SYNTHESIS: A Semi-Asynchronous Path-Integrated Stochastic Gradient Method for Distributed Learning in Computing Clusters

Zhuqing Liu[1], Xin Zhang[2], and Jia Liu[1]
[1]Department of Electrical and Computer Engineering, The Ohio State University
[2]Department of Statistics, Iowa State University

ABSTRACT

To increase the training speed of distributed learning, recent years have witnessed a significant amount of interest in developing both synchronous and asynchronous distributed stochastic variance-reduced optimization methods. However, all existing synchronous and asynchronous distributed training algorithms suffer from various limitations in either convergence speed or implementation complexity. This motivates us to propose an algorithm called SYNTHESIS (semi-asynchronous path-integrated stochastic gradient search), which leverages the special structure of the variance-reduction framework to overcome the limitations of both synchronous and asynchronous distributed learning algorithms, while retaining their salient features. We consider two implementations of SYNTHESIS under distributed and shared memory architectures. We show that our SYNTHESIS algorithms have $O(\sqrt{N}\epsilon^{-2}(\Delta+1)+N)$ and $O(\sqrt{N}\epsilon^{-2}(\Delta+1)d+N)$ computational complexities for achieving an ϵ-stationary point in non-convex learning under distributed and shared memory architectures, respectively, where N denotes the total number of training samples and Δ represents the maximum delay of the workers. Moreover, we investigate the generalization performance of SYNTHESIS by establishing algorithmic stability bounds for quadratic strongly convex and non-convex optimization. We further conduct extensive numerical experiments to verify our theoretical findings.

CCS CONCEPTS

• **Computing methodologies** → **Distributed algorithms**; *Machine learning*.

KEYWORDS

Machine learning, asynchronous distributed optimization

ACM Reference Format:
Zhuqing Liu[1], Xin Zhang[2], and Jia Liu[1] . 2022. SYNTHESIS: A Semi-Asynchronous Path-Integrated Stochastic Gradient Method for Distributed Learning in Computing Clusters. In *The Twenty-third International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '22), October 17–20, 2022, Seoul, Republic of Korea.* ACM, New York, NY, USA, 24 pages. https://doi.org/10.1145/3492866.3549722

MobiHoc '22, October 17–20, 2022, Seoul, Republic of Korea

ACM ISBN 978-1-4503-9165-8/22/10…$15.00
https://doi.org/10.1145/3492866.3549722

1 INTRODUCTION

From the early days of machine learning (ML), the classical first-order stochastic gradient descent (SGD) method has been used as the workhorse training algorithm due to the high dimensionality of ML computing tasks and the large size of datasets. However, it is well known that the SGD algorithm suffers from slow convergence. To accelerate the traditional SGD approach, there have been two main approaches in the literature. The first approach is to exploit algorithmic techniques, including momentum [19], adaptive learning rates [7], etc. One of the most notable algorithmic acceleration approaches in recent years is the family of *"variance-reduced"* (VR) methods (see, e.g., SVRG [11], SAG [24], and SAGA [5] and many of their variants). The basic idea of these VR methods is to construct accurate gradient estimators periodically by recomputing (near) full gradients to reduce variance. In the VR family, the state-of-the-art is the SPIDER method (stochastic path-integrated differential estimator) by [6] and its enhanced version called SpiderBoost [27] (see Section 2 for more in-depth discussions). The second major approach to accelerate SGD is to leverage the *parallelism* in distributed computing clusters (e.g., from chip-scale to datacenter-scale GPU farms), thanks to SGD's decomposable structure implied by the use of mini-batches. Notably, both *distributed memory* parallelism on multiple GPUs [2, 15, 30] and *shared memory* parallelism on a multi-core machine [20, 32] have been exploited in SGD-based ML tasks, such as parallel SVM [26], parallel matrix factorization [25, 29] and distributed deep learning [28], just to name a few.

