On Convergence and Generalization of Semi-Asynchronous SpiderBoost for Distributed Non-Convex Learning

ZHUQING LIU, Department of Electrical and Computer Engineering, The Ohio State University
XIN ZHANG, Department of Statistics, Iowa State University
JIA LIU, Department of Electrical and Computer Engineering, The Ohio State University

To increase the parallel training speed of many distributed machine learning applications, recent years have witnessed a significant amount of interest in developing distributed stochastic variance-reduced techniques to accelerate the conventional stochastic gradient descent algorithm. So far, most of these accelerated approaches are based on synchronous algorithms. That is, there is no timing uncertainty in synchronous systems, and the operations of all processes are executed in synchronous mode. For many applications in practice, however, asynchronous training algorithms could be more attractive due to their low implementation complexity and good scalability. Meanwhile, it is also well-known that due to the extra uncertainty in asynchronous training, the convergence performance of asynchronous training algorithms could become much worse than their synchronous counterparts. This motivates to consider the following questions: *Could we combine and achieve the best of both worlds of synchronous and asynchronous trainings by developing a new semi-asynchronous training algorithm? If yes, what about the training and generalization performances of this new algorithm?* In this paper, we provide the first theoretical analysis on the convergence and generalization performances for the semi-asynchronous version of SpiderBoost – a state-of-the-art stochastic variance-reduced method. We consider two asynchronous parallel implementations of SpiderBoost under two computing architectures, namely the distributed memory and shared memory architectures. Under the distributed and shared memory architectures, we show that the semi-asynchronous SpiderBoost algorithm (SA-SpiderBoost) has $O(\sqrt{N}\epsilon^{-2}(\Delta+1)+N)$ and $O(\sqrt{N}\epsilon^{-2}(\Delta+1)d+N)$ computational complexities for achieving an ϵ-stationary point in non-convex learning, respectively, where N denotes the total number of training samples. Moreover, we investigate the generalization performance of the SA-SpiderBoost algorithm. We establish algorithmic stability bounds for quadratic strongly convex and non-convex optimization under standard Lipschitz and smoothness assumptions. We prove that SA-SpiderBoost generalizes well with sufficiently many data samples and with sufficiently small learning rates. Our results also characterize a fundamental trade-off between convergence and generalization of the asynchronous stochastic first-order variance-reduced methods. We further conduct extensive numerical experiments to verify our theoretical findings.

CCS Concepts: • **Theory of computation → Distributed algorithms**; **Distributed algorithms**; • **Computing methodologies → Machine learning algorithms**.

Additional Key Words and Phrases: Distributed algorithms; Asynchronous algorithm; Algorithm generalization

Authors' addresses: Zhuqing Liu, liu.9384@osu.edu, Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH, 43210; Xin Zhang, xinzhang@iastate.edu, Department of Statistics, Iowa State University, Ames, IA, 50011; Jia Liu, liu@ece.osu.edu, Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH, 43210.

XXXX-XXXX/2021/2-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Zhuqing Liu, Xin Zhang, and Jia Liu. 2021. On Convergence and Generalization of Semi-Asynchronous SpiderBoost for Distributed Non-Convex Learning. 1, 1 (February 2021), 39 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

From its early days as a discipline, machine learning (ML) has made extensive use of optimization formulations and algorithms. Mathematically, the training process of a machine learning task is manifested as solving a large-scale non-convex optimization problem in the form of $\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}, \xi_i)$, where $f(\mathbf{x}, \xi_i)$ represents a loss function that evaluates the discrepancy between a learning model's output and the ground truth of sample ξ_i, and N is the number of samples. Moreover, due to the high dimensionality in d and unstructured non-convexity of many machine learning models (e.g., deep neural networks), the simple first-order stochastic gradient descent (SGD) method has long been used as the default approach for ML training process. However, it is well known that the SGD algorithm suffers from a high computational complexity in solving non-convex optimization problems. Specifically, for non-convex optimization problems, SGD takes an overall complexity of $O(\epsilon^{-4})$ to attain a first-order ϵ-approximate stationary point $\mathbf{x}$ (i.e., $\mathbb{E}[\|\nabla f(\mathbf{x})\|] \leq \epsilon$, where ϵ is some chosen accuracy parameter). With the rapidly growing demands on high-quality machine learning applications, the efficiency of the traditional SGD approach has become increasingly unsatisfactory.

To accelerate the traditional SGD approach, there have two main approaches in the literature. The first approach is to exploit various algorithmic *acceleration* techniques. These include but are not limited to the uses of momentum [21], adaptive learning rates [8], etc. One of the most notable acceleration approaches emerging in recent years is the the family of "variance-reduced" methods (see, e.g., SVRG [13], SAG [26], and SAGA [6] and many of their variants). Generally speaking, the basic idea of these variance-reduced methods is to construct more accurate stochastic gradient estimators than that of SGD by periodically recomputing (near) full gradients (see Section 2 for more detailed discussions). It has been shown that these variance-reduced methods yield an overall computational complexity of $O(N^{2/3}\epsilon^{-2})$ for solving non-convex machine learning problems, where N denotes the sample size. In the variance-reduced algorithm family, the state of the art is the so-called SPIDER (stochastic path-integrated differential estimator) method [7] and its variant SpiderBoost [29], which outperform other existing algorithms in the same family for solving non-convex optimization problems. Both algorithms have been shown to achieve a computational cost of $O(\min\{N^{1/2}\epsilon^{-2}, \epsilon^{-3}\})$ to find an ϵ-approximate first-order stationary point for non-convex optimization problems. Compared to SPIDER, SpiderBoost is an improved version that allows a larger learning rate $\eta = O(1/L)$ without sacrificing the overall complexity, where L is the Lipschitz constant of the objective gradient.

On the other hand, thanks to SGD's simple aggregation-based structure, another major approach to accelerate the traditional SGD method is exploit *distributed and parallel computing* architectures. In particular, with the classic Moore's law (i.e., the number of transistors per unit area will double every 1-2 years) nearing its end between 2020 and 2025, it has been widely accepted that a viable solution for accelerating ML training is to exploit parallelism at both machine and chip scales. Indeed, the rise of machine learning in recent years is largely due to the advancement of multi-core CPU / GPU [1, 5, 16]. Both *distributed memory* parallelism on multiple GPUs [2, 17, 33] and *shared memory* parallelism on a multi-core machine [22, 35] have been widely leveraged to facilitate the training phase in machine learning, such as parallel SVM[28], parallel matrix factorization [27, 32]and distributed deep learning[30], just to name a few.

Conundrum in Distributed Learning: Synchronous or Asynchronous? In distributed and parallel computing, a key design question is the architectural decision choices between *"synchronous" and "asynchronous"* implementations for the distributed algorithm. In the literature, most of the distributed optimization algorithms are synchronous in nature, i.e., a set of computing nodes perform some assigned computational tasks distributively following a *common clock*. A salient feature of synchronous parallel/distributed algorithms is that they are often relatively easier to analyze and understand theoretically. However, from a practical standpoint, synchronous implementation may not always be realistic in practice due to the following reasons. First, in many geographically dispersed distributed computing systems, maintaining a common clock for synchronization is often costly (or even infeasible in some cases), which could result in high implementation complexity and system overhead. Second, synchronous parallel/distributed algorithms usually do not perform well under heterogenous computing environments since all computing nodes must wait for the slowest one (often referred to as the "straggler problem" in the literature) to finish in each iteration. This could introduce a large amount of system idling that significantly affects the overall system utilization. Third, synchronous operations in parallel/distributed algorithms typically lead to *periodic spikes* in data traffic in information exchange, which could in turn induce heavy congestions in the systems. This periodic congestions cause high computational latency or packet losses.

By contrast, in asynchronous parallel/distributed algorithms, rather than making updates simultaneously, each node computes its own solution in each iteration without waiting for other computing nodes in the system. Compared to their synchronous counterparts, asynchronous algorithms are much easier to implement and cause less network traffic congestions and delays, all of which are attractive in practice for solving large-scale optimization problems for machine learning. However, one of the major limitations of asynchronous parallel/distributed algorithms is the use of *stale* system information due to asynchronous updates. If not treated appropriately, such delayed information could destroy the convergence of the asynchronous algorithm. This problem is particularly concerning for the asynchronous SGD-type algorithms, which have become a fundamental building block of many distributed machine learning frameworks in use today (e.g., TensorFlow, MXNet, PyTorch, etc.). Hence, understanding the convergence performance of asynchronous SGD-type algorithms has received growing attention in recent years (see, e.g., [12, 17, 22, 36]). However, due to the significant technical challenges, most existing design and analysis of asynchronous distributed learning algorithms remain focus on basic SGD-type algorithms and their simple variants. Asynchronous distributed algorithms with advanced acceleration techniques, e.g., variance reduction in particular, remain quite limited.

The above pros and cons of synchronous and asynchronous distributed algorithms prompt us to consider the following open questions in this paper:

- *Is it possible to develop a new distributed computing approach that achieves the best of both worlds while avoiding their pitfalls?*
- *Could this new distributed computing approach go beyond simple SGD-type algorithmic structure to include advanced acceleration techniques?*

SA-SpiderBoost: Semi-Asynchronous Distributed Learning with Variance Reduction: In this paper, we show that the answers to the above two questions are affirmative. More interestingly, we found that the "*double-loop*" structure of the state-of-the-art SpiderBoost algorithm naturally implies an elegant *semi-asynchronous* implementation, which entails the same ease of implementation as most asynchronous distributed algorithms, while providing strong convergence and generalization performance guarantees as most synchronous distributed algorithms. Specifically, in this paper, we introduce a new distributed learning algorithm called Semi-Asynchronous SpiderBoost (SA-SpiderBoost). As its name suggests, this algorithm is a semi-asynchronous version

of SpiderBoost, which allows larger step-sizes and converges with the same convergence rate as SPIDER on non-convex problems without sacrificing computational complexity. The basic idea of SA-SpiderBoost is: i) in each iteration of the outer loop, SA-SpiderBoost works in a *synchronous* mode to compute a full gradient; and ii) in each iteration of the inner loop, SA-SpiderBoost works in an *asynchronous* mode, where each worker is only responsible for its own computation task and does not need to coordinate with other workers when updating the parameter server.

Based on the above basic idea, in this paper, we investigate the convergence performance of SA-SpiderBoost on two different parallel architectures at different spatial scales: one is the *distributed memory* parallelism over multiple machines, and the other one is the *shared memory* parallelism on a single multi-core machine. The key difference is that the distributed memory parallelism works on *multiple* machines and allows to read and write on a whole vector $\mathbf{x}$, while the shared memory architecture works only on a *single multi-core machine* and only allows one to read and write a single coordinate of $\mathbf{x}$ at a time upon locking [22]. Also, we analyze the generalization performance of the SA-SpiderBoost through the lens of the algorithmic stability framework [10]. We note that the double-loop structure and the asynchrony between different workers render the performance analysis of SA-SpiderBoost highly challenging and necessitate new proof strategies and techniques. Our main contributions in this paper are summarized as follows:

- We first analyze the convergence performance of the distributed SA-SpiderBoost method for non-convex learning optimization problems under the distributed memory architecture. We show that SA-SpiderBoost achieves a computational complexity $O(\sqrt{N}\epsilon^{-2}(\Delta+1)+N)$ in terms of stochastic first-order oracle (SFO) evaluations, where Δ denotes the maximum delay in asynchronous update across all workers. Our result shows that larger delays only linearly affects the hidden constant in convergence performance and does *not* slow down the algorithm convergence rate in order sense. Also, fewer number of training samples N may speed up the running time when they reach similar training loss results (i.e., lower sample complexity). Experimental results also verify our theoretical findings.
- Next, we analyze the convergence performance of the SA-SpiderBoost method for non-convex learning optimization problems under the shared memory architecture. We show that the SA-SpiderBoost method achieves a computational complexity $O(\sqrt{N}\epsilon^{-2}(\Delta+1)d+N)$ in terms of SFO evaluations. This result reveals an interesting insight that, compared to distributed memory parallelism where all coordinates of a vector can be updated simultaneously, one needs to pay an additional cost that is d times larger due to the restriction that only one vector coordinate can be updated at a time. Note, however, that the restriction under shared memory parallelism does *not* affect the convergence rate with respect to ϵ. Our experimental results also verify this theoretical finding.
- We further study the generalization performance of SA-SpiderBoost under both distributed memory and shared memory parallel architectures based on the algorithmic stability framework [10]. We establish generalization error performance bounds for the expected generalization error of SA-SpiderBoost under both (quadratic) convex and non-convex cases, respectively. Our generalization bounds show that larger step-sizes and smaller training datasets cause larger generalization errors in SA-SpiderBoost. These results also characterize a fundamental trade-off between convergence and generalization in stochastic variance-reduced methods. Experimental results also corroborate this trade-off.

The rest of the paper is organized as follows. In Section 2, we first discuss the related work in the areas of variance-reduced methods and asynchronous algorithms for distributed machine learning, and then we provide a primer on parallel architectures in distributed computing to familiarize reader with the background and differences in distributed memory vs. shared memory parallelisms.

In Section 3, we present the system model, problem formulation, and basic assumptions used throughout the paper. In Section 4, we analyze both convergence and generalization performances of the SA-SpiderBoost method under the distributed memory parallelism. In Section 5, we analyze both convergence and generalization performances of the SA-SpiderBoost method under the shared memory parallelism. Section 6 presents numerical results and Section 7 concludes this paper.

2 RELATED WORK AND PRELIMINARIES

To put our work in comparison perspectives, in Section 2.1, we first provide a more in-depth overview on the related work on variance-reduced first-order methods for machine learning, asynchronous distributed first-order methods for non-convex learning, as well as the related work on generalization error analysis based on algorithmic stability. Then, in Section 2.2, we offer some necessary background on two widely adopted parallel computing architectures, namely distributed memory and shared memory.

2.1 Related Work

• **Variance-Reduced First-Order Methods:** As a standard algorithm for solving large-scale statistical learning problems, the history of the basic SGD algorithm dates back to 1951 [24]. As mentioned in Section 1, the vanilla SGD method achieves an ϵ-approximate stationary point with a gradient evaluation cost of $O(\epsilon^{-4})$ [9]. To address this limitation, variance-reduced techniques [14, 25] have emerged in recent years as an attractive solution and has been widely used to improve the convergence rate of SGD. Notably, SAGA [6] and SVRG (Stochastic Variance-Reduced Gradiet) [13] have been shown to achieve an overall SFO complexity of $O(N^{2/3}\epsilon^{-2})$ to attain an ϵ-approximate first-order stationary point for non-convex optimization. More recently, SPIDER (Stochastic Path-Integrated Differential Estimator) [7] and SpiderBoost [29] achieve an ϵ-approximate first-order stationary point with at most $O(\min(\sqrt{N}\epsilon^{-2}, \epsilon^{-3}))$ SFO complexity.

• **Asynchronous Distributed Optimization Algorithms for Learning:** As mentioned in Section 1, designing asynchronous SGD-type algorithms has received increasing attention recently due to its simplicity in implementation and practical relevance in many machine learning frameworks. One of the earliest studies on asynchronous SGD-type algorithms is the HOGWILD! algorithm in [22]. HOGWILD! is a lock-free asynchronous parallel implementation of SGD on the shared memory system with sublinear convergence rate for strongly convex smooth problems. At roughly the same time, the authors of [2] considered distributed stochastic convex optimization problems and studied the convergence performance of SGD-based optimization algorithms with asynchronous and yet delayed gradients. They showed that the convergence rate of their algorithm is $O(1/\sqrt{k})$ if the random gradient update delay is i.i.d second-moment-bounded. Interestingly, this $O(1/\sqrt{k})$ asymptotic convergence rate is the same as that of the synchronous and non-delayed case. However, stronger assumptions, such as compactness of feasible domain and bounded gradient, are required in this work. Later, the authors of [20] proposed a coordinate-descent-based asynchronous parallel optimization framework termed ARock, which generalizes asynchronous SGD (Async-SGD) for solving convex optimization problems. They showed that ARock achieves almost-sure weak and strong convergence, linear convergence rate, and almost-linear speedup under certain assumptions. The most related work to ours is [23], where an asynchronous stochastic variance-reduced method was analyzed for convex objectives and bounded delay. It has been shown that this algorithm has a linear convergence rate in such a setting. However, this work is only limited to convex cases and was proposed much earlier than the latest advances in variance-reduced methods, such as SPIDER, SpiderBoost, etc.

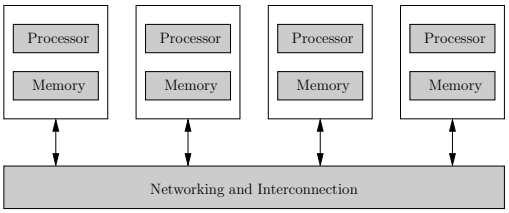


Fig. 1. The distributed memory architecture.

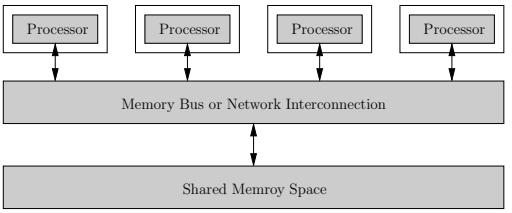


Fig. 2. The shared memory architecture.

• **Generalization error and algorithmic stability:** Generalization analysis for machine learning also has a long history. In recent years, algorithmic stability [10] has emerged as an attractive analytical framework for generalization error analysis. It has been shown in [10] that uniform stability implies generalization in expectation, based on which the authors further discussed how SGD ensures uniform stability on both convex and non-convex objectives. The work in [15, 19] showed that different learning rates, initialization, and update rules may cause different generalization behaviors. Generalization error of SGD with Gaussian noise is studied in [18]. Recently, a variance-dependent generalization bound for SGD under probability demands has been established in [37]. We note that the above studies on the generalization performance of SGD only considered the sequential setting rather that the parallel settings considered in this paper.

2.2 Distributed Memory vs. Shared Memory: A Primer

To facilitate later technical discussions, in this section, we provide a quick overview on distributed memory and shared memory parallel architectures, two of the most common distributed computing architectures in practice.

1) Distributed Memory: As it name suggests, distributed memory refers to a multi-processor computing system, where each processor has its own private memory. Computational tasks can only operate on local data, and if remote data is required, the computational task must communicate with one or more remote processors. As illustrated in Fig. 1, a distributed memory system typically contain processors and their associated local memory, as well as some form of interconnection that allows programs on each processor to interact with each other. The interconnect can be organized with point-to-point links or separate hardware can provide a switching network.

In the context of distributed learning, thanks to the independence between each computing node (referred to as worker in the rest of the paper), each worker can independently compute a stochastic gradient of the objective function based on local data and update all coordinates at the parameter server simultaneously without affecting other workers' operations.

2) Shared Memory: In contrast, a shared memory system offers a single memory space used by all processors. As shown in Fig. 2, processors do not have to be aware where data resides, except that race conditions (i.e., collisions) should be avoided. The shared memory space could be simultaneously accessed by multiple processors/programs. Note that shared memory also provides a means of passing data between programs. Depending on context, programs may run on a single processor or on multiple separate processors. Also, using memory for communication inside a single program, e.g., among its multiple threads, is also referred to as shared memory.

In the context of distributed learning with shared memory in this paper, the shared memory architecture only includes a single machine with multiple cores/GPUs sharing the same memory. The values of parameter $\mathbf{x}$ stored in the shared memory and there are P threads that can access it. Each thread can read the freshest $\mathbf{x}$-value from the shared memory and can randomly choose any subset S of samples to compute a stochastic gradient. Each thread then updates the current

parameters based on asynchronous stochastic gradient information. However, since the shared memory architecture works only on a multi-core machine, it can only allow to read a write a single coordinate of $\mathbf{x}$ at a time to prevent race conditions [22].

