_{i_k} f(x_k)\|^2 - \|\nabla_{i_k} f_{i_k}(\hat{x}_k)\|^2. \tag{34}
\end{aligned}$$

We then put (30), (31), (32) into (29):

$$\begin{aligned}
\mathbb{E}_{\zeta_k, S_k} (f(x_{k+1}) - f(x_k)) &\leq -\gamma_k \mathbb{E}_{\zeta_k, i_k} \langle \nabla_{i_k} f(x_k), G_{i_k}(\hat{x}_k; \zeta_k) \rangle \\
&\quad + \gamma_k^2 \frac{LY}{Y} \mathbb{E}_{\zeta_k, i_k} \|N \nabla_{i_k} F(\hat{x}_k; \zeta_k)\|^2 \\
&\quad + \gamma_k^2 \frac{LY}{Y} \mathbb{E}_{\zeta_k, i_k} \|G_{i_k}(\hat{x}_k; \zeta_k) - N \nabla_{i_k} F(\hat{x}_k; \zeta_k)\|^2 \\
&= -\gamma_k \mathbb{E}_{i_k} \langle \nabla_{i_k} f(x_k), N \nabla_{i_k} f_{i_k}(\hat{x}_k) \rangle \\
&\quad + \gamma_k^2 \frac{LY}{Y} N \mathbb{E}_{\zeta_k} \|\nabla F(\hat{x}_k; \zeta_k)\|^2 \\
&\quad + \gamma_k^2 \frac{LY}{Y} \mathbb{E}_{\zeta_k, i_k} \|G_{i_k}(\hat{x}_k; \zeta_k) - N \nabla_{i_k} F(\hat{x}_k; \zeta_k)\|^2 \\
&\stackrel{(33)}{=} -\gamma_k N \mathbb{E}_{i_k} \langle \nabla_{i_k} f(x_k), \nabla_{i_k} f_{i_k}(\hat{x}_k) \rangle \\
&\quad + \gamma_k^2 \frac{LY}{Y} N \overbrace{(\mathbb{E}_{\zeta_k} \|\nabla F(\hat{x}_k; \zeta_k) - \nabla f(\hat{x}_k)\|^2 + \|\nabla f(\hat{x}_k)\|^2)}^{\leq \sigma^2} \\
&\quad + \gamma_k^2 \frac{LY}{Y} \overbrace{\mathbb{E}_{\zeta_k, i_k} \|G_{i_k}(\hat{x}_k; \zeta_k) - N \nabla_{i_k} F(\hat{x}_k; \zeta_k)\|^2}^{\leq N^2 \omega / 4 \text{ from (26)}}
\end{aligned}$$

$$\begin{aligned}
&\stackrel{(34)}{\leq} -\frac{\gamma_k}{2} \left(\|\nabla f(x_k)\|^2 + N\mathbb{E}_{i_k} \|\nabla_{i_k} f_{i_k}(\hat{x}_k)\|^2 \right) \\
&\quad + \frac{\gamma_k}{2} \underbrace{N\mathbb{E}_{i_k} \|\nabla_{i_k} f(x_k) - \nabla_{i_k} f_{i_k}(\hat{x}_k)\|^2}_{=:T_1} \\
&\quad + \gamma_k^2 \frac{L_Y}{Y} N(\sigma^2 + \|\nabla f(\hat{x}_k)\|^2) \\
&\quad + \gamma_k^2 \frac{L_Y}{Y} N^2 \frac{\omega}{4}. \tag{35}
\end{aligned}$$

Then we study the upper bound for T_1 :

$$\begin{aligned}
T_1 &= \mathbb{E}_{i_k} \|\nabla_{i_k} f(x_k) - \nabla_{i_k} f_{i_k}(\hat{x}_k)\|^2 \\
&= \mathbb{E}_{i_k} \|\nabla_{i_k} f(x_k) - \nabla_{i_k} f(\hat{x}_k) + \nabla_{i_k} f(\hat{x}_k) - \nabla_{i_k} f_{i_k}(\hat{x}_k)\|^2 \\
&\stackrel{(28)}{\leq} 2\mathbb{E}_{i_k} \left(\|\nabla_{i_k} f(x_k) - \nabla_{i_k} f(\hat{x}_k)\|^2 + \|\nabla_{i_k} f(\hat{x}_k) - \nabla_{i_k} f_{i_k}(\hat{x}_k)\|^2 \right) \\
&\stackrel{(26)}{\leq} 2\mathbb{E}_{i_k} \|\nabla_{i_k} f(x_k) - \nabla_{i_k} f(\hat{x}_k)\|^2 + \frac{\omega}{2} \\
&\leq \frac{2}{N} L_T^2 \|x_k - \hat{x}_k\|^2 + \frac{\omega}{2} \\
&= \frac{2}{N} L_T^2 \underbrace{\left\| \sum_{j \in J(k)} (x_{j+1} - x_j) \right\|^2}_{=:T_2} + \frac{\omega}{2}. \tag{36}
\end{aligned}$$

We then bound T_2 by

$$\begin{aligned}
\mathbb{E}(T_2) &= \mathbb{E} \left\| \sum_{j \in J(k)} (x_{j+1} - x_j) \right\|^2 \\
&= \mathbb{E} \left\| \sum_{j \in J(k)} \gamma_j G_{S_j}(\hat{x}_j; \xi_j) \right\|^2 \\
&= \mathbb{E} \left\| \sum_{j \in J(k)} \left(\gamma_j \left(G_{S_j}(\hat{x}_j; \xi_j) - \frac{N}{Y} \sum_{m \in S_j} \nabla_m f_m(\hat{x}_j) \right) + \frac{N}{Y} \sum_{m \in S_j} \gamma_j \nabla_m f_m(\hat{x}_j) \right) \right\|^2 \\
&\stackrel{(28)}{\leq} 2\mathbb{E} \underbrace{\left\| \sum_{j \in J(k)} \gamma_j \left(G_{S_j}(\hat{x}_j; \xi_j) - \frac{N}{Y} \sum_{m \in S_j} \nabla_m f_m(\hat{x}_j) \right) \right\|^2}_{=:T_3}
\end{aligned}$$

$$+2\frac{N^2}{Y^2}\mathbb{E}\left\|\underbrace{\sum_{j\in J(k)}\gamma_j\sum_{m\in S_j}\nabla_m f_m(\hat{x}_j)}_{=:T_4}\right\|^2. \quad (37)$$