However, in the distributed and parallel computing approach for distributed ML, a key design dilemma is the architectural choices between *"synchronous" and "asynchronous"* implementations for the distributed SGD algorithm. A salient feature of synchronous parallel algorithms is that they have a more stable convergence performance in general. However, synchronous implementations suffer limitations in complexity in maintaining a common clock, straggling problems, and periodic spikes in data traffic. In comparison, asynchronous algorithms are easier to implement and cause less network traffic congestions and delays. However, a major limitation of asynchronous parallel algorithms is the inevitable impact of *stale stochastic gradient* information due to asynchronous updates. If not treated appropriately, the stale stochastic gradient information could significantly degrade the convergence performance of the asynchronous algorithm.

The pros and cons of "synchronous vs. asynchronous parallelisms" in the parallel SGD implementation motivate us to pursue a new **semi-asynchronous** distributed optimization method that achieves the best of both worlds while avoiding their pitfalls. Interestingly, our *key idea* in addressing the problems in "synchronous vs. asynchronous parallelisms" (i.e., the second SGD acceleration approach) comes from the VR-based algorithmic acceleration (i.e.,

the first SGD acceleration approach). Specifically, we show that the "*double-loop*" structure of the VR-based algorithms [6, 11, 27] naturally implies an elegant *semi-asynchronous* implementation, which entails a simple implementation as in asynchronous distributed algorithms, while providing strong convergence and generalization performance guarantees as in synchronous distributed algorithms. This key insight enables us to develop a new distributed learning algorithm called SYNTHESIS (semi-asynchronous path-integrated stochastic gradient search). Our main results and contributions are summarized as follows:

- We first analyze the convergence of SYNTHESIS for non-convex optimization for learning problems under the distributed memory. We show that SYNTHESIS achieves a computational complexity $O(\sqrt{N}\epsilon^{-2}(\Delta+1)+N)$ in terms of stochastic first-order oracle (SFO) evaluations to find an ϵ-approximate first-order stationary point (to be defined later), where N is the number of training samples, and Δ denotes the maximum delay in asynchronous update across all workers. Our result shows that delays in asynchrony only linearly affects the hidden constant in convergence performance and does *not* slow down the convergence rate order. 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).
- Next, we analyze the convergence of SYNTHESIS under the shared memory architecture. We show that the SYNTHESIS 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, under shared memory, 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. Nonetheless, this restriction does *not* affect the convergence rate with respect to ϵ.
- We further study the generalization performance of SYNTHESIS under both distributed memory and shared memory architectures. We establish the upper bounds of the expected generalization errors of SYNTHESIS for both convex and non-convex learnings. Our generalization bounds show that larger step-sizes and smaller training datasets cause larger generalization errors in SYNTHESIS. These results also characterize a fundamental trade-off between convergence and generalization in distributed learning algorithms.

The rest of the paper is organized as follows. In Section 2, we first discuss related work and preliminaries of distributed memory and shared memory architectures. In Section 3, we present the system model, problem formulation, and basic assumptions. In Sections 4 and 5, we analyze convergence and generalization performances of the SYNTHESIS method under the distributed memory and shared memory parallelisms, respectively. Section 6 presents numerical results and Section 7 concludes this paper. We note that the full version of the proofs in [16].

2 RELATED WORK AND PRELIMINARIES

In this section, we first provide an overview on variance-reduced first-order methods and asynchronous distributed first-order methods. Then, we offer some necessary background on two widely adopted parallel computing architectures, namely distributed memory and shared memory. To define the stochastic first-order oracle

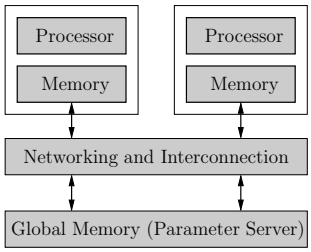


Figure 1: The distributed memory architecture.

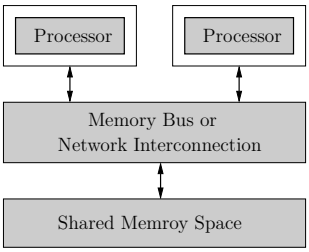


Figure 2: The shared memory architecture.

(SFO) complexity, we first introduce the notion of first-order ϵ-stationary point. We say that a point $\mathbf{x}$ is a first-order ϵ-stationary point of a function $f(\cdot)$ if the first-order gradient condition is satisfied: $\mathbb{E}[\|\nabla f(\mathbf{x})\|^2] \leq \epsilon^2$.