With the basic notions of distributed and shared memory architectures above, we are now in a position to analyze the convergence and generalization performances of our proposed SA-SpiderBoost algorithm under both distributed memory and shared memory architectures, respectively.

3 SYSTEM MODEL, PROBLEM FORMULATIONS AND BASIC ASSUMPTIONS

In this section, we present the system model, problem formulation and the assumptions that will be used throughout the rest of the paper. In this paper, we consider the standard supervised learning setting. As shown in Fig. 3, there are P workers and a parameter server in the system. There are N data samples in total in the global dataset, which is denoted as $\mathcal{S} = (\xi_1, ..., \xi_N)$. Each sample ξ_i is independently and identically distributed (i.i.d.) according to some latent distribution $\mathcal{D}$. The global dataset $\mathcal{S}$ in dispersed in each worker, and each local dataset at worker p is denoted as $\mathcal{S}_p$, with $\sum_{p=1}^{P} |\mathcal{S}_p| = N$. For simplicity, we assume equal distribution and let $n \triangleq |\mathcal{S}_p|$[1]. The goal of the system in distributed learning is to solve the following optimization problem in the form of:

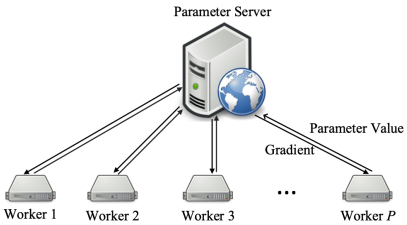


Fig. 3. A parallel computing architecture for distributed training.

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) \triangleq \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}, \xi_i) = \frac{1}{N}\sum_{p=1}^{P}\sum_{i=1}^{|\mathcal{S}_p|} f(\mathbf{x}, \xi_i) = \frac{1}{nP}\sum_{p=1}^{P}\sum_{i=1}^{n} f(\mathbf{x}, \xi_i).$$

In this paper, we make the following assumptions:

Assumption 1 (Bounded Objective Function). *The objective function $f(\cdot,\cdot)$ is bounded from below, i.e., $f^* = \inf_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}, \cdot) > -\infty$.*

Assumption 2 (IID Local Datasets). *All random samples ξ_i in dataset $\mathcal{S}$ are i.i.d. according to a common latent distribution $\mathcal{D}$.*

Assumption 3 (Continuously Differentiable Loss Function). *The loss function $f(\cdot,\cdot)$ is continuously differentiable.*

Assumption 4 (M-Lipschitz Continuous Loss Function:). *The loss function $f(\cdot,\cdot)$ is M-Lipschitz continuous, i.e., there exists a constant $M > 0$ such that the loss function $f(\cdot,\cdot)$ satisfies $|f(\mathbf{u}, \xi_i) - f(\mathbf{v}, \xi_i)| \leq M\|\mathbf{u} - \mathbf{v}\|$, $\forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^d$, $\forall \xi_i$.*

Assumption 5 (Bounded-Size gradient:). *There exists a constant $M > 0$ such that the norm of ∇f is uniformly bounded by M, i.e., $sup_{\mathcal{S}} \|\nabla f(\cdot, \xi_i)\| \leq M$.*

We remark that Assumptions 4 and 5 share the same constant M because Assumption 5 is implied by Assumption 4 (we state Assumption 5 explicitly for convenience in our subsequent analysis). We note that the bounded gradient norm can also be implied by stability-inducing operations, such as regularization, projection and gradient clipping.

[1]For simplicity, we assume here that N is divisible by P. Note that, with slightly more cumbersome notation in the analysis, all our proofs and results continue to hold in cases where N is not divisible by P or unequal distributions.

Assumption 6 (L-Lipschitz Continuous Gradient). *The loss function $f(\cdot,\cdot)$ has L-Lipschitz continuous gradients (or called L-smooth), i.e., there exists a constant $L > 0$ such that $\| \nabla f(\mathbf{u}, \xi_i) - \nabla f(\mathbf{v}, \xi_i) \| \leq L \| \mathbf{u} - \mathbf{v} \|, \forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^d, \forall \xi_i$.*

Assumption 7 (Bounded Delay in Stochastic Gradients). *The random delay τ of all stale stochastic gradients in asynchronous updates with respect to the current time-slot is upper bounded by a constant $\Delta > 0$, i.e., $\tau \leq \Delta$.*

We note that Assumptions 1–6 are standard for convergence analysis in the distributed learning and optimization literature. Assumption 7 is also a standard assumption in the asynchronous computing literature and holds in most practical computing systems.

4 SA-SPIDERBOOST UNDER THE DISTRIBUTED MEMORY ARCHITECTURE

In this section, we focus on our proposed SA-SpiderBoost algorithm for the distributed memory architecture. Toward this end, we first describe our algorithm in Section 4.1 and then present its convergence analysis in Section 4.2. Lastly, we will analyze the generalization performance of SA-SpiderBoost in Section 4.3.

4.1 Algorithm Description

To handle ML training problems with a large dataset that cannot fit in a single computer, we propose a robust and efficient SA-SpiderBoost algorithm under the distributed memory architecture. As mentioned in Section 1, it is costly to maintain a common clock in a large-scale distributed computing system, and asynchronous algorithms can significantly simplify the implementation complexity. However, a key challenge in asynchronously training large-scale machine learning problems stems from the "delay" issue. Specifically, when a "worker" node asynchronously contributes a gradient update to the parameter server, the global model parameter may have changed by another worker, rendering this gradient information stale. In a massively parallel computing system, if these delays are not treated carefully, they can affect the convergence of the training process in an intractable way. Therefore, in this paper, we pursue a new *semi-asynchronous* approach to achieve the best of both worlds.

Specifically, we note that the SpiderBoost algorithm [29] – a state-of-the-art variance-reduced method for ML training– lends itself to a simple and elegant semi-asynchronous implementation thanks to its *double-loop structure*. Specifically, in the outer loop of SpiderBoost and other algorithms in the family of variance-reduced methods, the parameter server needs to collect information from *all* workers to compute full gradients. Hence, it is natural to perform the outer loop in synchronous fashion. On the other hand, the inner loop only requires stochastic gradient information from the workers, which is a perfect fit for asynchronous operations. The server and worker algorithms of our SA-SpiderBoost are presented in Algorithms 1 and 2. In what follows, we take a closer look at these algorithms.

1) The Inner Loop (Asynchronous Mode): From Algorithm 1, it can be seen that the algorithm is executed K iterations in total. In the inner loop of the SA-SpiderBoost algorithm, on the worker side (Steps 3–6 of Algorithm 2), each worker i independently retrieves the freshest parameter $\mathbf{x}_{new}$ from the parameter server and then randomly selects a subset S of data samples from its local dataset and computes a stochastic gradient. Also, v_{old} is an unbiased gradient estimate of $\nabla f(\mathbf{x}_{old})$. Each worker immediately reports the computed stochastic gradient v_{new} to the parameter server. The parameter server then updates its current parameters with this stochastic gradient information without waiting for other workers, hence operating in an *asynchronous* mode. Note that, while this worker is trying to send its gradient information to the parameter server, the parameter server may

Algorithm 1: The Parameter Server Code of the Distributed SA-SpiderBoost Algorithm.

Input: $\epsilon, \eta, q, K \in \mathbb{N}$

Output: $\mathbf{x}_\zeta$, where ζ chosen uniformly at random from $\{1, \ldots, K\}$.

1 **for** $k = 0, 1, ..., K-1$ **do**
2 **if** $mod(k, q) = 0$ **then**
3 Send a signal to all workers to interrupt all unfinished computing jobs at each worker.
4 Push $\mathbf{x}_k$ to all workers.
5 Wait until receiving $G_k^{(p)}$, $\forall p$, from all workers.
6 Compute full gradient $\mathbf{v}_k = \nabla f(\mathbf{x}_k) = \frac{1}{N} \sum_{p=1}^{P} G_k^{(p)}$
7 Broadcast $\mathbf{x}_{old} = \mathbf{x}_k$ and $\mathbf{v}_{old} = \mathbf{v}_k$ to all workers.
8 **else**
9 Let $\mathbf{v}_k = \mathbf{v}_{new}$ be the feedback from a specific worker with delays.
10 Update parameter $\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \mathbf{v}_k$.

Algorithm 2: The Code of Worker p of the Distributed SA-SpiderBoost Algorithm.

1 If receiving interrupt signal at any time, go immediately to Step 2; otherwise, go to Step 3.
2 Receive $\mathbf{x}_k$ form server. Compute $G_k^{(p)} = \sum_{i \in \mathcal{S}_p} \nabla f(\mathbf{x}_k, \xi_i)$ and send it to the parameter server. Wait and receive $\mathbf{x}_{old}$ and $\mathbf{v}_{old}$ form server and go to Step 1.
/ In the following steps, stop and go to Step 2 immediately upon receiving an interrupt signal from the server */*
3 Pull the most fresh parameter of $\mathbf{x}$ as $\mathbf{x}_{new}$ from server.
4 Randomly select a subset S of samples from $\mathcal{S}_p$.
5 Compute $\mathbf{v}_{new} = \frac{1}{|S|} \sum_{i \in S} (\nabla f(\mathbf{x}_{new}, \xi_i) - \nabla f(\mathbf{x}_{old}, \xi_i) + \mathbf{v}_{old})$ and send it to the parameter server, which takes combined τ_k time-slots in computation and communication.
6 Let $\mathbf{x}_{old} = \mathbf{x}_{new}, \mathbf{v}_{old} = \mathbf{v}_{new}$, go to Step 3.

have been updated by other workers with gradient information associated with a fresher $\mathbf{x}$-value. Thus, this particular worker's gradient information becomes "delayed."

2) The Outer Loop (Synchronous Mode): On the other hand, at the beginning of every timeframe of length q, the parameter server executes the outer loop. Specifically, the parameter server sends out an interruption signal to all workers to cancel the unfinished computation at each worker if there is any (Step 3 in Algorithm 1). Upon receiving the interruption signal, every worker stops and then retrieves the freshest parameter $\mathbf{x}_k$ from the parameter server and uses all samples in its local dataset to compute a local full gradient and send the result to the parameter server (Step 2 in Algorithm 2). The parameter server collects the gradient information from *all* workers, hence operating in a *synchronous* mode. Finally, the server computes the global full gradient (Step 6 in Algorithm 1) and sends out the most recent $\mathbf{x}$- and $\mathbf{v}$-values as $\mathbf{x}_{old}$ and $\mathbf{v}_{old}$ to all workers (Step 7 in Algorithm 1), which are needed for the first variance-reduced operation in the first iteration of the inner loop. Finally, the parameter server updates the $\mathbf{x}$-value along the global full gradient direction (Step 10 in Algorithm 1).

In each iteration, parameter $\mathbf{x}$ is updated through the following update rule: $\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \mathbf{v}_k$, where η is a constant learning rate, $\mathbf{v}_k$ represents the variance-reduced update direction in iteration k, which can also be written as: $\mathbf{v}_k = \frac{1}{|S|} \sum_{i \in S} (\nabla f(\mathbf{x}_{k-\tau^k}, \xi_i) - \nabla f(\mathbf{x}_{k-\tau^k-1-\tau^{k-\tau^k-1}}, \xi_i) + \mathbf{v}_{k-1-\tau^k})$.

Here, $\mathbf{x}_{k-\tau^k}$ denotes the freshest parameter we used to compute the gradient with delay τ^k in a worker, $\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}}$ denotes the old parameter we used to compute the gradient with additional delay $\tau^{k-1-\tau^k}$. The index i denotes the index of a sample, τ^k denotes the time delay at iteration k, and Δ is the maximum delay of all workers. Upon receiving the updated gradient from a specific worker, the parameter server updates its current parameter with $\mathbf{v}_k$ if in the inner loop, or wait until finishing the collection of all workers' $\mathbf{v}$-values to compute the full gradient and update.

4.2 Convergence Analysis

In this subsection, we first present the main convergence result of SA-SpiderBoost under the distributed memory architecture. Due to space limitation, we provide a proof sketch in this subsection and relegate the full proof to Appendix A.

Theorem 1 (Convergence of SA-SpiderBoost for Distributed Memory). *Let f^* denote the global optimal value. Under distributed memory and the stated assumptions, for some $\epsilon > 0, P \leq \sqrt{N}$, by choosing parameters $q = |S| = \sqrt{N}$ and $\eta = \frac{1}{4L(\Delta+1)}$ for a given maximum delay $\Delta > 0$, the SA-SpiderBoost algorithm outputs an $\mathbf{x}_\zeta$ that satisfies $\mathbb{E}\left[\left\|\nabla f(\mathbf{x}_\zeta)\right\|\right] \leq \epsilon$ if the total number of iterations K satisfies $K = O\big(\frac{f(\mathbf{x}_0)-f^*}{\epsilon^2}\big)$. This also implies that the total SFO complexity is $O(\sqrt{N}\epsilon^{-2}(\Delta+1)+N)$.*

Remark 1. Two remarks on Theorem 1 are in order: i) Theorem 1 shows that the output of SA-SpiderBoost achieves a first-order stationary point with total computational complexity $O(\sqrt{N}\epsilon^{-2}(\Delta+1)+N)$. SA-SpiderBoost enhances Spider-Boost by utilizing asynchronous parallel system. For the special case with maximum delay $\Delta = 0$, the result of Theorem 1 recovers the total computational complexity $O(\sqrt{N}\epsilon^{-2}+N)$ of the original synchronous SpiderBoost algorithm, where N is the total training samples. ii) We note that the analysis of SA-SpiderBoost is very different from that of the synchronous SpiderBoost. From technical perspectives, the new major challenge in the proof of Theorem 1 that is different from the proof for the basic SpiderBoost [29] lies in the facts that: a) the analysis of distributed asynchronous Spider-Boost contains a variance-reduction component, which appears in the algorithm; and b) the asynchrony between different workers.

Proof Sketch of Theorem 1. Here, we provide a proof sketch and relegate the full proof to Appendix A. To prove the stated result in Theorem 1, we start with evaluating $\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|^2$ (ζ is chosen uniformly at random from $\{1,\ldots,K\}$). Following the inequality of arithmetic and geometric means, we have:

$$\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|^2 = \mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta) - \mathbf{v}_\zeta + \mathbf{v}_\zeta\right\|^2 \leq 2\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta) - \mathbf{v}_\zeta\right\|^2 + 2\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2. \tag{1}$$

Next, we bound the terms $\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta) - \mathbf{v}_\zeta\right\|^2$ and $\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2$ on the right-hand-side of (1) individually.

Step 1) Bounding $\mathbb{E}\left\|\mathbf{v}_\zeta - \nabla f(\mathbf{x}_\zeta)\right\|^2$: Toward this end, we will first bound the distance $\mathbb{E}\left\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau^k}, \xi_i)\right\|^2$ of the inner loop. Let $n_k = \lceil k/q \rceil$ (i.e., $(n_k-1)q \leq k \leq n_k q - 1$). Then, from the inner loop operations, we can show the following:

$$\mathbb{E}\left\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau^k}, \xi_i)\right\|^2 \leq \frac{L^2\eta^2(\Delta+1)}{|S|} \times \sum_{j=(n_k-1)q}^{k-1-\tau^k} \mathbb{E}\left\|\mathbf{v}_j\right\|^2 + \mathbb{E}\left\|\mathbf{v}_{(n_k-1)q} - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{(n_k-1)q}, \xi_i)\right\|^2. \tag{2}$$

Using (2) above, we can further derive that:

$$\mathbb{E}\left\|\mathbf{v}_k - \nabla f(\mathbf{x}_k)\right\|^2 = \mathbb{E}\left\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_k,\xi_i)\right\|^2 \leq \frac{2L^2\eta^2(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k-1-\tau^k}\mathbb{E}\left\|\mathbf{v}_j\right\|^2$$
$$+ 2\mathbb{E}\left\|\mathbf{v}_{(n_k-1)q} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{(n_k-1)q},\xi_i)\right\|^2 + 2L^2\eta^2\Delta\sum_{j=k-\tau^k}^{k-1}\mathbb{E}\left\|\mathbf{v}_j\right\|^2. \quad (3)$$

Based on (2), by using β_1 defined in Step 2) and the bound on $\sum_{i=1}^{K-1}\mathbb{E}\|\mathbf{v}_i\|^2$, we can further bound $\mathbb{E}\left\|\mathbf{v}_\zeta - \nabla f(\mathbf{x}_\zeta)\right\|^2$ as:

$$\mathbb{E}\left\|\mathbf{v}_\zeta - \nabla f(\mathbf{x}_\zeta)\right\|^2 \leq \left[\frac{2L^2(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^3}{\beta_1} + 2\right]\epsilon_1^2 + 2L^2(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^2\left[\frac{f(\mathbf{x}_0)-f^*}{K\beta_1}\right]. \quad (4)$$

Step 2) Bounding $\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2$*:* Now, consider the term $\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2$ in (1). To evaluate $\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2$, we start from the iteration relationship of our SA-SpiderBoost algorithm. From the L-smooth assumption, it can be shown that:

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) + \frac{\eta}{2}\left\|\mathbf{v}_k - \nabla f(\mathbf{x}_k)\right\|^2 - (\frac{\eta}{2} - \frac{L\eta^2}{2})\left\|\mathbf{v}_k\right\|^2. \quad (5)$$

It then follows from (3) and (5) that

$$\mathbb{E}f(\mathbf{x}_{k+1}) \leq \mathbb{E}f(\mathbf{x}_k) - (\frac{\eta}{2} - \frac{L\eta^2}{2})\mathbb{E}\left\|\mathbf{v}_k\right\|^2$$
$$+ \eta\epsilon_1^2 + \frac{L^2\eta^3(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 + L^2\eta^3\Delta\sum_{j=k-\tau^k}^{k-1}\mathbb{E}\left\|\mathbf{v}_j\right\|^2, \quad (6)$$

Inductively bounding $\mathbb{E}f(\mathbf{x}_{k+1})$ in the inner loop $(n_k-1)q \leq k \leq n_k q - 1$, we have:

$$\mathbb{E}f(\mathbf{x}_{k+1}) \leq \mathbb{E}f(\mathbf{x}_{(n_k-1)q}) + \sum_{i=(n_k-1)q}^{k}\eta\epsilon_1^2 - \left[\frac{\eta}{2} - \frac{L\eta^2}{2} - L^2\eta^3\left(\frac{q(\Delta+1)}{|S|} + \Delta^2\right)\right]\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2. \quad (7)$$

Next, letting $\beta_1 \triangleq \left[\frac{\eta}{2} - \frac{L\eta^2}{2} - L^2\eta^3\left(\frac{q(\Delta+1)}{|S|} + \Delta^2\right)\right]$ and inductively using (7), we can further drive the final upper bound of $\mathbb{E}f(\mathbf{x}_{k+1})$ as:

$$\mathbb{E}f(\mathbf{x}_K) - \mathbb{E}f(\mathbf{x}_0) \leq -\sum_{i=0}^{K-1}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2) + K\eta\epsilon_1^2, \quad (8)$$

which further implies that:

$$\mathbb{E}f(\mathbf{x}^*) - \mathbb{E}f(\mathbf{x}_0) \leq -\sum_{i=0}^{K-1}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2) + K\eta\epsilon_1^2. \quad (9)$$

Rearranging terms in (9) yields:

$$\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2 = \frac{1}{K}\sum_{i=0}^{K-1}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 \leq \frac{f(\mathbf{x}_0)-f^*}{K\beta_1} + \frac{\eta}{\beta_1}\epsilon_1^2. \quad (10)$$

Step 3): By combining results in Steps 1) and 2) and plugging them into (1), we arrive at:

$$\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|^2 \leq \left[\frac{2}{\beta_1}(\eta + 2L^2(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^3) + 4\right]\epsilon_1^2 +$$

$$\left[\frac{4L^2(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^2}{K\beta_1}+\frac{2}{K\beta_1}\right](f(\mathbf{x}_0)-f^*). \quad (11)$$

Lastly, we choose the following parameter:

$$q=\sqrt{N}, \quad S=\sqrt{N}, \quad \eta=\frac{1}{4L(\Delta+1)}. \quad (12)$$

Plugging the above parameters in the definitions of β_1, we obtain that:

$$\beta_1=\frac{\eta}{2}-\frac{L\eta^2}{2}-L^2\eta^3\left(\frac{q(\Delta+1)}{|S|}+\Delta^2\right)=\frac{1}{8L(\Delta+1)}\cdot\frac{14\Delta^2+26\Delta+10}{16(\Delta+1)^2}>0. \quad (13)$$

For $mod(k,q)=0$ we have $\mathbb{E}\left\|\mathbf{v}_k-\nabla f(\mathbf{x}_k)\right\|^2=0$. Then, after K iterations, we have:

$$\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|^2\leq 16L(\Delta+1)\frac{9\Delta^2+17\Delta+9}{K\cdot(7\Delta^2+13\Delta+5)}(f(\mathbf{x}_0)-f^*). \quad (14)$$

To ensure $\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|\leq\epsilon$, it sufficies to ensure $\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|^2\leq\epsilon^2$. It then follows from (14) that:

$$K=16L(\Delta+1)\frac{9\Delta^2+17\Delta+9}{\epsilon^2\cdot(7\Delta^2+13\Delta+5)}(f(\mathbf{x}_0)-f^*)=\mathcal{O}\left(\frac{f(\mathbf{x}_0)-f^*}{\epsilon^2}\right).$$

This completes the first part of the theorem.