Next we show the upper bounds for $\mathbb{E}(T_4)$ and $\mathbb{E}(T_3)$ respectively:

$$\begin{aligned} & \mathbb{E}(T_4) \\ &= \mathbb{E}\left\|\sum_{j\in J(k)}\gamma_j\sum_{m\in S_j}\nabla_m f_m(\hat{x}_j)\right\|^2 \\ &= \mathbb{E}\left(\sum_{j\in J(k)}\gamma_j^2\left\|\sum_{m\in S_j}\nabla_m f_m(\hat{x}_j)\right\|^2\right) \\ &\quad +2\mathbb{E}\left(\sum_{j,j'\in J(k),j>j'}\gamma_j\gamma_{j'}\left(\left\langle\sum_{m\in S_j}\nabla_m f_m(\hat{x}_j),\sum_{m'\in S_{j'}}\nabla_{m'} f_{m'}(\hat{x}_{j'})\right\rangle\right)\right) \\ &= \mathbb{E}\left(\sum_{j\in J(k)}\gamma_j^2 Y \mathbb{E}_{i_j}\left\|\nabla_{i_j} f_{i_j}(\hat{x}_j)\right\|^2\right) \\ &\quad +2\mathbb{E}\left(\sum_{j,j'\in J(k),j>j'}\gamma_j\gamma_{j'}\left\langle\mathbb{E}_{S_j}\sum_{m\in S_j}\nabla_m f_m(\hat{x}_j),\sum_{m'\in S_{j'}}\nabla_{m'} f_{m'}(\hat{x}_{j'})\right\rangle\right) \\ &\leq \sum_{j\in J(k)}\gamma_j^2 Y \mathbb{E}\left\|\nabla_{i_j} f_{i_j}(\hat{x}_j)\right\|^2 \\ &\quad +\mathbb{E}\left(\sum_{j,j'\in J(k),j>j'}\gamma_j\gamma_{j'}\left(\frac{1}{\alpha}\left\|\mathbb{E}_{S_j}\sum_{m\in S_j}\nabla_m f_m(\hat{x}_j)\right\|^2+\alpha\mathbb{E}_{S_{j'}}\left\|\sum_{m'\in S_{j'}}\nabla_{m'} f_{m'}(\hat{x}_{j'})\right\|^2\right)\right) \\ &\leq \sum_{j\in J(k)}\gamma_j^2 Y \mathbb{E}\left\|\nabla_{i_j} f_{i_j}(\hat{x}_j)\right\|^2 \\ &\quad +\mathbb{E}\left(\sum_{j,j'\in J(k),j>j'}\gamma_j\gamma_{j'}\left(\frac{1}{\alpha}\left\|Y\mathbb{E}_{i_j}\nabla_{i_j} f_{i_j}(\hat{x}_j)\right\|^2+\alpha\mathbb{E}_{S_{j'}}\left\|\sum_{m'\in S_{j'}}\nabla_{m'} f_{m'}(\hat{x}_{j'})\right\|^2\right)\right) \\ &\leq \sum_{j\in J(k)}\gamma_j^2 Y \mathbb{E}\left\|\nabla_{i_j} f_{i_j}(\hat{x}_j)\right\|^2 \\ &\quad +\mathbb{E}\left(\sum_{j,j'\in J(k),j>j'}\gamma_j\gamma_{j'}\left(\frac{Y^2}{\alpha N}\mathbb{E}_{i_j}\left\|\nabla_{i_j} f_{i_j}(\hat{x}_j)\right\|^2+\alpha Y\mathbb{E}_{i_{j'}}\left\|\nabla_{i_{j'}} f_{i_{j'}}(\hat{x}_{j'})\right\|^2\right)\right) \\ &= \sum_{j\in J(k)}\gamma_j\left(\gamma_j Y+\alpha Y\sum_{j'>j,j'\in J(k)}\gamma_{j'}+\frac{Y^2\sum_{j'<j,j'\in J(k)}\gamma_{j'}}{\alpha N}\right)\mathbb{E}\left\|\nabla_{i_j} f_{i_j}(\hat{x}_j)\right\|^2 \end{aligned}$$

$$= \sum_{j \in J(k)} \gamma_j \left(\gamma_j Y + \frac{Y^{3/2} \sum_{j' \in J(k) \setminus \{j\}} \gamma_{j'}}{\sqrt{N}} \right) \mathbb{E} \left\| \nabla_{i_j} f_{i_j}(\hat{x}_j) \right\|^2,$$

where we set $\alpha = \sqrt{Y/N}$ in the last step. We next consider $\mathbb{E}(T_3)$

$$\begin{aligned} \mathbb{E}(T_3) &= \mathbb{E} \left(\left\| \sum_{j \in J(k)} \gamma_j \left(G_{S_j}(\hat{x}_j; \xi_j) - \frac{N}{Y} \sum_{m \in S_j} \nabla_m f_m(\hat{x}_j) \right) \right\|^2 \right) \\ &= \mathbb{E} \left(\sum_{j \in J(k)} \gamma_j^2 \left\| G_{S_j}(\hat{x}_j; \xi_j) - \frac{N}{Y} \sum_{m \in S_j} \nabla_m f_m(\hat{x}_j) \right\|^2 \right) \\ &= \mathbb{E} \left(\sum_{j \in J(k)} \frac{\gamma_j^2}{Y^2} \left\| Y G_{S_j}(\hat{x}_j; \xi_j) - N \sum_{m \in S_j} \nabla_m f_m(\hat{x}_j) \right\|^2 \right) \\ &= \mathbb{E} \left(\sum_{j \in J(k)} \frac{\gamma_j^2}{Y} \left\| G_{i_j}(\hat{x}_j; \xi_j) - N \nabla_{i_j} f_{i_j}(\hat{x}_j) \right\|^2 \right), \end{aligned} \quad (38)$$

where the last equality is due to the same reason of (32) and the second last equality is from

$$\begin{aligned} &2\mathbb{E} \left(\sum_{j > j'; j, j' \in J(k)} \gamma_j \gamma_{j'} \langle \Gamma_j, \Gamma_{j'} \rangle \right) \\ &= 2\mathbb{E} \left(\sum_{j > j'; j, j' \in J(k)} \gamma_j \gamma_{j'} \left\langle \mathbb{E}_{\xi_j} G_{S_j}(\hat{x}_j; \xi_j) - \frac{N}{Y} \sum_{m \in S_j} \nabla_m f_m(\hat{x}_j), \Gamma_{j'} \right\rangle \right) \\ &= 0, \end{aligned}$$

where

$$\begin{aligned} \Gamma_j &:= G_{S_j}(\hat{x}_j; \xi_j) - \frac{N}{Y} \sum_{m \in S_j} \nabla_m f_m(\hat{x}_j) \\ \Gamma_{j'} &:= G_{S_{j'}}(\hat{x}_{j'}; \xi_{j'}) - \frac{N}{Y} \sum_{m' \in S_{j'}} \nabla_{m'} f_{m'}(\hat{x}_{j'}). \end{aligned}$$