• **Variance-Reduced First-Order Methods:** The history of the basic SGD algorithm dates back to 1951 [22]. As mentioned in Section 1, it is well-known that the basic SGD method achieves an ϵ-approximate stationary point with an SFO evaluation cost of $O(\epsilon^{-4})$ [8]. To address this limitation, variance reduction techniques (VR) [12, 23] have emerged in recent years to improve the convergence rate of SGD. Notably, SAGA [5] and SVRG [11] achieve an SFO complexity of $O(N^{2/3}\epsilon^{-2})$ iterations to attain a first-order ϵ-stationary point for non-convex optimization. In the VR family, SPIDER [6] and SpiderBoost [27] are the state-of-the-art, achieving an SFO complexity of $O(\min(\sqrt{N}\epsilon^{-2}, \epsilon^{-3}))$.

• **Asynchronous Distributed Optimization for Learning:** One of the earliest asynchronous SGD-type algorithms is HOGWILD! [20], which is a lock-free asynchronous parallel implementation of SGD under the shared memory architecture with a sublinear convergence rate for strongly convex problems. Roughly the same time, the convergence of SGD with asynchronous gradients was studied in [2], which has an SFO complexity of $O(\epsilon^{-4})$ if the random gradient update delay is i.i.d second-moment-bounded. This $O(\epsilon^{-4})$ SFO complexity is the same as that of the synchronous and non-delayed case. Later, a coordinate-descent-based asynchronous parallel optimization algorithm termed ARock was proposed in [18], which generalizes asynchronous SGD (Async-SGD) for solving convex problems. In [10, 21] an asynchronous stochastic VR-based Async-SVRG method is proposed to achieve linear convergence for bounded delays for convex problems. However, the computation complexity of Async-SVRG suffers from $O(n^{-2/3}\epsilon^{-}2)$, which is higher compared to our SYNTHESIS algorithm from $O(n^{-1/2}\epsilon^{-2})$. Our better sample complexity result is due to the recursive structure in the asynchronous gradient estimator in our SYNTHESIS algorithm.

• **Distributed Memory vs. Shared Memory:** To facilitate later discussions, we provide a brief 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 contains processors and their associated local memory, as well as some form of interconnection that allows programs on each processor to interact with each other. In the context of distributed ML, thanks to the independent memory, each computing unit can 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* that can be simultaneously accessed by all processors/programs. Depending on context, programs may run on a single processor with multiple threads or multiple processors. In the context of distributed ML, the shared memory architecture is often used by a single machine with multiple cores/GPUs. The parameter values are stored in the shared memory and multiple threads can access them. Each thread reads the parameter values from the shared memory and randomly chooses a batch of samples to compute a stochastic gradient. Each thread then updates the current parameters with its stochastic gradient. However, due to the shared memory restriction, each thread can only read or write *a single coordinate at a time* to prevent race conditions [20].

With the basic notions of distributed and shared memory architectures above, we are now in a position to present our proposed SYNTHESIS algorithm in Section 3.

3 SYSTEM MODEL, PROBLEM FORMULATIONS AND BASIC ASSUMPTIONS

In this section, we present the system model, problem formulation and the assumptions used in this paper. We consider a distributed learning system with P workers and a parameter server. 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.) following a 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 $n \triangleq |\mathcal{S}_p| = N/P$.[1] The goal of the distributed learning system is to solve an optimization problem, which is typically non-convex and in the following form:

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

In this paper, we make the following assumptions:

Assumption 1 (Bounded Objective Function). *The 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 (Continuously Differentiable Loss Function). *The loss function $f(\cdot,\cdot)$ is continuously differentiable.*

Assumption 3 (M-Lipschitz Loss Function:). *$f(\cdot,\cdot)$ is M-Lipschitz continuous, i.e., there exists a constant $M > 0$ such that $|f(\mathbf{u},\xi_i) - f(\mathbf{v},\xi_i)| \le M\|\mathbf{u}-\mathbf{v}\|$, $\forall \mathbf{u},\mathbf{v}\in\mathbb{R}^d$, $\forall \xi_i$.*