Finally, to show the SFO complexity, we note that the number of SFO calls in the outer loops can be calculated as $\lceil\frac{K}{q}\rceil N$. Also, the number of SFO calls in the inner loop can be calculated as KS. Hence, the total SFO complexity can be calculated as: $\lceil\frac{K}{q}\rceil N+K\cdot S\leq\frac{K+q}{q}N+K\sqrt{N}=K\sqrt{N}+N+K\sqrt{N}=O(\sqrt{N}\epsilon^{-2}(\Delta+1)+N)$. This completes the proof. □

4.3 Generalization Analysis for Distributed Memory SA-SpiderBoost

After studying the convergence performance of SA-SpiderBoost under the distributed memory architecture, in this subsection, we turn our attention to the generalization performance of SA-SpiderBoost under distributed memory, i.e., how accurate a model trained by SA-SpiderBoost is when it is fed by new data outside of the training dataset. Formally, generalization error can be defined as follows. Let $F_N(\mathbf{x})\triangleq\min_{\mathbf{x}\in\mathbb{R}^d}\frac{1}{N}\sum_{i=1}^N f(\mathbf{x};\xi_i)$ represent the minimum finite-sample empirical risk. The minimum empirical risk above is a sample-average proxy for the minimum population risk, i.e., $F(\mathbf{x})\triangleq\min_{\mathbf{x}\in\mathbb{R}^d}\mathbb{E}_{\xi\in\mathcal{D}}f(\mathbf{x};\xi)$, where the sample ξ is drawn from an underlying latent distribution $\mathcal{D}$. Then, the generalization error can be defined as follows:

Definition 1 (Generalization Error). *The average generalization error is defined as the gap between the empirical risk minimization problem and the population risk minimization problem, i.e.,* $|\mathbb{E}_{S,A}[F_N[A(S)]-F[A(S)]|$.

Note that, in the above definition, the expectation is taken over all randomness in the training dataset distribution and the algorithm A (A could potentially be randomized). Our generalization analysis is based on the algorithmic stability framework [10], which is formally defined as follows:

Definition 2 (Algorithmic Stability [10]). *Let S and S' be two datasets such that S and S' differ by at most one element, For an $\epsilon'>0$, an algorithm A is said to be ϵ'-uniformly stable if* $\sup_{\xi\in\mathcal{D}}\mathbb{E}_A[f(A(S);\xi)-f(A(S');\xi)]\leq\epsilon'$, *where $\mathcal{D}$ denotes the distribution of data sample ξ.*

Intuitively speaking, a stable learning algorithm has the property that changing one element in its training data set does not lead to a dramatic change in its outcome. It has been shown in [4]

that if an algorithm A is ϵ'-uniformly stable, then the average generalization error can be bounded by ϵ', which can be formally stated as follows:

Lemma 1 (Generalization in Expectation [4]). *Let A be ϵ'-uniformly stable. Then, the average generalization error is bounded as $|\mathbb{E}_{S,A}[F_N[A(S)] - F[A(S)]]| \leq \epsilon'$.*

In this subsection, our goal is to study the stability of the SA-SpiderBoost algorithm under the distributed memory architecture. Recall that the update rule for SA-SpiderBoost is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \mathbf{v}_k = \mathbf{x}_k - \eta \frac{1}{|S|} \sum_{i \in S} (\nabla f(\mathbf{x}_{k-\tau^k}, \xi_i) - \nabla f(\mathbf{x}_{k-\tau^k-1-\tau^{k-\tau^k-1}}, \xi_i) + \mathbf{v}_{k-1-\tau^k}),$$

and we run this update iteratively for a maximum number of K steps. Now, consider two datasets $S = \{\xi_1, \xi_2, ..., \xi_N\}$ and $S' = (\xi_1', \xi_2', ..., \xi_N')$, which differ by at most one sample. The next two theorems state our main results on algorithmic stability (or equivalently, generalization error) of SA-SpiderBoost for convex and non-convex loss functions, respectively. Due to space limitation, the proof details are provided in Appendix B.

Theorem 2 (Stability in the Quadratic Strongly Convex and Smooth Setting). *Suppose that the loss function $f(x; \xi_i)$ is a quadratic, L-smooth, μ -strongly convex, and M-Lipschitz. The distributed-memory SA-SpiderBoost algorithm is $(\frac{2\eta M^2 K}{N})$-uniformly stable.*

Theorem 3 (Stability in the Non-Convex and Smooth Setting). *Suppose that the loss function $f(x, \xi_i)$ is continuously differentiable, L-smooth and $M-Lipschitz$ for every i. The distributed-memory SA-SpiderBoost algorithm is $(2\eta M^2 K^2 + \frac{2M^2K^2\eta}{N})$-uniformly stable.*

Remark 2. Two remarks on Theorems 2 and 3 are in order: i) Although appearing restrictive, the quadratic strongly convex and smooth setting remains of practical interest. For example, in many applications, the square loss and linear learning model are widely adopted, which naturally fit the quadratic strongly convex and smooth setting. On the other hand, many other learning problems nowadays (e.g., deep neural networks) adopt highly non-convex models, which can be covered by the result in Theorem 3. ii) It can be seen that the distributed SA-SpiderBoost algorithm under both convexity settings generalizes better as the number of training data samples N increases, or the small learning rate η decreases. Since a small learning rate implies slower convergence, this suggests that there is a fundamental *trade-off* between training convergence speed and generalization performance. Furthermore, the learning problem under a stronger convexity condition leads to tighter stability bound (i.e., generalizes better). iii) From Theorems 2 and 3, we can see that SA-SpiderBoost has a larger stability error bound in non-convex cases. Whether or not the non-convex stability bound could be further sharpened remains an open question. However, our numerical results in Section 6 suggest that our non-convex stability error bound is tight.

We note that our generalization analysis offers the first theoretical understanding of generalization performance for semi-asynchronous variance-reduced learning algorithms. We note that our proof technique is different from existing works. The conventional idea used in existing works is to analyze the difference between $\|\mathbf{x}_{k+1} - \mathbf{x}_{k+1}'\|$ and $\|\mathbf{x}_k - \mathbf{x}_k'\|$, where k and k' denotes the outputs of the algorithm on datasets S and S', respectively. In contrast, our proof technique in Theorem 2 is inspired by linear control system analysis, where we analyze the difference between $\delta_{k+1} \triangleq \mathbf{x}_{k+1} - \mathbf{x}_{k+1}'$ and $\delta_k \triangleq \mathbf{x}_k - \mathbf{x}_k'$ directly. This helps us to transform the problem into a linear control system and thus obtain a tighter stability error bound in 2. Additionally, the variance reduction component and the asynchrony appears in the algorithm also create challenges in analyzing the stability performance of SA-SpiderBoost. We also note that the non-convexity in Theorem 3 poses further challenges in bounding the stability error.

Proof Sketch of Theorem 2. Here, we provide a sketch of proof of Theorem 2 and relegate the full proof in Appendix B.1. First, we first let $S = (\xi_1, \xi_2, ..., \xi_N)$ and $S' = (\xi_1', \xi_2', ..., \xi_N')$ be two adjacent datasets such that S and S' differ in at most one element. Let $\delta_k \triangleq \mathbf{x}_k - \mathbf{x}_k'$. We let $\mathbf{x}_0 = \mathbf{x}_0'$. To prove the result in Theorem 2, we start with evaluating $\mathbb{E}(\delta_{k+1})$.

Step 1) Simplifying the expression of δ_{k+1}: Since $\mathbf{v}_k$ is an unbiased estimate of $\nabla f(\mathbf{x}_{k-\tau^k})$, we have:

$$\begin{aligned}\mathbb{E}(\delta_{k+1}) &= \mathbb{E}(\mathbf{x}_{k+1}) - \mathbb{E}(\mathbf{x}_{k+1}') = \mathbb{E}(\delta_k) - \eta[\mathbb{E}(\mathbf{v}_k) - \mathbb{E}(\mathbf{v}_k')] \\ &= \mathbb{E}(\delta_k) - \eta[\mathbb{E}(\nabla f(\mathbf{x}_{k-\tau^k})) - \mathbb{E}(\nabla f(\mathbf{x}_{k-\tau^k}'))]. \end{aligned} \tag{15}$$

At Step k, with probability $1 - 1/N$, the training sample is the same in S and S'. On the other hand, with probability $\frac{1}{N}$, the training sample is different between S and S'. Next, we define

$$\epsilon'' \triangleq \mathbb{E}\Big[\frac{1}{N}\eta\nabla f((\mathbf{x}_{k-\tau^k}', \xi_i)) - \frac{1}{N}\eta\nabla f((\mathbf{x}_{k-\tau^k}', \xi_i'))\Big].$$

Based on the update rule of $\mathbf{v}_k$ and the quadratic property, we can show that:

$$\begin{aligned}\mathbb{E}(\delta_{k+1}) \leq& \mathbb{E}\Big[(\delta_k) - \eta\frac{1}{N}\sum_{i=1}^{N}\nabla f((\mathbf{x}_{k-\tau^k}, \xi_i)) + \eta\frac{1}{N}\sum_{i=1}^{N}\nabla f((\mathbf{x}_{k-\tau^k}', \xi_i))\Big] \\ &+ \mathbb{E}\Big[\frac{1}{N}\eta\nabla f((\mathbf{x}_{k-\tau^k}', \xi_i)) - \frac{1}{N}\eta\nabla f((\mathbf{x}_{k-\tau^k}', \xi_i'))\Big] \\ \leq& \mathbb{E}\Big[(\delta_k) - \eta A(\delta_{k-\tau^k})\Big] + \epsilon''. \end{aligned} \tag{16}$$

Step 2) Bounding $\mathbb{E}\|\delta_{k+1}\|$: Based on the relation between $\mathbb{E}(\delta_k), \mathbb{E}(\delta_{k-\tau^k})$ and $\mathbb{E}(\delta_{k+1})$ proved in Step 1), we can transform the problem to a *linear control system*, which lifts the state space into a higher dimensional space to bound $\mathbb{E}\|\delta_{k+1}\|$ as follows:

$$\begin{bmatrix}\delta_{k+1}\\ \delta_k\\ ...\\ \delta_{k-\tau^k+1}\\ \delta_{k-\tau^k}\end{bmatrix} = \begin{bmatrix}1 & 0 & \cdots & -\eta A & 0\\ 1 & 0 & \cdots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ \vdots & \vdots & & \ddots & \vdots\\ 0 & 0 & \cdots & 1 & 0\end{bmatrix}\begin{bmatrix}\delta_k\\ \delta_{k-1}\\ ...\\ \delta_{k-\tau^k}\\ \delta_{k-\tau^k-1}\end{bmatrix} + \begin{bmatrix}\epsilon''\\ 0\\ ...\\ 0\\ 0\end{bmatrix}. \tag{17}$$

For convenience, we define the coefficient matrix as $\mathbf{Q}$, i.e.,

$$\mathbf{Q} \triangleq \begin{bmatrix}1 & 0 & \cdots & -\eta A & 0\\ 1 & 0 & \cdots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ \vdots & \vdots & & \ddots & \vdots\\ 0 & 0 & \cdots & 1 & 0\end{bmatrix}. \tag{18}$$

We then show that the maximum eigenvalue of $\mathbf{Q}$ satisfies $\max(\lambda_{\mathbf{Q}}|) = 1$ after some algebraic manipulations. Since $\|\epsilon''\| \leq \frac{2\eta M}{N}$, then we have:

$$\mathbb{E}\|\delta_{k+1}\| \leq \mathbb{E}\|\delta_k\| + \frac{2\eta M}{N}. \tag{19}$$

For those k-values that satisfy $mod(k, q) = 0$, we can also show $\|(\delta_{k+1})\| = \|(\mathbf{x}_{k+1}) - (\mathbf{x}_{k+1}')\| \leq \|(\delta_k)\| + \frac{2\eta M}{N}$. Also, from (19), we always have $\|(\delta_{k+1})\| \leq \|(\delta_k)\| + \frac{2\eta M}{N}$ for $mod(k, q) \neq 0$. By applying this bound inductively, we can bound δ_{k+1} using the total number K iterations as:

$$\mathbb{E}\|\delta_{k+1}\| \leq \frac{2\eta MK}{N}. \tag{20}$$

Step 3) Bounding SA-SpiderBoost ϵ'-stability : Lastly, it follows from the definition of algorithmic stability and the M-Lipschitz assumption of the loss function that our SA-SpiderBoost algorithm has the following stability bound:

$$\epsilon' \leq M \cdot \mathbb{E} \parallel \delta_{K+1} \parallel \leq \frac{2\eta M^2 K}{N}. \tag{21}$$

This completes the proof. □

Proof Sketch of Theorem 3. Again, we provide a proof sketch here and relegate the full proof to Appendix B.2. Similar to the proof of Theorem 2, we first let $S = (\xi_1, \xi_2, ..., \xi_N)$ and $S' = (\xi_1', \xi_2', ..., \xi_N')$ be two adjacent datasets such that S and S' differ in at most one element. Let $\delta_k \triangleq \mathbf{x}_k - \mathbf{x}_k'$. Let $\mathbf{x}_0 = \mathbf{x}_0'$. We start with evaluating $\mathbb{E}(\delta_{k+1})$.

Step 1) Simplifying the expression of δ_{k+1}: Since $\mathbf{v}_k$ is an unbiased estimate of $\nabla f(\mathbf{x}_{k-\tau^k})$, we have:

$$\begin{aligned}\mathbb{E}(\delta_{k+1}) &= \mathbb{E}(\mathbf{x}_{k+1}) - \mathbb{E}(\mathbf{x}_{k+1}') = \mathbb{E}(\delta_k) - \eta[\mathbb{E}(\mathbf{v}_k) - \mathbb{E}(\mathbf{v}_k')] \\ &= \mathbb{E}(\delta_k) - \eta[\mathbb{E}(\nabla f(\mathbf{x}_{k-\tau^k})) - \mathbb{E}(\nabla f(\mathbf{x}_{k-\tau^k}'))].\end{aligned} \tag{22}$$

At Step k, with probability $1 - 1/N$, the sample is the same in S and S'. On the other hand, with probability $\frac{1}{N}$, the sample is different S between S'. Based on the update rule of $\mathbf{v}_k$, we have:

$$\begin{aligned}\mathbb{E}\|(\delta_{k+1})\| \leq& \mathbb{E}\Big\|(\delta_k) - \eta\frac{1}{N}\sum_{i=1}^{N}\nabla f((\mathbf{x}_{k-\tau^k}, \xi_i)) + \eta\frac{1}{N}\sum_{i=1}^{N}\nabla f((\mathbf{x}_{k-\tau^k}', \xi_i))\Big]\| \\ &+ \mathbb{E}\Big\|\frac{1}{N}\eta\nabla f((\mathbf{x}_{k-\tau^k}', \xi_i)) - \frac{1}{N}\eta\nabla f((\mathbf{x}_{k-\tau^k}', \xi_i'))\Big\| \\ \leq& \mathbb{E}\|(\delta_k)\| + \eta\mathbb{E}\|(\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau^k}, \xi_i)) - (\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau^k}', \xi_i))\| + \frac{2\eta M}{N} \\ \overset{(a)}{\leq}& \mathbb{E}\|(\delta_k)\| + 2\eta M + \frac{2\eta M}{N},\end{aligned} \tag{23}$$

where (a) follows from the bounded gradient assumption.

Step 2) Bounding $\mathbb{E}\|\delta_{k+1}\|$: For those k-values that satisfy $mod(k, q) = 0$, we have $\|(\delta_{k+1})\| = \|(\mathbf{x}_{k+1}) - (\mathbf{x}_{k+1}')\| \leq \|(\delta_k)\| + \frac{2\eta M}{N}$. Also, from (23), we always have $\|(\delta_{k+1})\| \leq \|(\delta_k)\| + 2\eta M + \frac{2\eta M}{N}$ for $mod(k, q) \neq 0$. By applying this bound inductively, we can bound δ_{k+1} using the total number K iterations as:

$$\mathbb{E} \parallel \delta_{k+1} \parallel \leq 2\eta MK + \frac{2\eta MK}{N}. \tag{24}$$

Step 3) Bounding SA-SpiderBoost ϵ'-stability: Lastly, it follows from the definition of algorithmic stability and the M-Lipschitz assumption of the loss function that our SA-SpiderBoost algorithm has the following stability error bound:

$$\epsilon' \leq M \cdot \mathbb{E} \parallel \delta_{K+1} \parallel \leq 2\eta M^2 K + \frac{2\eta M^2 K}{N}.$$

This completes the proof. □

5 SA-SPIDERBOOST FOR SHARED MEMORY ARCHITECTURE

In this section, we focus on our proposed SA-SpiderBoost algorithm for the shared memory architecture. Toward this end, we first describe our algorithm in Section 5.1 and then present its convergence results in Section 5.2. Lastly, we will analyze the generalization performance of SA-SpiderBoost for the shared memory architecture in Section 5.3.

Algorithm 3: The SA-SpiderBoost Algorithm for the Shared Memory Architecture.

Input: $\epsilon, \eta, q, K \in \mathbb{N}$

Output: $\mathbf{x}_\zeta$, where ζ chosen at random

1 **for** $k = 0, 1, ..., K-1$ **do**

2 **if** $mod(k, q) = 0$ **then**

3 Compute full gradient $\mathbf{v}_k = \nabla f(\mathbf{x}_k) = \frac{1}{N}\sum_{i=1}^{N} \nabla f(\mathbf{x}_k, \xi_i)$

4 Set $\mathbf{x}_{old} = \mathbf{x}_k$ and $\mathbf{v}_{old} = \mathbf{v}_k$.

5 Send a signal to all threads to interrupt all unfinished jobs at each thread.

6 **else**

7 */* Parallel Computation on muliple Threads: */*

8 If receiving interrupt signal at any time, go immediately to Step 9.

9 Read the parameter $\mathbf{x}_{old}$ and $\mathbf{v}_{old}$ from the shared memory. Set the most fresh parameter of x as $\mathbf{x}_{new}$.