Note that we have the following inequality

$$\begin{aligned} &\mathbb{E}_{\xi, i} \|G_i(x; \xi) - N \nabla_{i_j} f_{i_j}(x)\|^2 \\ &= \mathbb{E}_{\xi, i} \|G_i(x; \xi) - N \nabla_i F(x; \xi) + N \nabla_i F(x; \xi) - N \nabla_i f(x) + N \nabla_i f(x) - N \nabla_{i_j} f_{i_j}(x)\|^2 \\ &\leq 3\mathbb{E}_{\xi, i} (\|G_i(x; \xi) - N \nabla_i F(x; \xi)\|^2 + N^2 \|\nabla_i F(x; \xi) - \nabla_i f(x)\|^2 + N^2 \|\nabla_i f(x) - \nabla_{i_j} f_{i_j}(x)\|^2) \\ &\leq 3\mathbb{E}_{\xi} \left(\frac{N^2 \omega}{2} + N \|\nabla F(x; \xi) - \nabla f(x)\|^2 \right) \\ &\stackrel{(26)}{\leq} \frac{3N^2 \omega}{2} + 3N\sigma^2, \end{aligned}$$

where the first step uses $\|a + b + c\|^2 \leq 3(\|a\|^2 + \|b\|^2 + \|c\|^2)$. Applying it into (38) yields:

$$\mathbb{E}(T_3) \leq \left(\frac{3N^2\omega}{2Y} + \frac{3N\sigma^2}{Y} \right) \sum_{j \in J(k)} \gamma_j^2,$$

Applying the upper bounds of $\mathbb{E}(T_3)$ and $\mathbb{E}(T_4)$ to (37) we obtain

$$\begin{aligned} \mathbb{E}(T_2) &\leq 2\mathbb{E}(T_3) + 2\frac{N^2}{Y^2}\mathbb{E}(T_4) \\ &\leq 2\left(\frac{3N^2\omega}{2Y} + \frac{3N\sigma^2}{Y} \right) \sum_{j \in J(k)} \gamma_j^2 \\ &\quad + 2\frac{N^2}{Y^2} \sum_{j \in J(k)} \gamma_j \left(\gamma_j Y + \frac{Y^{3/2} \sum_{j' \in J(k) \setminus \{j\}} \gamma_{j'}}{\sqrt{N}} \right) \mathbb{E} \|\nabla_{i_j} f_{i_j}(\hat{x}_j)\|^2. \end{aligned}$$

We apply the upper bound of $\mathbb{E}(T_2)$ to (36):

$$\begin{aligned} \mathbb{E}(T_1) &\leq \frac{2}{N} L_T^2 \mathbb{E}(T_2) + \frac{\omega}{2} \\ &\leq \frac{4L_T^2}{Y} \left(\frac{3N\omega}{2} + 3\sigma^2 \right) \sum_{j \in J(k)} \gamma_j^2 + \frac{\omega}{2} \\ &\quad + 4L_T^2 \frac{N}{Y^2} \sum_{j \in J(k)} \gamma_j \left(\gamma_j Y + \frac{Y^{3/2} \sum_{j' \in J(k) \setminus \{j\}} \gamma_{j'}}{\sqrt{N}} \right) \mathbb{E} \|\nabla_{i_j} f_{i_j}(\hat{x}_j)\|^2. \end{aligned}$$

Substitute the upper bound of $\mathbb{E}(T_1)$ into (35)

$$\begin{aligned} &\mathbb{E}(f(x_{k+1}) - f(x_k)) \\ &\leq -\frac{\gamma_k}{2} \left(\mathbb{E} \|\nabla f(x_k)\|^2 + N \mathbb{E} |\nabla_{i_k} f_{i_k}(\hat{x}_k)|^2 \right) \\ &\quad + \frac{\gamma_k}{2} N \mathbb{E}(T_1) + \gamma_k^2 \frac{L_Y}{Y} N (\sigma^2 + \mathbb{E} \|\nabla f(\hat{x}_k)\|^2) + \gamma_k^2 \frac{L_Y}{Y} N^2 \frac{\omega}{4} \\ &\leq -\frac{\gamma_k}{2} \left(\mathbb{E} \|\nabla f(x_k)\|^2 + N \mathbb{E} \|\nabla_{i_k} f_{i_k}(\hat{x}_k)\|^2 \right) \\ &\quad + \frac{\gamma_k}{2} N \left(4L_T^2 \frac{\left(\frac{3N\omega}{2} + 3\sigma^2 \right)}{Y} \sum_{j \in J(k)} \gamma_j^2 + \frac{\omega}{2} \right) \\ &\quad + 2\gamma_k \left(L_T^2 \frac{N^2}{Y^2} \sum_{j \in J(k)} \gamma_j \left(\gamma_j Y + \frac{Y^{3/2} \sum_{j' \in J(k) \setminus \{j\}} \gamma_{j'}}{\sqrt{N}} \right) \mathbb{E} \|\nabla_{i_j} f_{i_j}(\hat{x}_j)\|^2 \right) \\ &\quad + \gamma_k^2 \frac{L_Y}{Y} N (\sigma^2 + \mathbb{E} \|\nabla f(\hat{x}_k)\|^2) + \gamma_k^2 \frac{L_Y}{Y} N^2 \frac{\omega}{4} \\ \stackrel{\text{rearrange}}{=} &-\frac{\gamma_k}{2} \mathbb{E} \|\nabla f(x_k)\|^2 + \gamma_k^2 \frac{L_Y}{Y} N^2 \underbrace{\frac{1}{N} \mathbb{E} \|\nabla f(\hat{x}_k)\|^2}_{=: T_5} \end{aligned}$$

$$\begin{aligned}
& - \left(\frac{\gamma_k N}{2} \right) \mathbb{E} \left\| \nabla_{i_k} f_{i_k}(\hat{x}_k) \right\|^2 \\
& + 2\gamma_k \left(L_T^2 \frac{N^2}{Y^2} \sum_{j \in J(k)} \gamma_j \left(\gamma_j Y + \frac{Y^{3/2} \sum_{j' \in J(k) \setminus \{j\}} \gamma_{j'}}{\sqrt{N}} \right) \mathbb{E} \left\| \nabla_{i_j} f_{i_j}(\hat{x}_j) \right\|^2 \right) \\
& + \frac{\gamma_k}{2} N \left(\frac{4L_T^2}{Y} \left(\frac{3N\omega}{2} + 3\sigma^2 \right) \sum_{j \in J(k)} \gamma_j^2 + \frac{\omega}{2} \right) \\
& + \gamma_k^2 \frac{L_Y}{Y} N \sigma^2 + \gamma_k^2 \frac{L_Y}{Y} N^2 \frac{\omega}{4}. \tag{39}
\end{aligned}$$