Assumption 4 (Uniformly Bounded-Size Gradient:). *There exists a constant $M > 0$ such that $\sup_{\mathcal{S}} \|\nabla f(\cdot,\xi_i)\| \le M$.*

[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 5 (L-Lipschitz Smoothness). *The function $f(\cdot,\cdot)$ is L-Lipschitz smooth, i.e., there exists a constant $L > 0$ such that $\|\nabla f(\mathbf{u},\xi_i) - \nabla f(\mathbf{v},\xi_i)\| \le L\|\mathbf{u}-\mathbf{v}\|$, $\forall \mathbf{u},\mathbf{v}\in\mathbb{R}^d$, $\forall \xi_i$.*

Assumption 6 (Bounded Delay in Asychrony). *The maximum of random delay τ of asynchronous stochastic gradient updates is upper bounded by a constant $\Delta > 0$, i.e., $\tau \le \Delta$.*

Several remarks on Assumptions 1–6 are in order. Assumptions 1–5 are standard in convergence analysis in the literature. Assumption 6 is also a standard assumption in the asynchronous computing literature and holds in most practical computing systems. We note that Assumptions 3 and 4 share the same constant M since Assumption 4 is implied by Assumption 3 (but we state Assumption 4 explicitly for convenience in subsequent analysis). The bounded gradient norm in Assumption 4 is often guaranteed by stability-inducing operations, e.g., regularization, projection, or gradient clipping.

4 SYNTHESIS: DISTRIBUTED MEMORY

In this section, We first propose the SYNTHESIS algorithm for the distributed memory architecture to handle distributed ML problems with a large dataset that cannot fit in a single machine's storage. 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 SYNTHESIS in Section 4.3.

4.1 Algorithm Description

As mentioned in Section 1, due to the various pros and cons in "synchronous vs. asynchronous parallelisms," we pursue a new *semi-asynchronous* approach to achieve the best of both worlds in this paper. Our *key idea* is motivated by the observation that the double-loop structure of the state-of-the-art VR-based methods [6, 11, 27] can be leveraged to construct a simple and elegant *semi-asynchronous algorithm*. The server and worker algorithms of our SYNTHESIS are presented in Algorithms 1 and 2. In what follows, we take a closer look at these two algorithms.

1) The Inner Loop of Algorithm 1 (Asynchronous Mode): From the Server Code in Algorithm 1, we can see that the inner loop (Lines 8–9) is executed $q-1$ iterations in total. In the inner loop of the SYNTHESIS algorithm, on the worker side (Steps 3–6 of the Worker Code in Algorithm 2), each worker i independently retrieves the freshest parameter $\mathbf{x}_{new}$ from the parameter server and then randomly selects a mini-batch S of data samples from its local dataset and computes a stochastic gradient. Also, $\mathbf{v}_{old}$ is an unbiased gradient estimate of $\nabla f(\mathbf{x}_{old})$. Each worker immediately reports the computed stochastic gradient $\mathbf{v}_{new}$ 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 have been updated by other workers with gradient information associated with a fresher $\mathbf{x}$-value. Thus, this particular worker's gradient information could turn out to be "delayed."

2) The Outer Loop of Algorithm 1 (Synchronous Mode): On the other hand, at the beginning of every timeframe of length q, the

Algorithm 1: The Parameter Server Code of the Distributed SYNTHESIS Algorithm.

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

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 $\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \mathbf{v}_k$.