10 Select a subset S of samples from N samples uniformly at random.

11 Compute $\mathbf{v}_{new} = \frac{1}{|S|}\sum_{i\in S}(\nabla f(\mathbf{x}_{new}, \xi_i) - \nabla f(\mathbf{x}_{old}, \xi_i) + \mathbf{v}_{old})$, which takes combined τ_k time-slots in computation and communication.

12 Let $\mathbf{x}_{old} = \mathbf{x}_{new}$ and $\mathbf{v}_{old} = \mathbf{v}_{new}$.

13 Select m_k from $\{1, ..., d\}$ uniformly at random; Update $(\mathbf{x}_{k+1})_{m_k} = (\mathbf{x}_k)_{m_k} - \eta(\mathbf{v}_{old})_{m_k}$.

5.1 Algorithm Description

Recall that the shared memory architecture usually models cases where a single machine with multiple cores/GPUs sharing the same memory. In the shared memory architecture, we have the parameter $\mathbf{x}$ stored in the shared memory space, and there are P threads that can access it. Each thread reads the freshest value of $\mathbf{x}$, denoted as $\mathbf{x}_{new}$, from the shared memory. Then, each thread randomly chooses any subset S of samples and computes a stochastic gradient $\frac{1}{|S|}\sum_{i\in S}(\nabla f(\mathbf{x}_{new}, \xi_i) - \nabla f(\mathbf{x}_{old}, \xi_i) + \mathbf{v}_{old})$ locally. All threads are allowed equal access to the shared memory and can update each individual component at will. Each thread updates its current parameters based on the asynchronous stochastic gradient information. Note that, while a single thread updates the parameters in the shared memory, the parameters may have been updated by other threads with their stochastic gradient information based on more recent $\mathbf{x}$-values. Hence, this update could be "delayed." To avoid race conditions in the shared memory, only a single coordinate of $\mathbf{x}$ can be updated at a time [22]. Hence, in each iteration, parameter $\mathbf{x}$ is updated as follows:

$$[\mathbf{x}_{k+1}]_{m_k} = [\mathbf{x}_k]_{m_k} - \eta[\mathbf{v}_k]_{m_k},$$

where $m_k \in \{1, 2, ..., d\}$ represents the updated coordinate in $\mathbf{x}$ in iteration k, η is a constant learning rate, $\mathbf{v}_k$ represents the variance-reduced gradient and can also be written as: $\mathbf{v}_k = \frac{1}{|S|}\sum_{i\in S}(\nabla f(\mathbf{x}_{k-\tau^k}, \xi_i) - \nabla f(\mathbf{x}_{k-\tau^k-1-\tau^{k-\tau^k-1}}, \xi_i) + \mathbf{v}_{k-1-\tau^k})$. We summarize the SA-SpiderBoost algorithm for the shared memory architecture in Algorithm 3. Note that, compared to the shared-memory SA-SpiderBoost algorithm, the algorithmic structure of Algorithm 3 is similar. The major differences are: i) there is no separation of server and worker codes due to the fact that only a single machine exists; and ii) only one coordinate can be updated at a time to avoid race conditions (cf. Step 13).

5.2 Convergence Analysis

In this subsection, we first present the main convergence result of SA-SpiderBoost under the shared memory architecture in Theorem 4. Due to space limitation and the similarity to the proof of Theorem 1, we omit the proof here and provide the full proof of Theorem 4 in Appendix C.

Theorem 4 (Convergence of SA-SpiderBoost for Shared Memory). *Let f^* denote the global optimal value. Under shared memory and the stated assumptions, for some $\epsilon > 0$, by choosing parameters $q = |S| = \sqrt{N}$ and $\eta = \frac{1}{2L(\Delta+1)}$ for a given maximum delay $\Delta > 0$, the SA-SpiderBoost algorithm outputs an $\mathbf{x}_\zeta$ that satisfies $\mathbb{E}\left[\left\|\nabla f(\mathbf{x}_\zeta)\right\|\right] \leq \epsilon$ if the total number of iterations K satisfies $K = O\big(\frac{f(\mathbf{x}_0)-f^*}{\epsilon^2}\big)$. This also implies that the total SFO complexity is $O(\sqrt{N}\epsilon^{-2}(\Delta+1)d + N)$.*

Remark 3. The computational complexity results in Theorem 4 has an extra d-factor compared to that in Theorem 1. This is because, under the shared memory architecture, one is allowed to read and write only a single coordinate of $\mathbf{x}$ at a time to avoid race conditions, i.e., the update rule of parameter $\mathbf{x}_k$ is $(\mathbf{x}_{k+1})_{m_k} = (\mathbf{x}_k)_{m_k} - \eta(\mathbf{v}_{old})_{m_k}$, where $m_k \in \{1, 2, ..., d\}$ is a randomly selected updated coordinate of $\mathbf{x}$ in iteration k. Thus, we have $\mathbb{E}\|(\mathbf{x}_{k+1})_{m_k} - (\mathbf{x}_k)_{m_k}\|^2 \leq \frac{1}{d}\mathbb{E}\|\eta(\mathbf{v}_{old})_{m_k}\|^2$ rather than $\mathbb{E}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2 = \mathbb{E}\|\eta\mathbf{v}_{old}\|^2$. This is the intuition behind the existence of an extra d-factor in the stated results.

5.3 Generalization Analysis for shared memory SA-SpiderBoost

In this section, our goal is to study the stability of the SA-SpiderBoost algorithm under the shared memory architecture. Recall that the update rule of SA-SpiderBoost for shared memory is given by $(\mathbf{x}_{k+1})_{m_k} = (\mathbf{x}_k)_{m_k} - \eta(\mathbf{v}_k)_{m_k} = (\mathbf{x}_k)_{m_k} - \eta\frac{1}{|S|}\sum_{i\in S}(\nabla f(\mathbf{x}_{k-\tau^k}, \xi_i)_{m_k} - \nabla f(\mathbf{x}_{k-\tau^k-1-\tau^{k-\tau^k-1}}, \xi_i)_{m_k} + (v_{k-1-\tau^k})_{m_k})$, where $m_k \in \{1, 2, ..., d\}$ is the updated coordinate in $\mathbf{x}$ in iteration k, and we run the update iteratively for a maximum number of K iterations. We consider two adjacent datasets $S = \{\xi_1, \xi_2, ..., \xi_N\}$ and $S' = (\xi_1', \xi_2', ..., \xi_N')$ from the same space, which differ in at most one element. For shared memory, we obtain the following main result on the algorithmic stability (or equivalently, generalization error) for SA-SpiderBoost:

Theorem 5 (Stability in the Quadratic Strongly Convex and Smooth Setting). *Suppose that the loss function $f(x; \xi_i)$ is a quadratic, L-smooth, μ -strongly convex, and M-Lipschitz. The shared-memory SA-SpiderBoost algorithm is $\left(\frac{2\eta M^2 K}{N\sqrt{d}}\right)$-uniformly stable.*

Theorem 6 (Stability in the Non-Convex and Smooth Setting). *Suppose that the loss function $f(x, \xi_i)$ is continuously differentiable, L-smooth and $M - Lipschitz$. The shared-memory SA-SpiderBoost algorithm is $\left(\frac{2\eta M^2 K}{\sqrt{d}} + \frac{2M^2 K\eta}{N\sqrt{d}}\right)$-uniformly stable.*

Remark 4. Theorems 5 and 6 provide algorithmic stability results for quadratic strongly-convex and non-convex settings, respectively. The proof techniques are similar to those of Theorem 2–3, and we omit the proofs here for brevity. We provide the full proofs of Theorem 5 and 6 in Appendix D. We can see that the stability error bounds in Theorems 5 and 6 have an extra $(1/\sqrt{d})$-factor compared to those in Theorems 2 and 3. This is because the shared memory architecture only allows one to read and write a single coordinate of $\mathbf{x}$ at a time to avoid race conditions, i.e., $\mathbf{x}_k$is $(\mathbf{x}_{k+1})_{m_k} = (\mathbf{x}_k)_{m_k} - \eta(\mathbf{v}_{old})_{m_k}$. Therefore, we have $\mathbb{E}\|\nabla f(\mathbf{x}'_{k-\tau^k}, \xi_i))_{m_k} - \nabla f(\mathbf{x}'_{k-\tau^k}, \xi_i'))_{m_k}\| \leq \frac{2M}{\sqrt{d}}$ rather than the relation $\mathbb{E}\|\nabla f(\mathbf{x}'_{k-\tau^k}, \xi_i) - \nabla f(\mathbf{x}'_{k-\tau^k}, \xi_i'))\| \leq 2M$ as in the distributed memory architecture, hence the existence of an extra $(1/\sqrt{d})$-factor.

6 NUMERICAL RESULTS

In this section, we conduct numerical experiments using several non-convex machine learning problems to verify our theoretical findings of the SA-SpiderBoost algorithms under distributed memory and shared memory architectures, respectively. In Section 6.1, we test the convergence performance of SA-SpiderBoost under the distributed memory architecture using the CIFAR-10 dataset. In Section 6.2, we test the convergence performance of SA-SpiderBoost under the shared

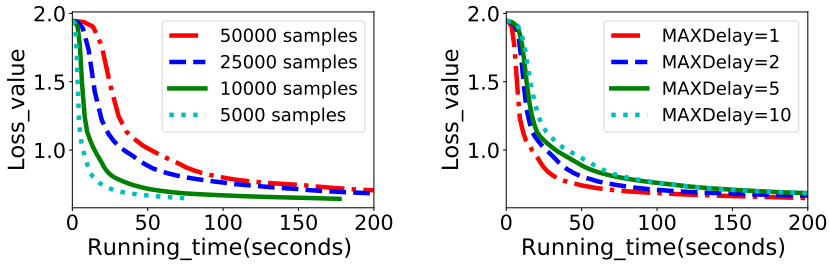


Fig. 4. Comparison of different number of samples and different maximum time delay on the CIFAR-10 dataset. It shows the convergence of training loss value with respect to time under distributed-memory SA-SpiderBoost algorithm.

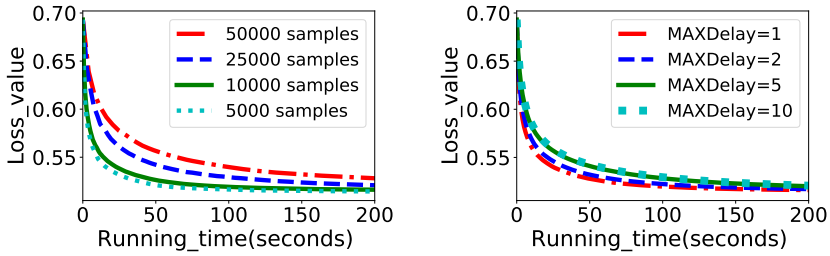
memory architecture using the MINIST dataset. In Section 6.3, we illustrate the generalization performance via simple logistic regression using the Breast-Cancer-Wisconsin dataset. In particular, we compare the generalization and convergence performances with two state-of-the-art algorithms in asynchronous first-order methods:

- *Async-SGD* [34]: This algorithm has a single-loop architecture, where each worker randomly picks one mini-batch from its local dataset at each iteration to compute a stochastic gradient and updates the parameter server in an asynchronous fashion. Here, the learning rate is chosen in such a way that, under certain delay distributions, it can accommodate potentially infinite delay.
- *Async-SVRG* [11]: This method has a double-loop architecture and employs unbiased gradient estimates. At the beginning of each outer loop iteration, a full pass over all N training samples is used to compute a full gradient. This full gradient is then used in its associated inner loop to adjust the stochastic gradients.

6.1 Convergence of SA-SpiderBoost under the Distributed Memory Architecture

Dataset and Learning Model: We run the SA-SpiderBoost algorithm under the distributed memory architecture. We conduct experiments on the Amazon Web Service (AWS) platform. There are four workers and one parameter server. We use a pre-trained convolutional neural network (CNN) model with the CIFAR-10 dataset [1]. Each sample is of dimension $d = 384$. We randomly select a subset of CIFAR-10 dataset with a varying size N as the training dataset. We fix the step-size as $\eta = 0.1$ and choose mini-batch size $\lceil\sqrt{N}\rceil$ according to our theoretical results. Then, we construct a fully connected three-layer $(384 \times 100 \times 10)$ neural network with the ReLU activation function and the Softmax loss function.

Numerical Results: We test the convergence of the training loss value of the SA-SpiderBoost algorithm with different training dataset size N and different values of maximum delay Δ on each worker. Figs. 4(a) and 4(b) illustrate our experiment results. It can be seen that as N or Δ decreases, the convergence rate increases, which is consistent with our theoretical algorithm complexity $O(\sqrt{N}\epsilon^{-2}\Delta + N)$. Note that the above algorithm complexity result implies an $O(1/K)$ convergence rate, which can also be observed in Fig. 4. This behavior makes intuitive sense because a larger number of training samples implies a heavier computation load. Also, a larger maximum delay value results in more stale stochastic gradient information, and thus inducing a negative impact on the convergence rate.



(a) (b)

Fig. 5. Comparison of different number of samples and different maximum time delay on the MNIST dataset. It illustrates the convergence of training loss value with respect to time under the shared-memory SA-SpiderBoost algorithm.

6.2 Convergence of SA-SpiderBoost under the Shared Memory Architecture

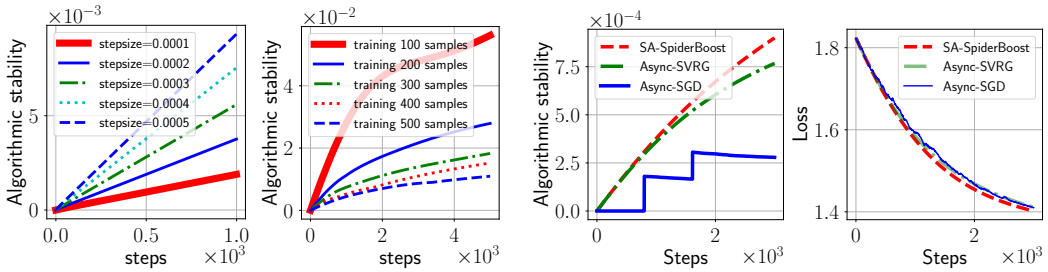
Dataset and Learning Model: We run the SA-SpiderBoost algorithm under the shared memory architecture. We conduct experiments on a single machine with multiple cores. We leverage the openMPI (Message Passing Interface) to facilitate the shared memory parallelism. There are four threads in the platform working in this experiment. We randomly select a subset of the MNIST dataset with a varying size N as the training dataset (each sample is a vector of dimension 784). We fix the step-size $\eta = 0.1$ and choose a mini-batch size $q = \lceil \sqrt{N} \rceil$. We build a fully connected three-layer$(784 \times 100 \times 10)$ neural network with ReLU activation and Softmax loss.

Numerical Results: Figs. 5(a) and 5(b) illustrate the convergence performance of training loss value with respect to the running time of the SA-SpiderBoost algorithm under the shared memory architecture with different N and Δ values. According to our theoretical results of the shared-memory SA-SpiderBoost algorithm, the larger the value of N or Δ, the slower the convergence. From Figs. 5(a) and 5(b), we also observe same behaviors, which verify our theoretical findings.

6.3 Generalization Performance

Dataset and Learning Model: We present experimental results for SA-SpiderBoost using the logistic regression problem for binary classification based on the Breast-Cancer-Wisconsin dataset [31] (there are $N = 699$ samples and each sample is of dimension 10). Specifically, consider a set of independent and identically distributed (i.i.d.) samples $\{\mathbf{a}_i, \mathbf{b}_i\}_{i=1}^N$, $\mathbf{a}_i \in \mathbb{R}^d$, $b_i \in \{0, 1\}$. Let $\mathbf{x}$ be the parameter we want to estimate. The ℓ_2-regularized log-likelihood logistic regression problem can be written as $f(\mathbf{x}) = \frac{1}{N}\big(-\mathbf{b}^\top \mathbf{A}\mathbf{x} + \sum_{i=1}^N \log(1 + e^{\mathbf{x}^\top \mathbf{a}_i}) + \frac{1}{2}\|\mathbf{x}\|^2\big)$, where matrix $\mathbf{A} \in \mathbb{R}^{N\times d}$ contains $\mathbf{a}_i$ as the i-th row, $\forall i$, and $\mathbf{b} = [b_1, b_2, \ldots, b_N]^\top$. Based on the definition of algorithmic stability, we need datasets that differ only by one sample. To do so, we first choose a subset S of samples from the entire dataset, then construct a perturbed subset S' by replacing the last sample in S with a randomly selected sample from the whole dataset. Finally, we run our optimization algorithms to compute the normalized Euclidean distance between $\mathbf{x}_k$ and $\mathbf{x}'_k$. Each point in our figures is averaged over 50 trials of random perturbations.

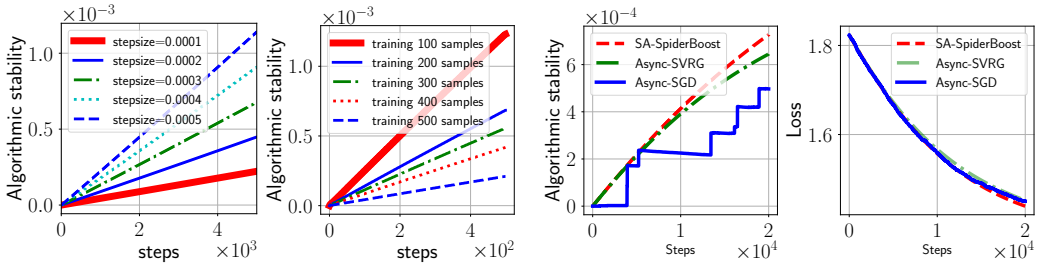
Numerical Results: Figs. 6 and 7 show the stability performance of the SA-SpiderBoost algorithm under distributed memory and shared memory architectures with different step-sizes and numbers of training samples, respectively. We observe that the stability error increases with the step-size η and number of training samples N. Also, we can observe that the stability error grows roughly



(a) SA-SpiderBoost stability with different step-size and different number of training samples.

(b) Algorithmic stability and loss value comparison of three methods (Async-SGD, Async-SVRG, SA-SpiderBoost).

Fig. 6. Algorithm comparison under distributed memory system.



(a) SA-SpiderBoost stability with different stepsize and different number of training samples.

(b) Algorithmic stability and loss value comparison of three methods (Async-SGD, Async-SVRG, SA-SpiderBoost).

Fig. 7. Algorithm comparison under shared memory system.

linearly with step-size η and is roughly inversely proportional to sample size N, which suggests that our theoretical stability upper bound $(2\eta M^2K^2 + \frac{2M^2K^2\eta}{N})$ in Theorem 3 is tight.

We plot and compare the performances of Async-SGD, Async-SVRG, and SA-SpiderBoost in Figs. 6(b) and 7(b). We choose the same starting points for all algorithms, which are generated randomly from a normal distribution. We choose a constant step-sizes 0.0001, training sample size $N = 500$ and mini-batch S with $|S| = \lceil\sqrt{N}\rceil$. The results show that Async-SGD has the smallest bound on the algorithmic stability, implying better generalization performance. This is consistent with the widely observed robustness of SGD (see, e.g., [3]). Our results also show that SA-SpiderBoost has a faster convergence rate at the expense of a slightly worse stability performance (an approximate 10^{-4} gap with 10^3 iterations). From Figs. 6(b) and 7(b), it can be seen that larger step-sizes and fewer training data samples lead to faster convergence rate and worse algorithmic stability. This indicates a fundamental trade-off between convergence and generalization. The overall performance of SA-SpiderBoost under the distributed memory architecture is better than that under the shared memory architecture, which again confirms our theoretical findings.