Note that

$$\begin{aligned}
T_5 &= \frac{1}{N} \mathbb{E} \left\| \nabla f(\hat{x}_k) \right\|^2 \\
&= \mathbb{E} \left\| \nabla_{i_k} f(\hat{x}_k) \right\|^2 \\
&\stackrel{(28)}{\leq} 2\mathbb{E} \left\| \nabla_{i_k} f_{i_k}(\hat{x}_k) \right\|^2 + 2\mathbb{E} \left\| \nabla_{i_k} f(\hat{x}_k) - \nabla_{i_k} f_{i_k}(\hat{x}_k) \right\|^2 \\
&\stackrel{(26)}{\leq} 2\mathbb{E} \left\| \nabla_{i_k} f_{i_k}(\hat{x}_k) \right\|^2 + \frac{\omega}{2}.
\end{aligned}$$

Substitute this upper bound of T_5 into (39):

$$\begin{aligned}
& \mathbb{E}(f(x_{k+1}) - f(x_k)) \\
\leq & -\frac{\gamma_k}{2} \mathbb{E} \left\| \nabla f(x_k) \right\|^2 + \gamma_k^2 \frac{L_Y}{2Y} N^2 \omega \\
& - \left(\frac{\gamma_k N}{2} - 2\gamma_k^2 \frac{L_Y}{Y} N^2 \right) \mathbb{E} \left\| \nabla_{i_k} f_{i_k}(\hat{x}_k) \right\|^2 \\
& + 2\gamma_k \left(L_T^2 \frac{N^2}{Y^2} \sum_{j \in J(k)} \gamma_j \left(\gamma_j Y + \frac{Y^{3/2} \sum_{j' \in J(k) \setminus \{j\}} \gamma_{j'}}{\sqrt{N}} \right) \mathbb{E} \left\| \nabla_{i_j} f_{i_j}(\hat{x}_j) \right\|^2 \right) \\
& + \frac{\gamma_k}{2} N \left(\frac{4L_T^2}{Y} \left(\frac{3N\omega}{2} + 3\sigma^2 \right) \sum_{j \in J(k)} \gamma_j^2 + \frac{\omega}{2} \right) \\
& + \gamma_k^2 \frac{L_Y}{Y} N \sigma^2 + \gamma_k^2 \frac{L_Y}{Y} N^2 \frac{\omega}{4} \\
\leq & -\frac{\gamma_k}{2} \mathbb{E} \left\| \nabla f(x_k) \right\|^2 - \left(\frac{\gamma_k N}{2} - 2\gamma_k^2 \frac{L_Y}{Y} N^2 \right) \mathbb{E} \left\| \nabla_{i_k} f_{i_k}(\hat{x}_k) \right\|^2 \\
& + 2\gamma_k \left(L_T^2 \frac{N^2}{Y^2} \sum_{j \in J(k)} \gamma_j \left(\gamma_j Y + \frac{Y^{3/2} \sum_{j' \in J(k) \setminus \{j\}} \gamma_{j'}}{\sqrt{N}} \right) \mathbb{E} \left\| \nabla_{i_j} f_{i_j}(\hat{x}_j) \right\|^2 \right) \\
& + 2\gamma_k N \frac{L_T^2}{Y} \left(\frac{3N\omega}{2} + 3\sigma^2 \right) \sum_{j \in J(k)} \gamma_j^2 \\
& + \gamma_k^2 \frac{L_Y}{Y} N \sigma^2 + \gamma_k^2 \frac{L_Y}{Y} N^2 \omega + \frac{\gamma_k}{4} N \omega. \tag{40}
\end{aligned}$$

Summarizing (40) from $k = 0$ to $k = K$ (note that Θ_k is defined in (27)) yields:

$$\begin{aligned}
& \frac{1}{2} \sum_{k=0}^K \gamma_k \mathbb{E} \|\nabla f(x_k)\|^2 \\
\leq & f(x_0) - f(x_{K+1}) - \sum_{k=0}^K \Theta_k \mathbb{E} \|\nabla_{i_k} f_{i_k}(\hat{x}_k)\|^2 \\
& + 2N \frac{L_T^2}{Y} \left(\frac{3N\omega}{2} + 3\sigma^2 \right) \sum_{k=0}^K \left(\gamma_k \sum_{j \in J(k)} \gamma_j^2 \right) \\
& + \left(\frac{L_Y}{Y} N\sigma^2 + \frac{L_Y}{Y} N^2\omega \right) \sum_{k=0}^K \gamma_k^2 + \frac{1}{4} N\omega \sum_{k=0}^K \gamma_k \\
\stackrel{\Theta_k \geq 0}{\leq} & f(x_0) - f^* + 2N \frac{L_T^2}{Y} \left(\frac{3N\omega}{2} + 3\sigma^2 \right) \sum_{k=0}^K \left(\gamma_k \sum_{j \in J(k)} \gamma_j^2 \right) \\
& + \left(\frac{L_Y}{Y} N\sigma^2 + \frac{L_Y}{Y} N^2\omega \right) \sum_{k=0}^K \gamma_k^2 + \frac{1}{4} N\omega \sum_{k=0}^K \gamma_k.
\end{aligned}$$

It completes the proof. \square

Corollary 8. Set all steplength γ_k to be a constant γ in Algorithm 1

$$\gamma = \frac{Y}{N} \frac{1}{2L_Y\chi'} \quad (41)$$

where χ satisfies

$$\chi \geq \sqrt{1 + \frac{L_T^2}{L_Y^2} \left(\frac{Y}{N} + \frac{Y^{3/2}T}{N^{3/2}} \right) T} + 1. \quad (42)$$

It ensures the following convergence rate

$$\frac{1}{2K} \sum_{k=0}^K \mathbb{E} \|\nabla f(x_k)\|^2 \leq \frac{2(f(x_0) - f^*)L_Y N}{KY} \chi + \frac{N\omega + \sigma^2}{\chi} \left(1 + \frac{2L_T^2 Y T}{L_Y^2 N} \frac{1}{\chi} \right) + N\omega.$$

Proof. To apply Theorem 7, we first verify that the choice of γ in (41) satisfies the prerequisite (27).