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

Algorithm 2: The Worker Code of Each Worker p of the Distributed SYNTHESIS 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 $\mathbf{v}_{old}$ from the parameter 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.

parameter server executes the outer loop (Line 1–7 in Algorithm 1). 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 Worker Code 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 (see Step 2 in Algorithm 2). 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,

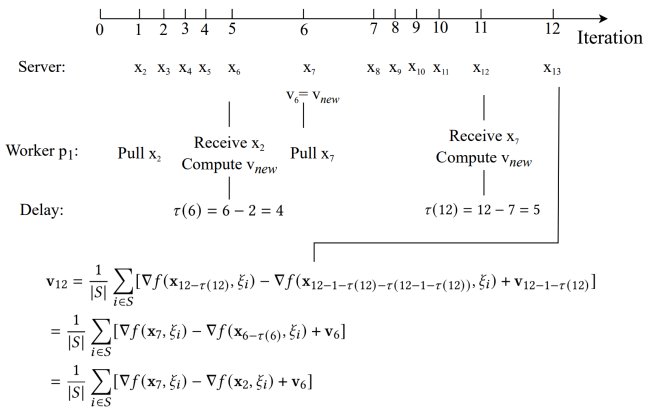


Figure 3: An example of the stale stochastic gradients due to asynchrony in SYNTHESIS: In the computation of direction $\mathbf{v}_{12}$ (for $\mathbf{x}_{13}$), stale stochastic gradients corresponding to $\mathbf{x}_7$ and $\mathbf{x}_2$ are used instead of $\mathbf{x}_{12}$ and $\mathbf{x}_{11}$.

$\mathbf{v}_k$ represents the *variance-reduced* update direction in iteration k. Following from Algorithms 1 (Step 10) and 2 (Step 5), we can express the algorithmic update of SYNTHESIS as follows:

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

where the index i denotes the index of a sample, and $\tau(k)$ denotes the delay of stochastic gradient information used to update $\mathbf{x}_k$ in iteration k (satisfying $\tau(k) \le \Delta$ for any iteration k). Thus, $\mathbf{x}_{k-\tau(k)}$ denotes the "new" parameter a worker uses to compute the gradient with delay $\tau(k)$ in a worker, and $\mathbf{x}_{(k-1-\tau(k))-\tau(k-1-\tau(k))}$ denotes the "old" parameter we used to compute the gradient with additional delay $\tau(k-1-\tau(k))$ (cf. Algorithm 2).

One important remark regarding the update in (1) is in order. We note that the algorithmic update in (1) integrates the path of the $\{\mathbf{v}_k\}$ trajectory (hence the name SYNTHESIS), which shares some similarity with the class of reccurssive variance-reduced gradient estimators in the family of VR methods (e.g., SARAH [17], SPIDER [6], SpiderBoost [27], and PAGE [14]). Thus, it is insightful to compare Eq. (1) with these existing works that have the following recursive form (denoted as "rec") in the sequence $\{\mathbf{v}_k^{(\mathrm{rec})}\}$:

$$\mathbf{v}_k^{(\mathrm{rec})} = \frac{1}{|S|}\sum_{i\in S}[\nabla f(\mathbf{x}_k, \xi_i) - \nabla f(\mathbf{x}_{k-1}, \xi_i) + \mathbf{v}_{k-1}^{(\mathrm{rec})}]. \quad (2)$$

The key difference and novelty in SYNTHESIS compared to these existing recursive VR methods is the *presence of stale stochastic gradient information* (due to asynchrony) from the iteration $k-\tau(k)$ and its previous step in iteration $k-1-\tau(k-1)$ (see Fig. 3 for an example of stale stochastic gradients in SYNTHESIS). This asynchrony-induced staleness coupled with the recursive structure render the algorithm performance analysis significantly more complex and challenging than the aforementioned existing works [6, 14, 17, 27].

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 Performance Analysis

In this subsection, we first present the main convergence result of SYNTHESIS under the distributed memory architecture. Due to space limitation, we provide a proof sketch here and relegate the proof to the full version of this paper.

Theorem 1 (Convergence of SYNTHESIS for Distributed Memory). *Let f^* denote the global optimal value. Under distributed memory and assumptions 1-6, for some $\epsilon > 0, P \leq \sqrt{N}$, by choosing parameters $q = |S| = \sqrt{N}$ and $\eta \leq \frac{1}{4L(\Delta+1)}$ for a given maximum delay $\Delta > 0$, the SYNTHESIS algorithm outputs an $\mathbf{x}_\zeta$ that satisfies $\mathbb{E}[\|\nabla f(\mathbf{x}_\zeta)\|^2] \leq \epsilon^2$ if the total number of iterations K satisfies $K = O(\frac{f(\mathbf{x}_0)-f^*}{\epsilon^2})$. 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 SYNTHESIS achieves a first-order stationary point with total computational complexity $O(\sqrt{N}\epsilon^{-2}(\Delta+1)+N)$. 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 state-of-the-art synchronous VR-based algorithms [6, 27], where N is the total training samples. ii) We note that the analysis of SYNTHESIS is very different from those of the synchronous VR-based algorithms. From a theoretical perspective, the major challenge in the proof of Theorem 1 is the asynchrony between different workers.