7 CONCLUSION

In this paper, we proposed and analyzed the semi-asynchronous SpiderBoost (SA-SpiderBoost) algorithm for non-convex learning under distributed memory and shared memory architectures. We showed that our SA-SpiderBoost algorithm achieves $O(\sqrt{N}\epsilon^{-2}\Delta + N)$ and $O(\sqrt{N}\epsilon^{-2}\Delta d + N)$ computational complexities under these two architectures, respectively. We also provided the

stability error upper bounds for the SA-SpiderBoost algorithm under distributed memory and shared memory architectures. We showed that smaller step-size and more training samples improve the algorithmic stability, while larger maximum delays in asynchronous updates have a negative impact on the convergence performance. We also revealed a fundamental trade-off between convergence and generalization: a faster converging performance under both memory architectures implies a worse algorithmic stability, and hence poorer generalization. Collectively, our results in this work advance the understanding of the convergence and generalization performances of asynchronous distributed first-order variance-reduced methods.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

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

[3] Navid Azizan, Sahin Lale, and Babak Hassibi. Stochastic mirror descent on overparameterized nonlinear models: Convergence, implicit regularization, and generalization. *arXiv:1906.03830*, 2019.

[4] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

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

[6] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pages 1646–1654, 2014.

[7] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*, pages 689–699, 2018.

[8] Abraham P George and Warren B Powell. Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Machine learning*, 65(1):167–198, 2006.

[9] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[10] Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pages 1225–1234, 2016.

[11] Zhouyuan Huo and Heng Huang. Asynchronous stochastic gradient descent with variance reduction for non-convex optimization. *arXiv preprint arXiv:1604.03584*, 2016.

[12] Zhouyuan Huo and Heng Huang. Asynchronous mini-batch gradient descent with variance reduction for non-convex optimization. In *AAAI*, pages 2043–2049, 2017.

[13] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

[14] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

[15] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[16] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *OSDI*, volume 1, page 3, 2014.

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

[18] Wenlong Mou, Liwei Wang, Xiyu Zhai, and Kai Zheng. Generalization bounds of sgld for non-convex learning: Two theoretical viewpoints. In *Conference on Learning Theory*, pages 605–638, 2018.

[19] Behnam Neyshabur, Russ R Salakhutdinov, and Nati Srebro. Path-sgd: Path-normalized optimization in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2422–2430, 2015.

[20] Zhimin Peng, Yangyang Xu, Ming Yan, and Wotao Yin. Arock: an algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing*, 38(5):A2851–A2879, 2016.

[21] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

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

[23] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alexander J Smola. On variance reduction in stochastic gradient descent and its asynchronous variants. In *Advances in Neural Information Processing Systems*, pages 2647–2655, 2015.

[24] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[25] Nicolas L Roux, Mark Schmidt, and Francis R Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in neural information processing systems*, pages 2663–2671, 2012.

[26] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

[27] Lei Tang, Patrick Harrington, and Tao Zhu. Providing personalized item recommendations using scalable matrix factorization with randomness, February 19 2015. US Patent App. 13/970,271.

[28] Shirin Tavara. Parallel computing of support vector machines: a survey. *ACM Computing Surveys (CSUR)*, 51(6):1–38, 2019.

[29] Zhe Wang, Kaiyi Ji, Yi Zhou, Yingbin Liang, and Vahid Tarokh. Spiderboost: A class of faster variance-reduced algorithms for nonconvex optimization. *arXiv preprint arXiv:1810.10690*, 2018.

[30] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems*, pages 1509–1519, 2017.

[31] William H Wolberg and Olvi L Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the national academy of sciences*, 87(23):9193–9196, 1990.

[32] Quanming Yao and James Kwok. Scalable robust matrix factorization with nonconvex loss. In *Advances in Neural Information Processing Systems*, pages 5061–5070, 2018.

[33] Ruiliang Zhang and James Kwok. Asynchronous distributed admm for consensus optimization. In *International conference on machine learning*, pages 1701–1709, 2014.

[34] Xin Zhang, Jia Liu, and Zhengyuan Zhu. Taming convergence for asynchronous stochastic gradient descent with unbounded delay in non-convex learning. In *Proc. IEEE CDC*, 2020.

[35] Shen-Yi Zhao and Wu-Jun Li. Fast asynchronous parallel stochastic gradient descent: A lock-free approach with convergence guarantee. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[36] Shuxin Zheng, Qi Meng, Taifeng Wang, Wei Chen, Nenghai Yu, Zhi-Ming Ma, and Tie-Yan Liu. Asynchronous stochastic gradient descent with delay compensation. In *International Conference on Machine Learning*, pages 4120–4129, 2017.

[37] Yi Zhou, Yingbin Liang, and Huishuai Zhang. Generalization error bounds with probabilistic guarantee for sgd in nonconvex optimization. *arXiv preprint arXiv:1802.06903*, 2018.

A CONVERGENCE ANALYSIS OF SA-SPIDERBOOST FOR DISTRIBUTED MEMORY

To prove the stated in Theorem 1, we first prove a useful lemma.

Lemma 2. *Let all assumptions hold and apply SA-SpiderBoost in Algorithm 1 and Algorithm 2, if the parameters η, q and S are chosen such that*

$$\beta_1 \triangleq \frac{\eta}{2} - \frac{L\eta^2}{2} - L^2\eta^3\big(\frac{q(\Delta+1)}{|S|} + \Delta^2\big) > 0, \tag{25}$$

and if for $mod(k,q) = 0$,we always have

$$\mathbb{E}\big\|\mathbf{v}_k - \nabla f(\mathbf{x}_k)\big\|^2 \leq \epsilon_1^2, \tag{26}$$

then the output point $\mathbf{x}_\zeta$ of SA-SpiderBoost satisfies

$$\mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta)\big\|^2 \leq \big[\frac{2}{\beta_1}(\eta + 2L^2(\Delta^2 + \frac{q(\Delta+1)}{|S|})\eta^3) + 4\big]\epsilon_1^2 + \big[\frac{4L^2(\Delta^2 + \frac{q(\Delta+1)}{|S|})\eta^2}{K\beta_1} + \frac{2}{K\beta_1}\big](f(\mathbf{x}_0) - f^*). \tag{27}$$

A.1 Proof of Lemma 2

Proof. In Lemma 2, we aim to bound $\mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta)\big\|^2$, where ζ is chosen from $\{1, ..., K\}$ uniformly at random. Following the inequality of arithmetic and geometric means, we have:

$$\mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta)\big\|^2 = \mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta) - \mathbf{v}_\zeta + \mathbf{v}_\zeta\big\|^2 \leq 2\mathbb{E}\big\|\mathbf{v}_\zeta\big\|^2 + 2\mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta) - \mathbf{v}_\zeta\big\|^2. \tag{28}$$

Next, we bound the terms $\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2$ and $\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta) - \mathbf{v}_\zeta\right\|^2$ on the right-hand-side of (28) individually.

To evaluate $\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta) - \mathbf{v}_\zeta\right\|^2$, we start from the iteration relationship of our SA-SpiderBoost algorithm, for which we have:

$$\begin{aligned}
&\mathbb{E}\big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_k, \xi_i)\big\|^2 \\
&\overset{(a)}{\leq} 2\mathbb{E}\big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau^k}, \xi_i)\big\|^2 + 2\mathbb{E}\big\|\frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau^k}, \xi_i) - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_k, \xi_i)\big\|^2 \\
&\overset{(b)}{\leq} 2\mathbb{E}\big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau^k}, \xi_i)\big\|^2 + 2L^2\mathbb{E}\big\|\mathbf{x}_{k-\tau^k} - \mathbf{x}_k\big\|^2 \\
&= 2\mathbb{E}\big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau^k}, \xi_i)\big\|^2 + 2L^2\mathbb{E}\big\|\sum_{j=k-\tau^k}^{k-1}(\mathbf{x}_{j+1} - \mathbf{x}_j)\big\|^2 \\
&\overset{(c)}{\leq} 2\mathbb{E}\big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau^k}, \xi_i)\big\|^2 + 2L^2\Delta\sum_{j=k-\tau^k}^{k-1}\mathbb{E}\big\|(\mathbf{x}_{j+1} - \mathbf{x}_j)\big\|^2 \\
&\overset{(d)}{=} 2\mathbb{E}\big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau^k}, \xi_i)\big\|^2 + 2L^2\eta^2\Delta\sum_{j=k-\tau^k}^{k-1}\mathbb{E}\big\|\mathbf{v}_j\big\|^2,
\end{aligned} \tag{29}$$

where (a) follows from the triangle inequality, (b) follows from L-smooth property, (c) is due to the triangle inequality and the maximum delay is Δ, and (d) uses the condition of update rule.

Next, we bound the terms $\mathbb{E}\big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau^k}, \xi_i)\big\|^2$ on the right-hand-side of (29), which denotes the distance of the inner loop. Let $n_k = \lceil k/q \rceil$ such that $(n_k - 1)q \leq k \leq n_k q - 1$, we have

$$
\begin{aligned}
&\mathbb{E}\Big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau^k},\xi_i)\Big\|^2 \\
&\overset{(a)}{=}\mathbb{E}\Big\|\frac{1}{|S|}\sum_{i\in S}\Big[\nabla f(\mathbf{x}_{k-\tau^k},\xi_i) - \nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i) - \frac{1}{N}\sum_{j=1}^{N}\nabla f(\mathbf{x}_{k-\tau^k},\xi_j) \\
&\quad + \frac{1}{N}\sum_{j=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_j)\Big]\Big\|^2 + \mathbb{E}\Big\|\frac{1}{|S|}\sum_{i\in S}\Big[\mathbf{v}_{k-1-\tau^k} - \frac{1}{N}\sum_{j=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_j)\Big]\Big\|^2 \\
&\overset{(b)}{\leq}\frac{1}{|S|^2}\sum_{i\in S}\mathbb{E}\Big\|\big[\nabla f(\mathbf{x}_{k-\tau^k},\xi_i) - \nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i)\big] - \frac{1}{N}\sum_{j=1}^{N}\nabla f(\mathbf{x}_{k-\tau^k},\xi_j) \\
&\quad + \frac{1}{N}\sum_{j=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_j)\Big\|^2 + \mathbb{E}\Big\|\mathbf{v}_{k-1-\tau^k} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i)\Big\|^2 \\
&\overset{(c)}{\leq}\frac{1}{|S|^2}\sum_{i\in S}\mathbb{E}\Big\|\nabla f(\mathbf{x}_{k-\tau^k},\xi_i) - \nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i)\Big\|^2 \\
&\quad + \mathbb{E}\Big\|\mathbf{v}_{k-1-\tau^k} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i)\Big\|^2 \\
&\overset{(d)}{\leq}\frac{L^2}{|S|^2}\sum_{i\in S}\mathbb{E}\Big\|\mathbf{x}_{k-\tau^k} - \mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}}\Big\|^2 + \mathbb{E}\Big\|\mathbf{v}_{k-1-\tau^k} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i)\Big\|^2 \\
&\overset{(e)}{=}\frac{L^2\eta^2}{|S|}\mathbb{E}\Big\|\sum_{i=k-\tau^k-1-\tau^{k-\tau^k-1}}^{k-\tau^k-1}\mathbf{v}_i\Big\|^2 + \mathbb{E}\Big\|\mathbf{v}_{k-1-\tau^k} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i)\Big\|^2 \\
&\overset{(f)}{\leq}\frac{L^2\eta^2}{|S|}(\Delta+1)\sum_{i=k-\tau^k-1-\tau^{k-\tau^k-1}}^{k-\tau^k-1}\mathbb{E}\big\|\mathbf{v}_i\big\|^2 + \mathbb{E}\Big\|\mathbf{v}_{k-1-\tau^k} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i)\Big\|^2, \qquad (30)
\end{aligned}
$$

where (a) follows from the gradient update rule

$$
\mathbf{v}_k = \frac{1}{|S|}\sum_{i\in S}(\nabla f(\mathbf{x}_{k-\tau^k},\xi_i) - \nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i) + \mathbf{v}_{k-1-\tau^k}),
$$

and Lemma 1 in [7], (b) and (c) use in [7, Appendix A.3], (d) follows from the Lipschitz continuity of $\nabla f(\mathbf{x})$, (e) is due to the condition on update rule, and (f) follows from the maximum delay is Δ.

Since $\mathbf{v}_{k-1-\tau^k}$ are generated from the previous step. Telescoping q iterations, we obtain that:

$$
\begin{aligned}
\mathbb{E}\Big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau^k},\xi_i)\Big\|^2 \leq &\frac{L^2\eta^2(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k-1-\tau^k}\mathbb{E}\big\|\mathbf{v}_j\big\|^2 \\
&+ \mathbb{E}\Big\|\mathbf{v}_{(n_k-1)q} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{(n_k-1)q},\xi_i)\Big\|^2. \qquad (31)
\end{aligned}
$$

By combining (29) and (31), we arrive at:

$$
\begin{aligned}
&\mathbb{E}\Big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_k,\xi_i)\Big\|^2 \\
&\leq \frac{2L^2\eta^2(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k-1-\tau^k}\mathbb{E}\big\|\mathbf{v}_j\big\|^2 + 2\mathbb{E}\Big\|\mathbf{v}_{(n_k-1)q} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{(n_k-1)q},\xi_i)\Big\|^2 \\
&+ 2L^2\eta^2\Delta\sum_{j=k-\tau^k}^{k-1}\mathbb{E}\big\|\mathbf{v}_j\big\|^2. \qquad (32)
\end{aligned}
$$

Next, we continue to bound the other term $\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2$ on the right-hand-side of (28). By Assumption 6, the entire objective function f is $L-$smooth, which further implies

$$
\begin{aligned}
f(\mathbf{x}_{k+1}) &\leq f(\mathbf{x}_k) + \langle\nabla f(\mathbf{x}_k), \mathbf{x}_{k+1}-\mathbf{x}_k\rangle + \frac{L}{2}\big\|\mathbf{x}_{k+1}-\mathbf{x}_k\big\|^2 \\
&\overset{(a)}{=} f(\mathbf{x}_k) - \eta\langle\nabla f(\mathbf{x}_k), \mathbf{v}_k\rangle + \frac{L\eta^2}{2}\big\|\mathbf{v}_k\big\|^2 \\
&= f(\mathbf{x}_k) - \eta\langle\nabla f(\mathbf{x}_k) - \mathbf{v}_k, \mathbf{v}_k\rangle - \eta\big\|\mathbf{v}_k\big\|^2 + \frac{L\eta^2}{2}\big\|\mathbf{v}_k\big\|^2 \\
&\overset{(b)}{\leq} f(\mathbf{x}_k) + \frac{\eta}{2}\big\|\mathbf{v}_k - \nabla f(\mathbf{x}_k)\big\|^2 - (\frac{\eta}{2} - \frac{L\eta^2}{2})\big\|\mathbf{v}_k\big\|^2, \qquad (33)
\end{aligned}
$$

where (a) follows from the update rule of SA-Spiderboost, (b) uses uses the inequality that $\langle\mathbf{x},\mathbf{y}\rangle \leq \frac{\|\mathbf{x}\|^2+\|\mathbf{y}\|^2}{2}$, for $\mathbf{x},\mathbf{y}\in\mathbb{R}^d$. Thus, we have

$$
\begin{aligned}
&\mathbb{E}f(\mathbf{x}_{k+1}) \leq \mathbb{E}f(\mathbf{x}_k) + \frac{\eta}{2}\mathbb{E}\big\|\mathbf{v}_k - \nabla f(\mathbf{x}_k)\big\|^2 - (\frac{\eta}{2} - \frac{L\eta^2}{2})\mathbb{E}\big\|\mathbf{v}_k\big\|^2 \\
&\overset{(a)}{\leq} \frac{\eta}{2}\Big(\frac{2L^2\eta^2(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k-1-\tau^k}\mathbb{E}\big\|\mathbf{v}_j\big\|^2 + 2\mathbb{E}\Big\|\mathbf{v}_{(n_k-1)q} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{(n_k-1)q},\xi_i)\Big\|^2 \\
&\quad + 2L^2\eta^2\Delta\sum_{j=k-\tau^k}^{k-1}\mathbb{E}\big\|\mathbf{v}_j\big\|^2\Big) - (\frac{\eta}{2} - \frac{L\eta^2}{2})\mathbb{E}\big\|\mathbf{v}_k\big\|^2 + \mathbb{E}f(\mathbf{x}_k) \\
&\overset{(b)}{\leq} \mathbb{E}f(\mathbf{x}_k) - (\frac{\eta}{2} - \frac{L\eta^2}{2})\mathbb{E}\big\|\mathbf{v}_k\big\|^2 + \eta\epsilon_1^2 + \frac{L^2\eta^3(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k-1-\tau^k}\mathbb{E}\big\|\mathbf{v}_j\big\|^2 + L^2\eta^3\Delta\sum_{j=k-\tau^k}^{k-1}\mathbb{E}\big\|\mathbf{v}_j\big\|^2 \\
&\leq \mathbb{E}f(\mathbf{x}_k) - (\frac{\eta}{2} - \frac{L\eta^2}{2})\mathbb{E}\big\|\mathbf{v}_k\big\|^2 + \eta\epsilon_1^2 + \frac{L^2\eta^3(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k}\mathbb{E}\big\|\mathbf{v}_j\big\|^2 + L^2\eta^3\Delta\sum_{j=k-\tau^k}^{k-1}\mathbb{E}\big\|\mathbf{v}_j\big\|^2, \qquad (34)
\end{aligned}
$$

where (a) follows from (29), (b) follows from the $\mathbb{E}\big\|\mathbf{v}_{(n_k-1)q} - \nabla f(\mathbf{x}_{(n_k-1)q})\big\|^2 \leq \epsilon_1^2$. Next, telescoping (34) over k from $(n_k-1)q$ to k, where $k \leq n_k q - 1$, we have:

$$
\mathbb{E}f(\mathbf{x}_{k+1}) \overset{(a)}{\leq} \mathbb{E}f(\mathbf{x}_{(n_k-1)q}) - (\frac{\eta}{2} - \frac{L\eta^2}{2})\sum_{i=(n_k-1)q}^{k}\mathbb{E}\big\|\mathbf{v}_i\big\|^2 + \sum_{i=(n_k-1)q}^{k}\eta\epsilon_1^2
$$

$$+\frac{L^2\eta^3(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k}\sum_{i=(n_k-1)q}^{j}\mathbb{E}\left\|\mathbf{v}_i\right\|^2+L^2\eta^3\Delta\sum_{j=(n_k-1)q}^{k}\sum_{i=j-\tau^j}^{j-1}\mathbb{E}\left\|\mathbf{v}_i\right\|^2$$

$$\overset{(b)}{\leq}\mathbb{E}f(\mathbf{x}_{(n_k-1)q})-(\frac{\eta}{2}-\frac{L\eta^2}{2})\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2+\sum_{i=(n_k-1)q}^{k}\eta\epsilon_1^2$$

$$+\frac{L^2\eta^3(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k}\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2+L^2\eta^3\Delta^2\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2, \tag{35}$$

where (a) follows from (34), (b) extends the summation of second term from j to k. Further relaxing (35) yields:

$$\begin{aligned}\mathbb{E}f(\mathbf{x}_{k+1})\overset{(a)}{\leq}&\mathbb{E}f(\mathbf{x}_{(n_k-1)q})-(\frac{\eta}{2}-\frac{L\eta^2}{2})\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2+\sum_{i=(n_k-1)q}^{k}\eta\epsilon_1^2\\&+\frac{qL^2\eta^3(\Delta+1)}{|S|}\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2+L^2\eta^3\Delta^2\sum_{j=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2\\=&\mathbb{E}f(\mathbf{x}_{(n_k-1)q})+\sum_{i=(n_k-1)q}^{k}\eta\epsilon_1^2-\big[\frac{\eta}{2}-\frac{L\eta^2}{2}-L^2\eta^3\big(\frac{q(\Delta+1)}{|S|}+\Delta^2\big)\big]\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2\\=&\mathbb{E}[f(\mathbf{x}_{(n_k-1)q})]-\sum_{i=(n_k-1)q}^{k}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2-\eta\epsilon_1^2),\end{aligned} \tag{36}$$

where (a) follows from the fact that $k\leq n_kq-1$. Then, by telescoping, we can further derive:

$$\begin{aligned}\mathbb{E}f(\mathbf{x}_K)-\mathbb{E}f(\mathbf{x}_0)=&(\mathbb{E}f(\mathbf{x}_q)-\mathbb{E}f(\mathbf{x}_0))+(\mathbb{E}f(\mathbf{x}_{2q})-\mathbb{E}f(\mathbf{x}_q)+\ldots+(\mathbb{E}f(\mathbf{x}_K)-\mathbb{E}f(\mathbf{x}_{(n_k-1)q})))\\\overset{(a)}{\leq}&-\sum_{i=0}^{q-1}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2-\eta\epsilon_1^2)-\sum_{i=q}^{2q-1}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2-\eta2\epsilon_1^2)-\ldots\\&-\sum_{(n_K-1)q}^{K-1}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2-\frac{\eta}{2}\epsilon_1^2)\\=&-\sum_{i=0}^{K-1}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2-\eta\epsilon_1^2)\\=&-\sum_{i=0}^{K-1}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2)+K\eta\epsilon_1^2,\end{aligned} \tag{37}$$

where (a) follows from (36). Since $\mathbb{E}f(\mathbf{x}_K)\geq f(x^*)$, then we have:

$$\mathbb{E}f(\mathbf{x}^*)-\mathbb{E}f(\mathbf{x}_0)\leq-\sum_{i=0}^{K-1}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2)+K\eta\epsilon_1^2. \tag{38}$$