Letting $\gamma_k = \gamma$, the prerequisite (27) in Theorem 7 reduces to

$$\frac{N\gamma}{2} - 2\gamma^2 \frac{L_Y}{Y} N^2 - 2L_T^2 \frac{N^2}{Y^2} \gamma \left(\gamma Y + \frac{Y^{3/2}T\gamma}{\sqrt{N}} \right) T \gamma \geq 0, \forall k,$$

or equivalently

$$\underbrace{\frac{L_T^2}{Y^2} \gamma^2 \left(Y + \frac{Y^{3/2}T}{\sqrt{N}} \right) T}_{=:l} + \gamma \frac{L_Y}{Y} - \frac{1}{4N} \leq 0, \forall k.$$

Here we denote $\left(Y + \frac{Y^{3/2}T}{\sqrt{N}}\right) T$ by l for short here.

To satisfy the above inequality, it suffices to show

$$\begin{aligned}
\gamma &\leq \frac{-L_Y/Y + \sqrt{L_Y^2/Y^2 + (L_T^2/Y^2)l/N}}{2L_T^2l/Y^2} \\
&= Y \frac{-L_Y + \sqrt{L_Y^2 + L_T^2l/N}}{2L_T^2l} \\
&= Y \frac{L_Y}{2L_T^2l} \left(\sqrt{1 + \frac{L_T^2l}{L_Y^2N}} - 1 \right) \\
&= Y \frac{1}{2L_YN} \frac{\sqrt{1 + \chi'} - 1}{\chi'} \\
&= Y \frac{1}{2L_YN} \frac{1}{\chi_0},
\end{aligned}$$

where

$$\begin{aligned}
\chi' &:= \frac{L_T^2l}{L_Y^2N} = \frac{L_T^2 \left(Y + \frac{Y^{3/2}T}{\sqrt{N}}\right) T}{L_Y^2N}, \\
\chi_0 &:= \sqrt{1 + \chi'} + 1 = \sqrt{1 + \frac{L_T^2 \left(Y + \frac{Y^{3/2}T}{\sqrt{N}}\right) T}{L_Y^2N}} + 1.
\end{aligned}$$

Due to $\chi \geq \chi_0$, the choice of γ in (27) satisfies the prerequisite (27).

Now by applying Theorem 7

$$\begin{aligned}
\frac{1}{2} \sum_{k=0}^K \gamma_k \mathbb{E} \|\nabla f(x_k)\|^2 &\leq f(x_0) - f^* + 2N \frac{L_T^2}{Y} \left(\frac{3N\omega}{2} + 3\sigma^2 \right) \sum_{k=0}^K \left(\gamma_k \sum_{j \in J(k)} \gamma_j^2 \right) \\
&\quad + \left(\frac{L_Y}{Y} N\sigma^2 + \frac{L_Y}{Y} N^2\omega \right) \sum_{k=0}^K \gamma_k^2 + \frac{1}{4} N\omega \sum_{k=0}^K \gamma_k.
\end{aligned}$$

and letting $\gamma_k = \gamma$ and dividing both sides by $K\gamma$, we obtain

$$\begin{aligned}
\frac{1}{2K} \sum_{k=0}^K \mathbb{E} \|\nabla f(x_k)\|^2 &\leq \frac{f(x_0) - f^*}{K\gamma} + 2N \frac{L_T^2}{Y} \left(\frac{3N\omega}{2} + 3\sigma^2 \right) T\gamma^2 \\
&\quad + \left(\frac{L_Y}{Y} N\sigma^2 + \frac{L_Y}{Y} N^2\omega \right) \gamma + \frac{1}{4} N\omega.
\end{aligned} \tag{43}$$

Substituting γ into (43), we have

$$\frac{1}{2K} \sum_{k=0}^K \mathbb{E} \|\nabla f(x_k)\|^2 \leq \frac{f(x_0) - f^*}{K\gamma} + 2N \frac{L_T^2}{Y} \left(\frac{3N\omega}{2} + 3\sigma^2 \right) T\gamma^2$$

$$\begin{aligned}
& + \left(\frac{L_Y}{Y} N \sigma^2 + \frac{L_Y}{Y} N^2 \omega \right) \gamma + \frac{1}{4} N \omega \\
= & \frac{2(f(x_0) - f^*) L_Y N}{KY} \chi \\
& + \frac{L_T^2 Y T}{2L_Y^2 N} \left(\frac{3N\omega}{2} + 3\sigma^2 \right) \frac{1}{\chi^2} + (\sigma^2 + N\omega) \frac{1}{2\chi} + \frac{1}{4} N \omega \\
\leq & \frac{2(f(x_0) - f^*) L_Y N}{KY} \chi \\
& + \frac{2L_T^2 Y T}{L_Y^2 N} (N\omega + \sigma^2) \frac{1}{\chi^2} + (\sigma^2 + N\omega) \frac{1}{\chi} + N\omega \\
= & \frac{2(f(x_0) - f^*) L_Y N}{KY} \chi + \frac{N\omega + \sigma^2}{\chi} \left(1 + \frac{2L_T^2 Y T}{L_Y^2 N} \frac{1}{\chi} \right) + N\omega,
\end{aligned}$$

which completing the proof. \square

Theorem 9. Set all steplength γ_k to be a constant γ in Algorithm 1

$$\gamma = \frac{Y}{N} \frac{1}{2L_Y \chi},$$

where

$$\chi = \sqrt{\frac{\alpha_1^2}{K(N\omega + \sigma^2)\alpha_2 + \alpha_1}} + \sqrt{K(N\omega + \sigma^2)\alpha_2}. \quad (44)$$

It ensures the following convergence rate

$$\frac{\sum_{k=0}^K \mathbb{E} \|\nabla f(x_k)\|^2}{2K} \leq \frac{2\alpha_1}{K\alpha_2 \sqrt{K(N\omega + \sigma^2)\alpha_2 + \alpha_1}} + \frac{3\sqrt{N\omega + \sigma^2}}{\sqrt{K}\alpha_2} + \frac{2L_T^2 Y T}{L_Y^2 N K \alpha_2} + N\omega.$$

Proof. In order to apply Corollary 8, we first verify that the choice of χ in (44) satisfies the requirement in (42). (44) suggests that

$$\chi^2 \geq \frac{\alpha_1^2}{K(N\omega + \sigma^2)\alpha_2 + \alpha_1} + K(N\omega + \sigma^2)\alpha_2 \geq \alpha_1.$$

where the second inequality is due to that $K = 0$ minimize the second part. Also note that

$$\begin{aligned}
\alpha_1 & = 4 \left(1 + \frac{L_T^2 \left(Y + \frac{Y^{3/2} T}{\sqrt{N}} \right) T}{L_Y^2 N} \right) \\
& = \left(2 \sqrt{1 + \frac{L_T^2}{L_Y^2} \left(\frac{Y}{N} + \frac{Y^{3/2} T}{N^{3/2}} \right) T} \right)^2 \\
& \geq \left(\sqrt{1 + \frac{L_T^2}{L_Y^2} \left(\frac{Y}{N} + \frac{Y^{3/2} T}{N^{3/2}} \right) T} + 1 \right)^2.
\end{aligned}$$