By rearranging (38), we have:

$$\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2=\frac{1}{K}\sum_{i=0}^{K-1}\mathbb{E}\left\|\mathbf{v}_i\right\|^2\leq\frac{f(\mathbf{x}_0)-f^*}{K\beta_1}+\frac{\eta}{\beta_1}\epsilon_1^2, \tag{39}$$

It then follows that:

$$\begin{aligned}
\mathbb{E}\left\|\mathbf{v}_\zeta - \nabla f(\mathbf{x}_\zeta)\right\|^2 &\overset{(a)}{\leq} \mathbb{E}\frac{2L^2\eta^2(\Delta+1)}{|S|}\sum_{j=(n_\zeta-1)q}^{\zeta-1-\tau^\zeta}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 + 2\mathbb{E}\left\|\mathbf{v}_{(n_\zeta-1)q} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{(n_\zeta-1)q}, \xi_i)\right\|^2 \\
&\quad + 2L^2\eta^2\Delta\mathbb{E}\sum_{j=\zeta-\tau^\zeta}^{\zeta-1}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 \\
&\overset{(b)}{\leq} \frac{2L^2\eta^2(\Delta+1)}{|S|}\mathbb{E}\sum_{j=(n_\zeta-1)q}^{\zeta-1-\tau^\zeta}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 + 2\epsilon_1^2 + 2L^2\eta^2\Delta\mathbb{E}\sum_{i=\zeta-\tau^\zeta}^{\zeta-1}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 \\
&\leq \frac{2L^2\eta^2(\Delta+1)}{|S|}\mathbb{E}\sum_{j=(n_\zeta-1)q}^{\xi}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 + 2\epsilon_1^2 + 2L^2\eta^2\Delta\mathbb{E}\sum_{i=\zeta-\tau^\zeta}^{\zeta-1}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 \\
&\overset{(c)}{\leq} \frac{2L^2\eta^2(\Delta+1)}{|S|}\mathbb{E}\sum_{i=(n_\zeta-1)q}^{min\{(n_\zeta)q-1,K-1\}}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + 2\epsilon_1^2 + 2L^2\eta^2\Delta\mathbb{E}\sum_{i=\zeta-\tau^\zeta}^{\zeta-1}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 \\
&\overset{(d)}{\leq} \frac{q}{K}\sum_{i=0}^{K-1}\frac{2L^2\eta^2(\Delta+1)}{|S|}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + 2\epsilon_1^2 + \frac{\Delta}{K}\sum_{i=0}^{K-1}2L^2\eta^2\Delta\mathbb{E}\left\|\mathbf{v}_i\right\|^2 \\
&= \left(\frac{2L^2\eta^2q(\Delta+1)}{|S|} + 2L^2\eta^2\Delta^2\right)\frac{1}{K}\sum_{i=0}^{K-1}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + 2\epsilon_1^2 \\
&\overset{(e)}{\leq} \Big[\frac{2L^2(\Delta^2 + \frac{q(\Delta+1)}{|S|})\eta^3}{\beta_1} + 2\Big]\epsilon_1^2 + 2L^2(\Delta^2 + \frac{q(\Delta+1)}{|S|})\eta^2\Big[\frac{f(\mathbf{x}_0) - f^*}{K\beta_1}\Big], \qquad (40)
\end{aligned}$$

where (a) follows from (32), (b) follows from (26), (c) follows from the definition of n_ζ, which implies $\zeta \leq min\{(n_\zeta)q - 1, K-1\}$, (d)follows from the fact that the probability that $n_\zeta = 1, 2, ..., n_K$ is less than or equal to $\frac{q}{K}$ and (e) follows from (39).

By combining (39) and (40), we arrive at:

$$\begin{aligned}
&\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|^2 \leq 2\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta) - \mathbf{v}_\zeta\right\|^2 + 2\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2 \\
&\leq 2\Big\{\Big[\frac{2L^2(\Delta^2 + \frac{q(\Delta+1)}{|S|})\eta^3}{\beta_1} + 2\Big]\epsilon_1^2 + 2L^2(\Delta^2 + \frac{q(\Delta+1)}{|S|})\eta^2\Big[\frac{f(\mathbf{x}_0) - f^*}{K\beta_1}\Big]\Big\} \\
&\quad + 2\Big[\frac{f(\mathbf{x}_0) - f^*}{K\beta_1} + \frac{\eta}{\beta_1}\epsilon_1^2\Big] \\
&= \Big[\frac{2}{\beta_1}(\eta + 2L^2(\Delta^2 + \frac{q(\Delta+1)}{|S|})\eta^3) + 4\Big]\epsilon_1^2 + \Big[\frac{4L^2(\Delta^2 + \frac{q(\Delta+1)}{|S|})\eta^2}{K\beta_1} + \frac{2}{K\beta_1}\Big](f(\mathbf{x}_0) - f^*). \qquad (41)
\end{aligned}$$

This completes the proof of Lemma 2. □

With Lemma 2, we are in a position to prove the result in Theorem 1. Setting the parameters

$$q = \sqrt{N}, \quad S = \sqrt{N}, \quad \eta = \frac{1}{4L(\Delta+1)}, \qquad (42)$$

we obtain:

$$\beta_1 \triangleq \frac{\eta}{2} - \frac{L\eta^2}{2} - L^2\eta^3\Big(\frac{q(\Delta+1)}{|S|} + \Delta^2\Big) = \frac{1}{8L(\Delta+1)} \cdot \frac{14\Delta^2 + 26\Delta + 10}{16(\Delta+1)^2} > 0. \qquad (43)$$

For $mod(k,q)=0$ we have $\mathbb{E}\left\|\mathbf{v}_k-\nabla f(\mathbf{x}_k)\right\|^2=0$. Then, after K iterations, we have

$$\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|^2 \le 16L(\Delta+1)\frac{9\Delta^2+17\Delta+9}{K\cdot(7\Delta^2+13\Delta+5)}(f(\mathbf{x}_0)-f^*). \tag{44}$$

To ensure $\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|\le\epsilon$, it suffices to ensure$\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|^2\le\epsilon^2$. Solving for K yields:

$$K=16L(\Delta+1)\frac{9\Delta^2+17\Delta+9}{\epsilon^2\cdot(7\Delta^2+13\Delta+5)}(f(\mathbf{x}_0)-f^*)=O\big(\frac{f(\mathbf{x}_0)-f^*}{\epsilon^2}\big).$$

Lastly, to show the SFO complexity, we note that the number of SFO calls in the outer loops can be calculated as: $\lceil\frac{K}{q}\rceil N$. Also, the number of SFO calls in the inner loop can be calculated as KS. Hence, the total SFO complexity can be calculated as: $\lceil\frac{K}{q}\rceil N+K\cdot S\le\frac{K+q}{q}N+K\sqrt{N}=K\sqrt{N}+N+K\sqrt{N}=O(\sqrt{N}\epsilon^{-2}(\Delta+1)+N)$. This completes the proof.

B GENERALIZATION ANALYSIS OF SA-SPIDERBOOST FOR DISTRIBUTED MEMORY

B.1 Proofs of Theorem 2

Recall that update rule for SA-SpiderBoost for shared-memory system is given by:

$$\mathbf{x}_{k+1}=\mathbf{x}_k-\eta\frac{1}{|S|}\sum_{i\in S}(\nabla f(\mathbf{x}_{k-\tau^k},\xi_i)-\nabla f(\mathbf{x}_{k-\tau^k-1-\tau^{k-\tau^k-1}},\xi_i)+\mathbf{v}_{k-1-\tau^k}). \tag{45}$$

S and S' are two data sets such that S and S' differ in at most one example, where $S=(\xi_1,\xi_2,...,\xi_N)$ and $S'=(\xi_1',\xi_2',...,\xi_N')$. Let $\delta_k\triangleq\mathbf{x}_k-\mathbf{x}_k'$. Suppose $\mathbf{x}_0=\mathbf{x}_0'$.

Now, taking expectation of δ_{k+1} with respect of the algorithm, we get

$$\mathbb{E}(\delta_{k+1})=\mathbb{E}(\mathbf{x}_{k+1})-\mathbb{E}(\mathbf{x}_{k+1}')=\mathbb{E}(\delta_k)-\eta[\mathbb{E}(\mathbf{v}_k)-\mathbb{E}(\mathbf{v}_k')]. \tag{46}$$

Since we $\mathbf{v}_k$ is the unbiased estimated of $\nabla f(\mathbf{x}_{k-\tau^k})$, we have

$$\mathbb{E}(\delta_{k+1})=\mathbb{E}(\delta_k)-\eta[\mathbb{E}(\nabla f(\mathbf{x}_{k-\tau^k}))-\mathbb{E}(\nabla f(\mathbf{x}_{k-\tau^k}'))]. \tag{47}$$

At Step k, with probability $1-1/N$, the example is the same in S and S'. With probability $\frac{1}{N}$, the example is different in S and S'. Define $\epsilon''\triangleq\mathbb{E}\big[\frac{1}{N}\eta\nabla f((\mathbf{x}_{k-\tau^k}',\xi_i))-\frac{1}{N}\eta\nabla f((\mathbf{x}_{k-\tau^k}',\xi_i'))\big]$. We have:

$$\begin{aligned}\mathbb{E}(\delta_{k+1}) \le&\mathbb{E}\big[(\delta_k)-\eta\frac{1}{N}\sum_{i=1}^N\nabla f((\mathbf{x}_{k-\tau^k},\xi_i))+\eta\frac{1}{N}\sum_{i=1}^N\nabla f((\mathbf{x}_{k-\tau^k}',\xi_i))\big]\\&+\mathbb{E}\big[\frac{1}{N}\eta\nabla f((\mathbf{x}_{k-\tau^k}',\xi_i))-\frac{1}{N}\eta\nabla f((\mathbf{x}_{k-\tau^k}',\xi_i'))\big]\\\overset{(a)}{\le}&\mathbb{E}\big[(\delta_k)-\eta A(\delta_{k-\tau^k})\big]+\epsilon'',\end{aligned} \tag{48}$$

where (a) follows from f is a quadratic convex function. We note that $\|\epsilon''\|\le\frac{2\eta M}{N}$ since we have the bounded gradient Assumption 5. Then, we have

$$\begin{bmatrix}\delta_{k+1}\\\delta_k\\\vdots\\\delta_{k-\tau^k+1}\\\delta_{k-\tau^k}\end{bmatrix}=\begin{bmatrix}1&0&\cdots&-\eta A&0\\1&0&\cdots&0&0\\\vdots&\vdots&\ddots&\vdots&\vdots\\\vdots&\vdots&&\ddots&\vdots\\0&0&\cdots&1&0\end{bmatrix}\begin{bmatrix}\delta_k\\\delta_{k-1}\\\vdots\\\delta_{k-\tau^k}\\\delta_{k-\tau^k-1}\end{bmatrix}+\begin{bmatrix}\epsilon''\\0\\\vdots\\0\\0\end{bmatrix}. \tag{49}$$

Let Matrix

$$\mathbf{Q} \triangleq \begin{bmatrix} 1 & 0 & \cdots & -\eta A & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}. \tag{50}$$

Consider the characteristic polynomial

$$\begin{bmatrix} 1 & 0 & \cdots & -\eta A & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \vdots \\ \mathbf{v}_{\tau^k+2} \end{bmatrix} = \lambda_{\mathrm{Q}} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \vdots \\ \mathbf{v}_{\tau^k+2} \end{bmatrix}, \tag{51}$$

which implies $\mathbf{v}_1 = \lambda_{\mathrm{Q}}\mathbf{v}_2$, $\mathbf{v}_2 = \lambda_{\mathrm{Q}}\mathbf{v}_3$, ...,$\mathbf{v}_{\tau^k+1} = \lambda_{\mathrm{Q}}\mathbf{v}_{\tau^k+2}$. Plugging this in to the first row, we have:

$$\big(- \lambda_{\mathrm{Q}}^{\tau^k+2} + \lambda_{\mathrm{Q}}^{\tau^k+1} + \lambda_{\mathrm{Q}}[-\eta A]\big)\mathbf{v}_{\tau^k+2} = 0. \tag{52}$$

Since $\mathbf{A}$ is symmetric, then it has eigenvalue decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$. Then equation can be written as:

$$\mathbf{U}\big(- \lambda_{\mathrm{Q}}^{\tau^k+2} + \lambda_{\mathrm{Q}}^{\tau^k+1} + \lambda_{\mathrm{Q}}[-\eta \mathbf{A}]\big)\mathbf{U}^\top = 0. \tag{53}$$

Let $\lambda_1, ..., \lambda_b \in [\mu, M]$ be eigenvalue of symmetric matrix $\mathbf{A}$. Then we have

$$\lambda_{\mathrm{Q}}^{\tau_k+1} \cdot \big[- \lambda_{\mathrm{Q}} + 1\big] = \lambda_{\mathrm{Q}}[\eta\lambda_i]. \tag{54}$$

Since $0 < \eta < \frac{1}{M}, d \geq 1, \lambda_1, ..., \lambda_b \in [\mu, M]$, then we have $0 \leq \eta\lambda_i \leq 1$. Thus, $\max(|\lambda_{\mathrm{Q}}|) = 1$. Thus, for $mod(k, q) \neq 0$ we have

$$\mathbb{E}\|\delta_{k+1}\| \leq \mathbb{E}\|\delta_k\| + \frac{2\eta M}{N}. \tag{55}$$

For those k-values that satisfy $mod(k, q) = 0$, we can also show $\|(\delta_{k+1})\| = \|(\mathbf{x}_{k+1}) - (\mathbf{x}'_{k+1})\| \leq \|(\delta_k)\| + \frac{2\eta M}{N}$. Also, from (55), we always have $\|(\delta_{k+1})\| \leq \|(\delta_k)\| + \frac{2\eta M}{N}$ for $mod(k, q) \neq 0$. By applying this bound inductively, we can bound δ_{k+1} using the total number K iterations as:

$$\| \delta_{k+1} \| \leq \frac{2\eta M K}{N}. \tag{56}$$

Lastly, it follows from the definition of algorithm stability and the M-Lipschitz Assumption 4 of the loss function that our SA-SpiderBoost algorithm has the following stability bound:

$$\epsilon' \leq M \cdot \mathbb{E} \| \delta_{t+1} \| \leq \frac{2\eta M^2 K}{N}.$$

This completes the proof.

B.2 Proofs of Theorem 3

Recall that update rule for SpiderBoost is given by:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \frac{1}{|S|} \sum_{i \in S} (\nabla f(\mathbf{x}_{k-\tau^k}, \xi_i) - \nabla f(\mathbf{x}_{k-\tau^k-1-\tau^{k-\tau^k-1}}, \xi_i) + \mathbf{v}_{k-1-\tau^k}). \tag{57}$$

Let S and S' be two data sets such that S and S' differ in at most one example, where $S = (\xi_1, \xi_2, ..., \xi_N)$ and$S' = (\xi_1', \xi_2', ..., \xi_N')$. Let $\delta_k \triangleq \mathbf{x}_k - \mathbf{x}_k'$. Suppose $\mathbf{x}_0 = \mathbf{x}_0'$.

Now, taking expectation of δ_{k+1} with respect of the algorithm, we get

$$\mathbb{E}(\delta_{k+1}) = \mathbb{E}(\mathbf{x}_{k+1}) - \mathbb{E}(\mathbf{x}_{k+1}') = \mathbb{E}(\delta_k) - \eta[\mathbb{E}(\mathbf{v}_k) - \mathbb{E}(\mathbf{v}_k')]. \tag{58}$$

Since we $\mathbf{v}_k$ is the unbiased estimated of $\nabla f(\mathbf{x}_{k-\tau^k})$, we have:

$$\mathbb{E}(\delta_{k+1}) = \mathbb{E}(\delta_k) - \eta[\mathbb{E}(\nabla f(\mathbf{x}_{k-\tau^k})) - \mathbb{E}(\nabla f(\mathbf{x}_{k-\tau^k}'))]. \tag{59}$$

At Step k, with probability $1 - 1/N$, the example is the same in S and S'. With probability $\frac{1}{N}$, the example is different in S and S'. Hence, we have

$$\begin{aligned}
\mathbb{E}\|(\delta_{k+1})\| \leq & \mathbb{E}\|(\delta_k) - \eta(\frac{1}{N}\sum_{i=1}^{N} \nabla f(\mathbf{x}_{k-\tau^k}, \xi_i)) + \eta(\frac{1}{N}\sum_{i=1}^{N} \nabla f(\mathbf{x}_{k-\tau^k}', \xi_i))\| \\
& + \frac{1}{N}\eta\|\nabla f(\mathbf{x}_{k-\tau^k}', \xi_i)) - \nabla f(\mathbf{x}_{k-\tau^k}', \xi_i'))\| \\
\overset{(a)}{\leq} & \mathbb{E}\|(\delta_k) - \eta(\frac{1}{N}\sum_{i=1}^{N} \nabla f(\mathbf{x}_{k-\tau^k}, \xi_i)) + \eta(\frac{1}{N}\sum_{i=1}^{N} \nabla f(\mathbf{x}_{k-\tau^k}', \xi_i))\| + \frac{2\eta M}{N} \\
\overset{(b)}{\leq} & \mathbb{E}\|(\delta_k)\| + \eta\mathbb{E}\|(\frac{1}{N}\sum_{i=1}^{N} \nabla f(\mathbf{x}_{k-\tau^k}, \xi_i)) - (\frac{1}{N}\sum_{i=1}^{N} \nabla f(\mathbf{x}_{k-\tau^k}', \xi_i))\| + \frac{2\eta M}{N} \\
\overset{(c)}{\leq} & \mathbb{E}\|(\delta_k)\| + 2\eta M + \frac{2\eta M}{N},
\end{aligned} \tag{60}$$

where (a) and (b) follows from the triangle inequality, (c) follows from the bounded gradient Assumption 5. We note that $\eta\|\nabla f(\mathbf{x}_{k-\tau^k}', \xi_i)) - \nabla f(\mathbf{x}_{k-\tau^k}', \xi_i'))\| \leq 2\eta M$ derives form the bounded gradient Assumption 5.

For those k-values that satisfy $mod(k, q) = 0$, we have $\|(\delta_{k+1})\| = \|(\mathbf{x}_{k+1}) - (\mathbf{x}_{k+1}')\| \leq \|(\delta_k)\| + \frac{2\eta M}{N}$. Also, from (60), we always have $\|(\delta_{k+1})\| \leq \|(\delta_k)\| + 2\eta M + \frac{2\eta M}{N}$ for $mod(k, q) \neq 0$. By applying this bound inductively, we can bound δ_{k+1} using the total number K iterations as:

$$\mathbb{E} \parallel \delta_{k+1} \parallel \leq 2\eta MK + \frac{2\eta MK}{N}. \tag{61}$$

Lastly, it follows from the definition of algorithm stability and the M-Lipschitz Assumption 4 of the loss function that our SA-SpiderBoost algorithm has the following stability bound:

$$\epsilon' \leq M \cdot \mathbb{E} \parallel \delta_{K+1} \parallel \leq 2\eta M^2 K + \frac{2\eta M^2 K}{N}.$$

C CONVERGENCE ANALYSIS OF SA-SPIDERBOOST FOR SHARED MEMORY

To prove the result stated in the theorem, we first prove a useful lemma below.

Lemma 3. *Let all assumptions hold and apply SA-SpiderBoost in Algorithm 3, if the parameters η, q and S are chosen such that*

$$\beta_1 \triangleq \frac{\eta}{2d} - \frac{L\eta^2}{2d} - \frac{L^2\eta^3}{d^2}\Big(\frac{q(\Delta+1)}{|S|} + \Delta^2\Big) > 0, \tag{62}$$

and if for $mod(k,q)=0$*,we always have*

$$\mathbb{E}\left\|\mathbf{v}_k-\nabla f(\mathbf{x}_k)\right\|^2\le\epsilon_1^2, \tag{63}$$

then the output point $\mathbf{x}_\zeta$ *of SA-SpiderBoost satisfies*

$$\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|^2\le\Big[\frac{2}{\beta_1 d}\big(\eta+\frac{2L^2}{d}(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^3\big)\Big]\epsilon_1^2+\Big[\frac{4L^2(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^2}{K\beta_1 d}+\frac{2}{K\beta_1}\Big](f(\mathbf{x}_0)-f^*). \tag{64}$$

Proof. We aim to bound $\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|^2$, ζ is random choose from array $1,...,K$. Following the inequality of arithmetic and geometric means, we have:

$$\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|^2=\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)-\mathbf{v}_\zeta+\mathbf{v}_\zeta\right\|^2\le 2\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)-\mathbf{v}_\zeta\right\|^2+2\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2. \tag{65}$$

Next, we bound the terms $\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2$ and $\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)-\mathbf{v}_\zeta\right\|^2$ on the right-hand-side of (65) individually.

To evaluate $\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)-\mathbf{v}_\zeta\right\|^2$, we start from the iteration relationship of our SA-SpiderBoost algorithm. Toward this end, we first bound the distance of the inner loop $\mathbb{E}\left\|\mathbf{v}_k-\frac{1}{N}\sum_{i=1}^N\nabla f(\mathbf{x}_{k-\tau^k},\xi_i)\right\|^2$. Let $n_k=\lceil k/q\rceil$ such that $(n_k-1)q\le k\le n_kq-1$, we have:

$$\begin{aligned}
&\mathbb{E}\left\|\mathbf{v}_k-\frac{1}{N}\sum_{i=1}^N\nabla f(\mathbf{x}_{k-\tau^k},\xi_i)\right\|^2\\
&\overset{(a)}{=}\mathbb{E}\Big\|\frac{1}{|S|}\sum_{i\in S}\Big[\nabla f(\mathbf{x}_{k-\tau^k},\xi_i)-\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i)-\frac{1}{N}\sum_{j=1}^N\nabla f(\mathbf{x}_{k-\tau^k},\xi_j)\\
&\quad+\frac{1}{N}\sum_{j=1}^N\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_j)\Big]\Big\|^2+\mathbb{E}\Big\|\frac{1}{|S|}\sum_{i\in S}\Big[\mathbf{v}_{k-1-\tau^k}-\frac{1}{N}\sum_{j=1}^N\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_j)\Big]\Big\|^2\\
&\overset{(b)}{\le}\frac{1}{|S|^2}\sum_{i\in S}\mathbb{E}\Big\|\big[\nabla f(\mathbf{x}_{k-\tau^k},\xi_i)-\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i)\big]-\frac{1}{N}\sum_{j=1}^N\nabla f(\mathbf{x}_{k-\tau^k},\xi_j)\\
&\quad+\frac{1}{N}\sum_{j=1}^N\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_j)\Big\|^2+\mathbb{E}\Big\|\mathbf{v}_{k-1-\tau^k}-\frac{1}{N}\sum_{i=1}^N\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i)\Big\|^2\\
&\overset{(c)}{\le}\frac{1}{|S|^2}\sum_{i\in S}\mathbb{E}\left\|\nabla f(\mathbf{x}_{k-\tau^k},\xi_i)-\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i)\right\|^2\\
&\quad+\mathbb{E}\Big\|\mathbf{v}_{k-1-\tau^k}-\frac{1}{N}\sum_{i=1}^N\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i)\Big\|^2\\
&\overset{(d)}{\le}\frac{L^2}{|S|^2}\sum_{i\in S}\mathbb{E}\left\|\mathbf{x}_{k-\tau^k}-\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}}\right\|^2+\mathbb{E}\Big\|\mathbf{v}_{k-1-\tau^k}-\frac{1}{N}\sum_{i=1}^N\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i)\Big\|^2\\
&\overset{(e)}{=}\frac{L^2\eta^2}{|S|d}\mathbb{E}\Big\|\sum_{i=k-\tau^k-1-\tau^{k-\tau^k-1}}^{k-\tau^k-1}\mathbf{v}_i\Big\|^2+\mathbb{E}\Big\|\mathbf{v}_{k-1-\tau^k}-\frac{1}{N}\sum_{i=1}^N\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i)\Big\|^2\\
&\overset{(f)}{\le}\frac{L^2\eta^2}{|S|d}(\Delta+1)\sum_{i=k-\tau^k-1-\tau^{k-\tau^k-1}}^{k-\tau^k-1}\mathbb{E}\left\|\mathbf{v}_i\right\|^2+\mathbb{E}\Big\|\mathbf{v}_{k-1-\tau^k}-\frac{1}{N}\sum_{i=1}^N\nabla f(\mathbf{x}_{k-1-\tau^k-\tau^{k-1-\tau^k}},\xi_i)\Big\|^2,
\end{aligned} \tag{66}$$

where (a)follows from the gradient update rule $\mathbf{v}_k = \frac{1}{|S|}\sum_{i\in S}(\nabla f(\mathbf{x}_{k-\tau^k},\xi_i) - \nabla f(\mathbf{x}_{k-1-\tau^k,i}) + \mathbf{v}_{k-1-\tau^k-\tau^{k-1-\tau^k}})$ and Lemma 1 in [7], (b) and (c)use A.3 in [7], (d) follows from Lipchitz continuity of $\nabla f(x)$. (e) is due to the condition on update rule. (f) follows from the maximum delay is Δ.

Since $\mathbf{v}_{k-1-\tau^k}$ are generated from previous step, telescoping q iterations, we obtain that:

$$\mathbb{E}\Big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau^k},\xi_i)\Big\|^2 \le \frac{L^2\eta^2(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k-1-\tau^k}\mathbb{E}\Big\|\mathbf{v}_j\Big\|^2 + \mathbb{E}\Big\|\mathbf{v}_{(n_k-1)q} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{(n_k-1)q},\xi_i)\Big\|^2. \tag{67}$$

Next, we can conclude that,

$$\begin{aligned}
&\mathbb{E}\Big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_k,\xi_i)\Big\|^2\\
&\overset{(a)}{\le} 2\mathbb{E}\Big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau^k},\xi_i)\Big\|^2 + 2\mathbb{E}\Big\|\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau^k},\xi_i) - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_k,\xi_i)\Big\|^2\\
&\overset{(b)}{\le} 2\mathbb{E}\Big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau^k},\xi_i)\Big\|^2 + 2L^2\mathbb{E}\Big\|\mathbf{x}_{k-\tau^k} - \mathbf{x}_k\Big\|^2\\
&= 2\mathbb{E}\Big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau^k},\xi_i)\Big\|^2 + 2L^2\mathbb{E}\Big\|\sum_{j=k-\tau^k}^{k-1}(\mathbf{x}_{j+1}-\mathbf{x}_j)\Big\|^2\\
&\overset{(c)}{\le} 2\mathbb{E}\Big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau^k},\xi_i)\Big\|^2 + 2L^2\Delta\sum_{j=k-\tau^k}^{k-1}\mathbb{E}\Big\|(\mathbf{x}_{j+1}-\mathbf{x}_j)\Big\|^2\\
&\overset{(d)}{=} 2\mathbb{E}\Big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau^k},\xi_i)\Big\|^2 + \frac{2L^2\eta^2\Delta}{d}\sum_{j=k-\tau^k}^{k-1}\mathbb{E}\Big\|\mathbf{v}_j\Big\|^2\\
&\overset{(e)}{\le} \frac{2L^2\eta^2(\Delta+1)}{|S|d}\sum_{j=(n_k-1)q}^{k-1-\tau^k}\mathbb{E}\Big\|\mathbf{v}_j\Big\|^2 + 2\mathbb{E}\Big\|\mathbf{v}_{(n_k-1)q} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{(n_k-1)q},\xi_i)\Big\|^2\\
&+ \frac{2L^2\eta^2\Delta}{d}\sum_{j=k-\tau^k}^{k-1}\mathbb{E}\Big\|\mathbf{v}_j\Big\|^2,
\end{aligned} \tag{68}$$

where (a)follows from the triangle inequality, (b) follows from L-smooth property of $f(x)$. (c) is due to the triangle inequality and the maximum delay is Δ. (d) uses the condition on update rule and (e) follows from (67).

Next, we continue to bound the other term $\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2$ on the right-hand-side of (65). To evaluate $\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2$, we start from the iteration relationship of our SA-SpiderBoost algorithm. By Assumption 6, the entire objective function f is $L-$smooth, which further implies

$$\begin{aligned}
f(\mathbf{x}_{k+1}) &\le f(\mathbf{x}_k) + \langle\nabla f(\mathbf{x}_k), \mathbf{x}_{k+1}-\mathbf{x}_k\rangle + \frac{L}{2d}\Big\|\mathbf{x}_{k+1}-\mathbf{x}_k\Big\|^2\\
&\overset{(a)}{\le} f(\mathbf{x}_k) + \frac{\eta}{2d}\Big\|\mathbf{v}_k - \nabla f(\mathbf{x}_k)\Big\|^2 - (\frac{\eta}{2d} - \frac{L\eta^2}{2d})\Big\|\mathbf{v}_k\Big\|^2,
\end{aligned} \tag{69}$$

where (a) uses the update rule of SA-Spiderboost, and the inequality that $\langle \mathbf{x}, \mathbf{y}\rangle \leq \frac{\|\mathbf{x}\|^2+\|\mathbf{y}\|^2}{2}$, for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. Thus, we have

$$
\begin{aligned}
\mathbb{E}f(\mathbf{x}_{k+1}) \leq & \mathbb{E}f(\mathbf{x}_k) + \frac{\eta}{2}\mathbb{E}\left\|\mathbf{v}_k - \nabla f(\mathbf{x}_k)\right\|^2 - (\frac{\eta}{2} - \frac{L\eta^2}{2})\mathbb{E}\left\|\mathbf{v}_k\right\|^2 \\
\overset{(a)}{\leq} & \frac{\eta}{2d}\Big(\frac{2L^2\eta^2(\Delta+1)}{|S|d}\sum_{j=(n_k-1)q}^{k-1-\tau^k}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 + 2\mathbb{E}\left\|\mathbf{v}_{(n_k-1)q} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{(n_k-1)q}, \xi_i)\right\|^2\Big) \\
& + \frac{\eta}{2d}\Big(\frac{2L^2\eta^2\Delta}{d}\sum_{j=k-\tau^k}^{k-1}\mathbb{E}\left\|\mathbf{v}_j\right\|^2\Big) - (\frac{\eta}{2d} - \frac{L\eta^2}{2d})\mathbb{E}\left\|\mathbf{v}_k\right\|^2 + \mathbb{E}f(\mathbf{x}_k) \\
\overset{(b)}{\leq} & \mathbb{E}f(\mathbf{x}_k) - (\frac{\eta}{2d} - \frac{L\eta^2}{2d})\mathbb{E}\left\|\mathbf{v}_k\right\|^2 + \frac{\eta}{d}\epsilon_1^2 + \frac{L^2\eta^3(\Delta+1)}{|S|d^2}\sum_{j=(n_k-1)q}^{k-1-\tau^k}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 + \frac{L^2\eta^3\Delta}{d^2}\sum_{j=k-\tau^k}^{k-1}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 \\
\leq & \mathbb{E}f(\mathbf{x}_k) - (\frac{\eta}{2d} - \frac{L\eta^2}{2d})\mathbb{E}\left\|\mathbf{v}_k\right\|^2 + \frac{\eta}{d}\epsilon_1^2 + \frac{L^2\eta^3(\Delta+1)}{|S|d^2}\sum_{j=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 + \frac{L^2\eta^3\Delta}{d^2}\sum_{j=k-\tau^k}^{k-1}\mathbb{E}\left\|\mathbf{v}_j\right\|^2, \quad (70)
\end{aligned}
$$

where $\overset{(a)}{\leq}$ follows from (68), $\overset{(b)}{\leq}$ follows from the $\mathbb{E}\left\|\mathbf{v}_{(n_k-1)q} - \nabla f(\mathbf{x}_{(n_k-1)q})\right\|^2 \leq \epsilon_1^2$.

Next, telescoping (70)over k from $(n_k-1)q$ to k where $k \leq n_k q - 1$, we have

$$
\begin{aligned}
\mathbb{E}f(\mathbf{x}_{k+1}) \overset{(a)}{\leq} & \mathbb{E}f(\mathbf{x}_{(n_k-1)q}) - (\frac{\eta}{2d} - \frac{L\eta^2}{2d})\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + \sum_{i=(n_k-1)q}^{k}\frac{\eta}{d}\epsilon_1^2 \\
& + \frac{L^2\eta^3(\Delta+1)}{|S|d^2}\sum_{j=(n_k-1)q}^{k}\sum_{i=(n_k-1)q}^{j}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + \frac{L^2\eta^3\Delta}{d^2}\sum_{j=(n_k-1)q}^{k}\sum_{i=j-\tau^j}^{j-1}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 \\
\overset{(b)}{\leq} & \mathbb{E}f(\mathbf{x}_{(n_k-1)q}) - (\frac{\eta}{2d} - \frac{L\eta^2}{2d})\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + \sum_{i=(n_k-1)q}^{k}\frac{\eta}{d}\epsilon_1^2 \\
& + \frac{L^2\eta^3(\Delta+1)}{|S|d^2}\sum_{j=(n_k-1)q}^{k}\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + \frac{L^2\eta^3\Delta^2}{d^2}\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2, \quad (71)
\end{aligned}
$$

where (a) follows from (70), (b) extends the summation of second term form j to k. It then follows that:

$$
\begin{aligned}
\mathbb{E}f(\mathbf{x}_{k+1}) \overset{(a)}{\leq} & \mathbb{E}f(\mathbf{x}_{(n_k-1)q}) - (\frac{\eta}{2d} - \frac{L\eta^2}{2d})\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + \sum_{i=(n_k-1)q}^{k}\frac{\eta}{d}\epsilon_1^2 \\
& + \frac{qL^2\eta^3(\Delta+1)}{|S|d^2}\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + \frac{L^2\eta^3\Delta^2}{d^2}\sum_{j=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 \\
= & \mathbb{E}f(\mathbf{x}_{(n_k-1)q}) + \sum_{i=(n_k-1)q}^{k}\frac{\eta}{d}\epsilon_1^2 - \Big[\frac{\eta}{2d} - \frac{L\eta^2}{2d} - \frac{L^2\eta^3}{d^2}\Big(\frac{q(\Delta+1)}{|S|} + \Delta^2\Big)\Big]\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 \\
= & \mathbb{E}[f(\mathbf{x}_{(n_k-1)q})] - \sum_{i=(n_k-1)q}^{k}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2 - \frac{\eta}{d}\epsilon_1^2), \quad (72)
\end{aligned}
$$

where (a) follows from (71). Then, we can further derive:

$$\begin{aligned}
&\mathbb{E}f(\mathbf{x}_K) - \mathbb{E}f(\mathbf{x}_0) = (\mathbb{E}f(\mathbf{x}_q) - \mathbb{E}f(\mathbf{x}_0)) + (\mathbb{E}f(\mathbf{x}_{2q}) - \mathbb{E}f(\mathbf{x}_q) + ... + (\mathbb{E}f(\mathbf{x}_K) - \mathbb{E}f(\mathbf{x}_{(n_k-1)q})))\\
&\overset{(a)}{\leq} -\sum_{i=0}^{q-1}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2 - \frac{\eta}{d}\epsilon_1^2) - \sum_{i=q}^{2q-1}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2 - \frac{\eta}{d}\epsilon_1^2) - ... - \sum_{(n_K-1)q}^{K-1}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2 - \frac{\eta}{d}\epsilon_1^2)\\
&= -\sum_{i=0}^{K-1}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2 - \frac{\eta}{d}\epsilon_1^2) = -\sum_{i=0}^{K-1}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2) + \frac{K\eta}{d}\epsilon_1^2, \qquad (73)
\end{aligned}$$

where (a) follows from eq.(72),. Since $\mathbb{E}f(\mathbf{x}_K) \geq f(x^*)$, then we have

$$\mathbb{E}f(x^*) - \mathbb{E}f(\mathbf{x}_0) \leq -\sum_{i=0}^{K-1}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2) + \frac{K\eta}{d}\epsilon_1^2. \qquad (74)$$

By rearranging (74), we have:

$$\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2 = \frac{1}{K}\sum_{i=0}^{K-1}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 \leq \frac{f(\mathbf{x}_0) - f^*}{K\beta_1} + \frac{\eta}{d\beta_1}\epsilon_1^2. \qquad (75)$$