Therefore the choice of χ in (44) satisfies the condition in (42) required by Corollary 8. Applying Corollary 8 yields the following convergence rate

$$\begin{aligned}
& \frac{1}{2K} \sum_{k=0}^K \mathbb{E} \|\nabla f(x_k)\|^2 \\
& \leq \frac{2(f(x_0) - f^*)L_Y N}{KY} \chi + \frac{N\omega + \sigma^2}{\chi} \left(1 + \frac{2L_T^2 Y T}{L_Y^2 N} \frac{1}{\chi} \right) + N\omega \\
(44) \quad & \stackrel{=}{=} \frac{2(f(x_0) - f^*)L_Y N}{KY} \sqrt{\frac{\alpha_1^2}{K(N\omega + \sigma^2)\alpha_2 + \alpha_1}} \\
& + \frac{2(f(x_0) - f^*)L_Y N}{\sqrt{KY}} \sqrt{(N\omega + \sigma^2)\alpha_2} \\
& + \frac{N\omega + \sigma^2}{\underbrace{\sqrt{\frac{\alpha_1^2}{K(N\omega + \sigma^2)\alpha_2 + \alpha_1}} + \sqrt{K(N\omega + \sigma^2)\alpha_2}}_{\text{discard}}} \\
& + \frac{2L_T^2 Y T}{L_Y^2 N} \frac{N\omega + \sigma^2}{\left(\underbrace{\sqrt{\frac{\alpha_1^2}{K(N\omega + \sigma^2)\alpha_2 + \alpha_1}} + \sqrt{K(N\omega + \sigma^2)\alpha_2}}_{\text{discard}} \right)^2} \\
& + N\omega \\
& \leq \frac{2(f(x_0) - f^*)L_Y N \alpha_1}{KY \sqrt{K(N\omega + \sigma^2)\alpha_2 + \alpha_1}} \\
& + \frac{2(f(x_0) - f^*)L_Y N \sqrt{(N\omega + \sigma^2)\alpha_2}}{\sqrt{KY}} \\
& + \frac{\sqrt{N\omega + \sigma^2}}{\sqrt{K\alpha_2}} + \frac{2L_T^2 Y T}{L_Y^2 N K \alpha_2} + N\omega \\
& = \frac{2\alpha_1}{K\alpha_2 \sqrt{K(N\omega + \sigma^2)\alpha_2 + \alpha_1}} + \frac{3\sqrt{N\omega + \sigma^2}}{\sqrt{K\alpha_2}} + \frac{2L_T^2 Y T}{L_Y^2 N K \alpha_2} + N\omega.
\end{aligned}$$

It completes the proof. □

Proof to Theorem 1

Proof. Note that in Theorem 1 we have the same steplength as in Theorem 9, so we can safely apply Theorem 9 to obtain

$$\frac{\sum_{k=0}^K \mathbb{E} \|\nabla f(x_k)\|^2}{2K}$$

$$\begin{aligned}
&\leq \frac{2\alpha_1}{K\alpha_2\sqrt{K(N\omega + \sigma^2)\alpha_2 + \alpha_1}} + \frac{3\sqrt{N\omega + \sigma^2}}{\sqrt{K\alpha_2}} + \frac{2L_T^2YT}{L_Y^2NK\alpha_2} + N\omega \\
&\stackrel{(8)}{=} \frac{8\left(1 + \frac{L_T^2T}{L_Y^2N}\left(Y + \frac{Y^{3/2}T}{\sqrt{N}}\right)\right)}{K\alpha_2\sqrt{K(N\omega + \sigma^2)\alpha_2 + 4 + \underbrace{\frac{4L_T^2T}{L_Y^2N}\left(Y + \frac{Y^{3/2}T}{\sqrt{N}}\right)}_{\text{discard}}}} \\
&\quad + \frac{3\sqrt{N\omega + \sigma^2}}{\sqrt{K\alpha_2}} + \frac{2L_T^2YT}{L_Y^2NK\alpha_2} + N\omega \\
&\leq \frac{8}{K\alpha_2\sqrt{K(N\omega + \sigma^2)\alpha_2 + 4}} + \frac{8\frac{L_T^2T}{L_Y^2N}\left(Y + \frac{Y^{3/2}T}{\sqrt{N}}\right)}{K\alpha_2\sqrt{K(N\omega + \sigma^2)\alpha_2 + 4}} \\
&\quad + \frac{3\sqrt{N\omega + \sigma^2}}{\sqrt{K\alpha_2}} + \frac{2L_T^2YT}{L_Y^2NK\alpha_2} + N\omega. \tag{45}
\end{aligned}$$

Next from the condition of T in Theorem 1 and the definition of α_3 , we can obtain

$$\frac{L_T^2\left(Y + \frac{Y^{3/2}T}{\sqrt{N}}\right)T}{L_Y^2N} \leq K(N\omega + \sigma^2)\alpha_2 + 4 = \alpha_3\frac{L_T^2}{L_Y^2}. \tag{46}$$

To see why it is true, it suffices to show that

$$\begin{aligned}
T &\leq \frac{-Y\sqrt{N} + \sqrt{NY^2 + 4Y^{3/2}N^{3/2}\alpha_3}}{2Y^{3/2}} \\
&= \frac{-\sqrt{N} + \sqrt{N + 4Y^{-1/2}N^{3/2}\alpha_3}}{2Y^{1/2}} \\
&= \frac{\sqrt{N}}{2Y^{1/2}} \left(\sqrt{1 + 4Y^{-1/2}N^{1/2}\alpha_3} - 1 \right), \tag{47}
\end{aligned}$$

which is implied by the prerequisite for T in Theorem 1.

Then we apply (46) to (45) and obtain

$$\begin{aligned}
&\frac{\sum_{k=0}^K \mathbb{E} \|\nabla f(x_k)\|^2}{2K} \\
&\leq \frac{8}{K\alpha_2\sqrt{K(N\omega + \sigma^2)\alpha_2 + 4}} + \frac{8\frac{L_T^2\left(Y + \frac{Y^{3/2}T}{\sqrt{N}}\right)T}{L_Y^2N}}{K\alpha_2\sqrt{K(N\omega + \sigma^2)\alpha_2 + 4}} \\
&\quad + \frac{2L_T^2YT}{L_Y^2NK\alpha_2} + \frac{3\sqrt{N\omega + \sigma^2}}{\sqrt{K\alpha_2}} + N\omega \\
&\stackrel{(47)}{\leq} \frac{8}{K\alpha_2\sqrt{K(N\omega + \sigma^2)\alpha_2 + 4}} + \frac{8\sqrt{K(N\omega + \sigma^2)\alpha_2 + 4}}{K\alpha_2} \\
&\quad + \frac{L_T^2\sqrt{NY}\left(\sqrt{1 + 4Y^{-1/2}N^{1/2}\alpha_3} - 1\right)}{L_Y^2NK\alpha_2} + \frac{3\sqrt{N\omega + \sigma^2}}{\sqrt{K\alpha_2}} + N\omega
\end{aligned}$$

$$\begin{aligned}
&\leq \frac{8}{K\alpha_2\sqrt{K(N\omega + \sigma^2)\alpha_2 + 4}} + \frac{8\sqrt{K(N\omega + \sigma^2)\alpha_2}}{K\alpha_2} + \frac{16}{K\alpha_2} + \frac{3\sqrt{N\omega + \sigma^2}}{\sqrt{K\alpha_2}} + N\omega \\
&\quad + \frac{1}{K\alpha_2} \frac{L_T^2}{L_Y^2} \frac{\sqrt{Y} \left(\sqrt{1 + 4Y^{-1/2}N^{1/2}\alpha_3} - 1 \right)}{\sqrt{N}} \\
&= \frac{1}{K\alpha_2} \left(16 + \frac{L_T^2}{L_Y^2} \frac{\sqrt{Y} \left(\sqrt{1 + 4Y^{-1/2}N^{1/2}\alpha_3} - 1 \right)}{\sqrt{N}} + \overbrace{\frac{8}{\sqrt{K(N\omega + \sigma^2)\alpha_2 + 4}}}^{\leq 4} \right) \\
&\quad + \frac{11\sqrt{N\omega + \sigma^2}}{\sqrt{K\alpha_2}} + N\omega \\
&\leq \frac{1}{K\alpha_2} \left(20 + \frac{L_T^2}{L_Y^2} \frac{\sqrt{Y} \left(\sqrt{1 + 4Y^{-1/2}N^{1/2}\alpha_3} - 1 \right)}{\sqrt{N}} \right) + \frac{11\sqrt{N\omega + \sigma^2}}{\sqrt{K\alpha_2}} + N\omega \\
&= \frac{20}{K\alpha_2} + \frac{1}{K\alpha_2} \left(\frac{L_T^2}{L_Y^2} \frac{\sqrt{Y} \left(\sqrt{1 + 4Y^{-1/2}N^{1/2}\alpha_3} - 1 \right)}{\sqrt{N}} + 11\sqrt{N\omega + \sigma^2}\sqrt{K\alpha_2} \right) + N\omega.
\end{aligned}$$

It completes the proof. \square

A.3 Proofs to Corollaries

We prove all corollaries using Theorem 1 in this subsection.

Proof to Corollary 2

Proof. For ASCD, letting $\sigma = 0$, $\omega = 0$, and $Y = 1$ in Theorem 1, we have

$$\begin{aligned}
\gamma^{-1} &= 2L_{\max}N \left(\sqrt{\frac{\alpha_1^2}{K(N\omega + \sigma^2)\alpha_2 + \alpha_1}} + \sqrt{K(N\omega + \sigma^2)\alpha_2} \right) \\
&= 2L_{\max}N\sqrt{\alpha_1},
\end{aligned}$$

and the prerequisite becomes

$$\begin{aligned}
T &\leq \frac{\sqrt{N}}{2} \left(\sqrt{1 + 4\alpha_3N^{1/2}} - 1 \right) \\
&= \frac{\sqrt{N}}{2} \left(\sqrt{1 + 4\frac{L_Y^2}{L_T^2} (K(N\omega + \sigma^2)\alpha_2 + 4)N^{1/2}} - 1 \right) \\
&= \frac{\sqrt{N}}{2} \left(\sqrt{1 + 16\frac{L_{\max}^2}{L_T^2}N^{1/2}} - 1 \right). \\
&= O(N^{3/4})
\end{aligned}$$

The convergence rate turns out to be

$$\begin{aligned}
& \frac{\sum_{k=0}^K \mathbb{E} \|\nabla f(x_k)\|^2}{2K} \\
& \leq \frac{1}{K\alpha_2} \left(20 + \frac{L_T^2}{L_{\max}^2} \frac{\sqrt{1 + 16N^{1/2} \frac{L_{\max}^2}{L_T^2} - 1}}{\sqrt{N}} \right) \\
& = \frac{(f(x_0) - f^*)L_{\max}N}{K} \left(20 + \frac{L_T^2}{L_{\max}^2} \frac{\sqrt{1 + 16N^{1/2} \frac{L_{\max}^2}{L_T^2} - 1}}{\sqrt{N}} \right).
\end{aligned}$$

It completes the proof. \square

Proof to Corollary 3

Proof. For ASGD in (12), letting $\omega = 0$ and $Y = N$ in α_1, α_2 , and α_3 in (8), we have

$$\begin{aligned}
\alpha_1 &= 4 \left(1 + \frac{L_T^2(1+T)T}{L^2} \right), \\
\alpha_2 &= \frac{1}{(f(x_0) - f^*)L'} \\
\alpha_3 &= \frac{L^2}{L_T^2} (K\sigma^2\alpha_2 + 4).
\end{aligned}$$

Next letting $\omega = 0$ and $Y = N$ in Theorem 1, the prerequisite for T becomes

$$\begin{aligned}
T &\leq \frac{\sqrt{N}}{2\sqrt{N}} \left(\sqrt{1 + 4N^{-1/2}N^{1/2}\alpha_3} - 1 \right) \\
&= \frac{1}{2} \left(\sqrt{1 + 4\alpha_3} - 1 \right) \\
&= \frac{1}{2} \left(\sqrt{1 + 4 \frac{L^2}{L_T^2} \left(\frac{K\sigma^2}{(f(x_0) - f^*)L} + 4 \right)} - 1 \right) \\
&= O \left(\sqrt{K\sigma^2 + 1} \right).
\end{aligned}$$