It then follow that:

$$\begin{aligned}
\mathbb{E}\left\|\mathbf{v}_\zeta - \nabla f(\mathbf{x}_\zeta)\right\|^2 &\overset{(a)}{\leq} \mathbb{E}\frac{2L^2\eta^2(\Delta+1)}{|S|d}\sum_{j=(n_\zeta-1)q}^{\zeta-1-\tau^\zeta}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 + 2\mathbb{E}\left\|\mathbf{v}_{(n_k-1)q} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{(n_k-1)q}, \xi_i)\right\|^2\\
&+ \frac{2L^2\eta^2\Delta}{d}\mathbb{E}\sum_{j=\zeta-\tau^\zeta}^{\zeta-1}\mathbb{E}\left\|\mathbf{v}_j\right\|^2\\
&\overset{(b)}{\leq} \frac{2L^2\eta^2(\Delta+1)}{|S|d}\mathbb{E}\sum_{j=(n_\zeta-1)q}^{\zeta-1-\tau^\zeta}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 + 2\epsilon_1^2 + \frac{2L^2\eta^2\Delta}{d}\mathbb{E}\sum_{i=\zeta-\tau^\zeta}^{\zeta-1}\mathbb{E}\left\|\mathbf{v}_i\right\|^2\\
&\leq \frac{2L^2\eta^2(\Delta+1)}{|S|d}\mathbb{E}\sum_{j=(n_\zeta-1)q}^{\xi}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 + 2\epsilon_1^2 + \frac{2L^2\eta^2\Delta}{d}\mathbb{E}\sum_{i=\zeta-\tau^\zeta}^{\zeta-1}\mathbb{E}\left\|\mathbf{v}_i\right\|^2\\
&\overset{(c)}{\leq} \frac{2L^2\eta^2(\Delta+1)}{|S|d}\mathbb{E}\sum_{i=(n_\zeta-1)q}^{min\{(n_\zeta)q-1,K-1\}}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + 2\epsilon_1^2 + \frac{2L^2\eta^2\Delta}{d}\mathbb{E}\sum_{i=\zeta-\tau^\zeta}^{\zeta-1}\mathbb{E}\left\|\mathbf{v}_i\right\|^2\\
&\overset{(d)}{\leq} \frac{q}{K}\sum_{i=0}^{K-1}\frac{2L^2\eta^2(\Delta+1)}{|S|d}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + 2\epsilon_1^2 + \frac{\Delta}{K}\sum_{i=0}^{K-1}\frac{2L^2\eta^2\Delta}{d}\mathbb{E}\left\|\mathbf{v}_i\right\|^2\\
&= (\frac{2L^2\eta^2q(\Delta+1)}{|S|d} + \frac{2L^2\eta^2\Delta^2}{d})\frac{1}{K}\sum_{i=0}^{K-1}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + 2\epsilon_1^2\\
&\overset{(e)}{\leq} \Big[\frac{2L^2(\Delta^2 + \frac{q(\Delta+1)}{|S|})\eta^3}{\beta_1 d^2} + 2\Big]\epsilon_1^2 + \frac{2L^2(\Delta^2 + \frac{q(\Delta+1)}{|S|})\eta^2}{d}\Big[\frac{f(\mathbf{x}_0) - f^*}{K\beta_1}\Big], \qquad (76)
\end{aligned}$$

where (a) follows from (64), (b) follows from (70), (c) follows from the definition of n_ζ, which implies $\xi \leq min\{(n_\zeta)q - 1, K-1\}$, (d)follows from the fact that the probability that $n_\zeta = 1, 2, ..., n_K$ is less than or equal to $\frac{q}{K}$, and (e)follows from (77). Then we can obtain

$$
\begin{aligned}
&\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|^2 \le 2\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)-\mathbf{v}_\zeta\right\|^2+2\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2\\
&\overset{(a)}{\le} 2\Big\{\Big[\frac{2L^2(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^3}{\beta_1 d^2}+2\Big]\epsilon_1^2+\frac{2L^2(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^2}{d}\Big[\frac{f(\mathbf{x}_0)-f^*}{K\beta_1}\Big]\Big\}\\
&\quad+2[\frac{f(\mathbf{x}_0)-f^*}{K\beta_1}+\frac{\eta}{\beta_1 d}\epsilon_1^2]\\
&=\Big[\frac{2}{\beta_1 d}\big(\eta+\frac{2L^2}{d}(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^3\big)+4\Big]\epsilon_1^2+\Big[\frac{4L^2(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^2}{K\beta_1 d}+\frac{2}{K\beta_1}\Big](f(\mathbf{x}_0)-f^*),
\end{aligned}
\tag{77}
$$

where (a) follows the (76) and (77). This completes the proof. □

To wrap up the proof of the theorem, we set the parameters as:

$$
q=\sqrt{N}, S=\sqrt{N},\quad \eta=\frac{1}{2L(\Delta+1)}. \tag{78}
$$

Then, we obtain:

$$
\beta_1=\frac{\eta}{2d}-\frac{L\eta^2}{2d}-\frac{L^2\eta^3}{d^2}\big(\frac{q(\Delta+1)}{|S|}+\Delta^2\big)=\frac{1}{4Ld(\Delta+1)}\cdot\big(1-\frac{1}{2(\Delta+1)}-\frac{(1+\Delta+\Delta^2)}{2d(\Delta+1)^2}\big)>0 \tag{79}
$$

for $mod(k,q)=0$ we have $\mathbb{E}\left\|\mathbf{v}_k-\nabla f(\mathbf{x}_k)\right\|^2=0$. Then, after K iterations, we have

$$
\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|^2\le\frac{8Ld(\Delta+1)}{K}\frac{2(\Delta+1)^2d+\Delta^2+\Delta+1}{2d(\Delta+1)^2-d(\Delta+1)-(\Delta^2+\Delta+1)}(f(\mathbf{x}_0)-f^*). \tag{80}
$$

To ensure $\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|\le\epsilon$, it suffices to ensure$\mathbb{E}\left\|\nabla f(\mathbf{x}_\zeta)\right\|^2\le\epsilon^2$. Solving for K yields:

$$
K=\frac{8Ld(\Delta+1)}{\epsilon^2}\frac{2(\Delta+1)^2d+\Delta^2+\Delta+1}{2d(\Delta+1)^2-d(\Delta+1)-(\Delta^2+\Delta+1)}(f(\mathbf{x}_0)-f^*)=O\big(\frac{f(\mathbf{x}_0)-f^*}{\epsilon^2}\big). \tag{81}
$$

Similar to Theorem 1, the total SFO complexity can be calculated as:$\lceil\frac{K}{q}\rceil N+K\cdot S\le\frac{K+q}{q}N+K\cdot S=K\sqrt{N}+N+K\sqrt{N}=O(\sqrt{N}\epsilon^{-2}(\Delta+1)d+N)$. This completes the proof.

D GENERALIZATION ANALYSIS OF SA-SPIDERBOOST FOR SHARED MEMORY

D.1 Proofs of Theorem 5

Proof. Recall that update rule for SA-SpiderBoost for shared-memory system is given by:

$$
(\mathbf{x}_{k+1})_{m_k}=(\mathbf{x}_k)_{m_k}-\eta\frac{1}{|S|}\sum_{i\in S}(\nabla f(\mathbf{x}_{k-\tau^k},\xi_i)-\nabla f(\mathbf{x}_{k-\tau^k-1-\tau^{k-\tau^k}-1},\xi_i)+\mathbf{v}_{k-1-\tau^k})_{m_k}. \tag{82}
$$

Let S and S' be two data sets such that S and S' differ in at most one example, where $S=(\xi_1,\xi_2,...,\xi_N)$ and $S'=(\xi_1',\xi_2',...,\xi_N')$. Let $\delta_k\triangleq\mathbf{x}_k-\mathbf{x}_k'$. Suppose $\mathbf{x}_0=\mathbf{x}_0'$.

Now, taking expectation of δ_{k+1} with respect of the algorithm, we get

$$
\mathbb{E}(\delta_{k+1})_{m_k}=\mathbb{E}(\mathbf{x}_{k+1})_{m_k}-\mathbb{E}(\mathbf{x}_{k+1}')_{m_k}=\mathbb{E}(\delta_k)_{m_k}-\eta[\mathbb{E}(\mathbf{v}_k)_{m_k}-\mathbb{E}(\mathbf{v}_k')_{m_k}]. \tag{83}
$$

Since we $\mathbf{v}_k$ is the unbiased estimated of $\nabla f(\mathbf{x}_{k-\tau^k})$, we have:

$$
\mathbb{E}(\delta_{k+1})_{m_k}=\mathbb{E}(\delta_k)_{m_k}-\eta[\mathbb{E}(\nabla f(\mathbf{x}_{k-\tau^k}))_{m_k}-\mathbb{E}(\nabla f(\mathbf{x}_{k-\tau^k}'))_{m_k}]. \tag{84}
$$

At Step k, with probability $1-1/N$, the example is the same in S and S'. With probability $\frac{1}{N}$, the example is different in S and S'. Hence, we have

$$\begin{aligned}\mathbb{E}(\delta_{k+1})_{m_k} \leq & \mathbb{E}\Big[(\delta_k)_{m_k} - \eta\frac{1}{N}\sum_{i=1}^{N}\nabla f((\mathbf{x}_{k-\tau^k},\xi_i))_{m_k} + \eta\frac{1}{N}\sum_{i=1}^{N}\nabla f((\mathbf{x}'_{k-\tau^k},\xi_i))_{m_k}\Big] \\ & + \mathbb{E}\Big[\frac{1}{N}\eta\nabla f((\mathbf{x}'_{k-\tau^k},\xi_i))_{m_k} - \frac{1}{N}\eta\nabla f((\mathbf{x}'_{k-\tau^k},\xi'_i))_{m_k}\Big] \\ \overset{(a)}{\leq} & \mathbb{E}\Big[(\delta_k)_{m_k} - \eta A(\delta_{k-\tau^k})_{m_k}\Big] + \epsilon'', \end{aligned} \tag{85}$$

where (a) follows from f is a quadratic convex function. We note that $\|\epsilon''\| \leq \frac{2\eta M}{N\sqrt{d}}$ since we have the bounded gradient Assumption 5 and $m_k \in 1, 2, ..., d$ is the uniformly selected updated coordinate in x in iteration k. Then, we have

$$\begin{bmatrix}(\delta_{k+1})_{m_k}\\(\delta_k)_{m_k}\\\vdots\\\delta_{k-\tau^k+1}\\\delta_{k-\tau^k}\end{bmatrix} = \begin{bmatrix}1&0&\cdots&-\eta A&0\\1&0&\cdots&0&0\\\vdots&\vdots&\ddots&\vdots&\vdots\\\vdots&\vdots&&\ddots&\vdots\\0&0&\cdots&1&0\end{bmatrix}\begin{bmatrix}(\delta_k)_{m_k}\\(\delta_{k-1})_{m_k}\\\vdots\\(\delta_{k-\tau^k})_{m_k}\\(\delta_{k-\tau^k-1})_{m_k}\end{bmatrix} + \begin{bmatrix}\epsilon''\\0\\\vdots\\0\\0\end{bmatrix}. \tag{86}$$

Let Matrix

$$\mathbf{Q} \triangleq \begin{bmatrix}1&0&\cdots&-\eta A&0\\1&0&\cdots&0&0\\\vdots&\vdots&\ddots&\vdots&\vdots\\\vdots&\vdots&&\ddots&\vdots\\0&0&\cdots&1&0\end{bmatrix}. \tag{87}$$

Consider the characteristic polynomial

$$\begin{bmatrix}1&0&\cdots&-\eta A&0\\1&0&\cdots&0&0\\\vdots&\vdots&\ddots&\vdots&\vdots\\\vdots&\vdots&&\ddots&\vdots\\0&0&\cdots&1&0\end{bmatrix}\begin{bmatrix}\mathbf{v}_1\\\mathbf{v}_2\\\vdots\\\vdots\\\mathbf{v}_{\tau^k+2}\end{bmatrix} = \lambda_{\mathbf{Q}}\begin{bmatrix}\mathbf{v}_1\\\mathbf{v}_2\\\vdots\\\vdots\\\mathbf{v}_{\tau^k+2}\end{bmatrix}, \tag{88}$$

which implies $\mathbf{v}_1 = \lambda_{\mathbf{Q}}\mathbf{v}_2$., $\mathbf{v}_2 = \lambda_{\mathbf{Q}}\mathbf{v}_3$, ...,$\mathbf{v}_{\tau^k+1} = \lambda_{\mathbf{Q}}\mathbf{v}_{\tau^k+2}$ plugging this in to the first row, then we have

$$\big(-\lambda_{\mathbf{Q}}^{\tau^k+2} + \lambda_{\mathbf{Q}}^{\tau^k+1} + \lambda_{\mathbf{Q}}[-\eta A]\big)\mathbf{v}_{\tau^k+2} = 0. \tag{89}$$

Since A is symmetric, then it has eigenvalue decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$. Then equation can be written as:

$$\mathbf{U}\big(-\lambda_{\mathbf{Q}}^{\tau^k+2} + \lambda_{\mathbf{Q}}^{\tau^k+1} + \lambda_{\mathbf{Q}}[-\eta A]\big)\mathbf{U}^\top = 0. \tag{90}$$

Let $\lambda_1, ..., \lambda_b \in [\mu, M]$ be eigenvalue of symmetric matrix A. Then we have

$$\lambda_{\mathbf{Q}}^{\tau_k+1}\cdot\big[-\lambda_{\mathbf{Q}}+1\big] = \lambda_{\mathbf{Q}}[\eta\lambda_i]. \tag{91}$$

Since $0 < \eta < \frac{1}{M}, d \geq 1, \lambda_1, ..., \lambda_b \in [\mu, M]$, then we have $0 \leq \eta\lambda_i \leq 1$. Thus, $max(|\lambda_Q|) = 1$. Thus, for $mod(k, q) \neq 0$ we have

$$\mathbb{E}\|(\delta_{k+1})_{m_k}\| \leq \mathbb{E}\|(\delta_k)_{m_k}\| + \frac{2\eta M}{N\sqrt{d}}. \tag{92}$$

For those k-values that satisfy $mod(k, q) = 0$, we can also show $\|(\delta_{k+1})_{m_k}\| = \|(\mathbf{x}_{k+1})_{m_k} - (\mathbf{x}'_{k+1})_{m_k}\| \leq \|(\delta_k)_{m_k}\| + \frac{2\eta M}{N\sqrt{d}}$. Also, from (92), we always have $\|(\delta_{k+1})_{m_k}\| \leq \|(\delta_k)_{m_k}\| + \frac{2\eta M}{N\sqrt{d}}$. By applying this bound inductively, we can bound δ_{k+1} using the total number K iterations as:

$$\mathbb{E} \parallel \delta_{k+1} \parallel \leq \frac{2\eta MK}{N\sqrt{d}}. \tag{93}$$

Similar to Theorem 2, the SA-SpiderBoost algorithm has the following stability bound:

$$\epsilon' \leq M \cdot \mathbb{E} \parallel \delta_{t+1} \parallel \leq \frac{2\eta M^2 K}{N\sqrt{d}}. \tag{94}$$

This completes the proof. □

D.2 Proofs of Theorem 6

Proof. Recall that update rule for SpiderBoost is given by:

$$(\mathbf{x}_{k+1})_{m_k} = (\mathbf{x}_k)_{m_k} - \eta \frac{1}{|S|} \sum_{i \in S} (\nabla f(\mathbf{x}_{k-\tau^k}, \xi_i) - \nabla f(\mathbf{x}_{k-\tau^k-1-\tau^{k-\tau^k-1}}, \xi_i) + \mathbf{v}_{k-1-\tau^k})_{m_k}. \tag{95}$$

Let S and S' be two data sets such that S and S' differ in at most one example, where $S = (\xi_1, \xi_2, ..., \xi_N)$ and $S' = (\xi'_1, \xi'_2, ..., \xi'_N)$. Let $\delta_k \triangleq \mathbf{x}_k - \mathbf{x}'_k$. Suppose $\mathbf{x}_0 = \mathbf{x}'_0$.

Now, taking expectation of δ_{k+1} with respect of the algorithm, we get

$$\mathbb{E}(\delta_{k+1})_{m_k} = \mathbb{E}(\mathbf{x}_{k+1})_{m_k} - \mathbb{E}(\mathbf{x}'_{k+1})_{m_k} = \mathbb{E}(\delta_k)_{m_k} - \eta[\mathbb{E}(\mathbf{v}_k)_{m_k} - \mathbb{E}(\mathbf{v}'_k)_{m_k}]. \tag{96}$$

Since we $\mathbf{v}_k$ is the unbiased estimated of $\nabla f(\mathbf{x}_{k-\tau^k})$. we have

$$\mathbb{E}(\delta_{k+1})_{m_k} = \mathbb{E}(\delta_k)_{m_k} - \eta[\mathbb{E}(\nabla f(\mathbf{x}_{k-\tau^k}))_{m_k} - \mathbb{E}(\nabla f(\mathbf{x}'_{k-\tau^k}))_{m_k}]. \tag{97}$$

At Step k, with probability $1 - 1/N$, the example is the same in S and S'. With probability $\frac{1}{N}$, the example is different in S and S'. Hence, we have

$$\begin{aligned}
\mathbb{E}\|(\delta_{k+1})_{m_k}\| \leq & \mathbb{E}\|(\delta_k)_{m_k} - \eta(\frac{1}{N}\sum_{i=1}^{N} \nabla f(\mathbf{x}_{k-\tau^k}, \xi_i))_{m_k} + \eta(\frac{1}{N}\sum_{i=1}^{N} \nabla f(\mathbf{x}'_{k-\tau^k}, \xi_i))_{m_k}\| \\
& + \frac{1}{N}\eta\|\nabla f(\mathbf{x}'_{k-\tau^k}, \xi_i))_{m_k} - \nabla f(\mathbf{x}'_{k-\tau^k}, \xi'_i))_{m_k}\| \\
\overset{(a)}{\leq} & \mathbb{E}\|(\delta_k)_{m_k} - \eta(\frac{1}{N}\sum_{i=1}^{N} \nabla f(\mathbf{x}_{k-\tau^k}, \xi_i))_{m_k} + \eta(\frac{1}{N}\sum_{i=1}^{N} \nabla f(\mathbf{x}'_{k-\tau^k}, \xi_i))_{m_k}\| + \frac{2\eta M}{N\sqrt{d}} \\
\overset{(b)}{\leq} & \mathbb{E}\|(\delta_k)_{m_k}\| + \eta\mathbb{E}\|(\frac{1}{N}\sum_{i=1}^{N} \nabla f(\mathbf{x}_{k-\tau^k}, \xi_i))_{m_k} - (\frac{1}{N}\sum_{i=1}^{N} \nabla f(\mathbf{x}'_{k-\tau^k}, \xi_i))_{m_k}\| + \frac{2\eta M}{N\sqrt{d}} \\
\overset{(c)}{\leq} & \mathbb{E}\|(\delta_k)_{m_k}\| + \frac{2\eta M}{\sqrt{d}} + \frac{2\eta M}{N\sqrt{d}},
\end{aligned} \tag{98}$$

where (a) and (b) follows from the triangle inequality, (c) follows from the bounded gradient Assumption 5. We note that $\eta\|\nabla f(\mathbf{x}'_{k-\tau^k}, \xi_i))_{m_k} - \nabla f(\mathbf{x}'_{k-\tau^k}, \xi'_i))_{m_k}\| \leq \frac{2\eta M}{\sqrt{d}}$ derives form the

bounded gradient Assumption 5 and $m_k \in 1, 2, ..., d$ is the uniformly selected updated coordinate in x in iteration k.

While $i \neq m_k$, we have $(\delta_k)_{i+1} = (\mathbf{x}_k)_{i+1} - (\mathbf{x}'_k)_{i+1} == (\mathbf{x}_k)_i - (\mathbf{x}'_k)_i = (\delta_k)_i$ Then we have

$$\mathbb{E}\|(\delta_{k+1})\| \leq \mathbb{E}\|(\delta_k)\| + \frac{2\eta M}{\sqrt{d}} + \frac{2\eta M}{N\sqrt{d}}. \tag{99}$$

Thus, total number K of iterations satisfies

$$\| \delta_{k+1} \| \leq \frac{2\eta MK}{\sqrt{d}} + \frac{2\eta MK}{N\sqrt{d}}. \tag{100}$$

Similar to Theorem 3, the SA-SpiderBoost algorithm has the following stability bound:

$$\epsilon' \leq M \cdot \| \delta_{t+1} \| \leq \frac{2\eta M^2 K}{\sqrt{d}} + \frac{2\eta M^2 K}{N\sqrt{d}}. \tag{101}$$

This completes the proof. □