We finally obtain the following convergence rate

$$\begin{aligned}
& \frac{\sum_{k=0}^K \mathbb{E} \|\nabla f(x_k)\|^2}{2K} \\
& \leq \frac{1}{K\alpha_2} \left(20 + \frac{L_T^2}{L_Y^2} \frac{\sqrt{Y} \left(\sqrt{1 + 4Y^{-1/2}N^{1/2}\alpha_3} - 1 \right)}{\sqrt{N}} \right)
\end{aligned}$$

$$\begin{aligned}
& + \frac{11\sqrt{N\omega + \sigma^2}}{\sqrt{K\alpha_2}} + N\omega \\
= & \frac{1}{K\alpha_2} \left(20 + \frac{L_T^2}{L^2} \sqrt{1 + 4\alpha_3} - 1 \right) + \frac{11\sigma}{\sqrt{K\alpha_2}} \\
\leq & \frac{1}{K\alpha_2} \left(20 + \frac{L_T^2}{L^2} \sqrt{5\alpha_3} \right) + \frac{11\sigma}{\sqrt{K\alpha_2}} \\
\leq & \frac{1}{K\alpha_2} \left(20 + \frac{L_T^2}{L^2} \sqrt{5\alpha_3} \right) + \frac{11\sigma}{\sqrt{K\alpha_2}} \\
= & \frac{1}{K\alpha_2} \left(20 + \frac{L_T^2}{L^2} \sqrt{5 \frac{L^2}{L_T^2} (K\sigma^2\alpha_2 + 4)} \right) + \frac{11\sigma}{\sqrt{K\alpha_2}} \\
= & \frac{1}{K\alpha_2} \left(20 + \frac{L_T\sqrt{5}}{L} \sqrt{K\sigma^2\alpha_2 + 4} \right) + \frac{11\sigma}{\sqrt{K\alpha_2}} \\
\leq & \frac{1}{K\alpha_2} \left(20 + \frac{L_T\sqrt{5}}{L} \left(\sqrt{K\sigma^2\alpha_2 + 2} \right) \right) + \frac{11\sigma}{\sqrt{K\alpha_2}} \\
= & O\left(\frac{1}{K} + \frac{\sigma}{\sqrt{K}}\right),
\end{aligned}$$

It completes the proof. □

Proof to Corollary 4 and Corollary 5

Proof. Corollary 4 can be considered a special case of Corollary 5 with $\omega = 0$. We only prove Corollary 5 here, which automatically implies Corollary 4. Letting $Y = 1$ in (8) and Theorem 1, we obtain

$$\gamma^{-1} = 2L_{\max}N \left(\sqrt{\frac{\alpha_1^2}{K(N\omega + \sigma^2)\alpha_2 + \alpha_1}} + \sqrt{K(N\omega + \sigma^2)\alpha_2} \right),$$

and

$$\begin{aligned}
\alpha_1 &= 4 \left(1 + \frac{L_T^2 \left(1 + \frac{T}{\sqrt{N}} \right) T}{L_{\max}^2 N} \right), \\
\alpha_2 &= \frac{1}{(f(x_0) - f^*)L_{\max}N'}, \\
\alpha_3 &= \frac{L_{\max}^2}{L_T^2} (K(N\omega + \sigma^2)\alpha_2 + 4), \\
\omega &= \frac{\sum_{i=1}^N L_{(i)}^2 \mu_i^2}{N}.
\end{aligned}$$

The prerequisite for T in Theorem 1 becomes

$$\begin{aligned}
T &\leq \frac{\sqrt{N}}{2} \left(\sqrt{1 + 4\alpha_3 N^{1/2}} - 1 \right) \\
&= \frac{\sqrt{N}}{2} \left(\sqrt{1 + 4 \frac{L_{\max}^2}{L_T^2} (K(N\omega + \sigma^2)\alpha_2 + 4) N^{1/2}} - 1 \right) \\
&= O \left(N^{3/4} \sqrt{K \left(\omega + \frac{\sigma^2}{N} \right) + 1} \right) \\
&= O \left(\sqrt{N^{3/2} + KN^{1/2}\sigma^2} \right).
\end{aligned} \tag{48}$$

The convergence rate becomes

$$\begin{aligned}
&\frac{\sum_{k=0}^K \mathbb{E} \|\nabla f(x_k)\|^2}{2K} \\
&\leq \frac{1}{K\alpha_2} \left(20 + \frac{L_T^2}{L_{\max}^2} \frac{\sqrt{1 + 4N^{1/2}\alpha_3} - 1}{\sqrt{N}} \right) + \frac{11\sqrt{N\omega + \sigma^2}}{\sqrt{K\alpha_2}} + N\omega \\
&= O \left(\frac{N}{K} + \frac{N^{3/4}}{\sqrt{K}} \sqrt{\omega + \frac{\sigma^2}{N} + \frac{1}{K}} + \frac{N\sqrt{\omega + \frac{\sigma^2}{N}}}{\sqrt{K}} + N\omega \right) \\
&\leq O \left(\frac{N}{K} + \frac{N^{3/4}}{\sqrt{K}} \left(\sqrt{\omega + \frac{\sigma^2}{N} + \frac{1}{K}} \right) + \frac{N\sqrt{\omega + \frac{\sigma^2}{N}}}{\sqrt{K}} + N\omega \right) \\
&= O \left(\frac{N}{K} + \frac{\sqrt{N}\sqrt{N\omega + \sigma^2}}{\sqrt{K}} + N\omega \right) \\
&\leq O \left(\frac{N}{K} + \frac{\sqrt{N}\sigma}{\sqrt{K}} + \frac{N\sqrt{\omega}}{\sqrt{K}} + N\omega \right).
\end{aligned}$$

Since $\omega = \frac{\sum_{i=1}^N L_{(i)}^2 \mu_i^2}{N}$, if we use a constant μ for all μ_i , and if we let $N\omega \leq O(N/K)$, $\frac{N\sqrt{\omega}}{\sqrt{K}} \leq O(N/K)$, we need

$$\mu \leq \sqrt{\frac{N}{K \sum_{i=1}^N L_{(i)}^2}} = O \left(\frac{1}{\sqrt{K}} \right). \tag{49}$$

If let $N\omega \leq O(\frac{\sqrt{N}\sigma}{\sqrt{K}})$, $\frac{N\sqrt{\omega}}{\sqrt{K}} \leq O(\frac{\sqrt{N}\sigma}{\sqrt{K}})$, it suffices that

$$\mu \leq O \left(\min \left\{ \frac{\sqrt{\sigma}}{(K)^{1/4} (N)^{1/4}}, \sigma / \sqrt{N} \right\} \right). \tag{50}$$

Since (19) satisfies either (49) or (50), we obtain a convergence rate of

$$\frac{\sum_{k=0}^K \mathbb{E} \|\nabla f(x_k)\|^2}{2K} \leq O\left(\frac{N}{K} + \frac{\sqrt{N}\sigma}{\sqrt{K}}\right).$$

The prerequisite (48) becomes

$$\begin{aligned} T &\leq O\left(N^{3/4} \sqrt{K\left(\omega + \frac{\sigma^2}{N}\right) + 1}\right) \\ &\leq O\left(N^{3/4} \sqrt{1 + \frac{K\sigma^2}{N}}\right), \end{aligned}$$

which completing the proof. □