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

$$\begin{aligned}\mathbb{E}[\|\nabla f(\mathbf{x}_\zeta)\|^2] &= \mathbb{E}[\|\nabla f(\mathbf{x}_\zeta) - \mathbf{v}_\zeta + \mathbf{v}_\zeta\|^2] \\ &\leq 2\mathbb{E}[\|\nabla f(\mathbf{x}_\zeta) - \mathbf{v}_\zeta\|^2] + 2\mathbb{E}[\|\mathbf{v}_\zeta\|^2]. \end{aligned} \tag{3}$$

Next, we bound the terms $\mathbb{E}[\|\nabla f(\mathbf{x}_\zeta) - \mathbf{v}_\zeta\|^2]$ and $\mathbb{E}[\|\mathbf{v}_\zeta\|^2]$ on the right-hand-side of (3) individually.

Step 1) Bounding the gradient estimator bias $\mathbb{E}[\|\mathbf{v}_\zeta - \nabla f(\mathbf{x}_\zeta)\|^2]$*:* Toward this end, we first bound the distance between the local update direction and the node-average gradient direction $\mathbb{E}[\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i)\|^2]$ of the inner loop. Let $n_k = \lceil k/q \rceil$ denote the epoch index that iteration k belongs to (i.e., $(n_k-1)q \leq k \leq n_k q - 1$). Then, from the inner loop operations, we can show the following relationship:

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

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

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

Step 2) Bounding the second moment of the moving direction $\mathbb{E}[\|\mathbf{v}_\zeta\|^2]$*:* Consider the term $\mathbb{E}[\|\mathbf{v}_\zeta\|^2]$ in (3). To evaluate $\mathbb{E}[\|\mathbf{v}_\zeta\|^2]$, we start from the iteration relationship of our SYNTHESIS algorithm. From the L-smooth assumption, it can be shown that:

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

It follows from (5) and inductively bounding $\mathbb{E}[f(\mathbf{x}_{k+1})]$ in the inner loop $(n_k-1)q \leq k \leq n_k q - 1$ that:

$$\mathbb{E}[f(\mathbf{x}_{k+1})] \leq \mathbb{E}[f(\mathbf{x}_{(n_k-1)q})] + \sum_{i=(n_k-1)q}^{k}\eta\epsilon_1^2 - \Big[\frac{\eta}{2} - \frac{L\eta^2}{2} - L^2\eta^3\big(\frac{q(\Delta+1)}{|S|} + \Delta^2\big)\Big]\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left[\|\mathbf{v}_i\|^2\right]. \tag{6}$$

Next, letting $\beta_1 \triangleq \left[\frac{\eta}{2} - \frac{L\eta^2}{2} - L^2\eta^3\big(\frac{q(\Delta+1)}{|S|} + \Delta^2\big)\right]$ and inductively using (6), we can further drive the final upper bound of $\mathbb{E}[f(\mathbf{x}_{k+1})]$ as: $\mathbb{E}[f(\mathbf{x}_K)] - \mathbb{E}[f(\mathbf{x}_0)] \leq -\sum_{i=0}^{K-1}(\beta_1\mathbb{E}[\|\mathbf{v}_i\|^2]) + K\eta\epsilon_1^2$, which further implies that:

$$\mathbb{E}[f(\mathbf{x}^*)] - \mathbb{E}[f(\mathbf{x}_0)] \leq -\sum_{i=0}^{K-1}(\beta_1\mathbb{E}[\|\mathbf{v}_i\|^2]) + K\eta\epsilon_1^2. \tag{7}$$

Rearranging terms in (7) yields:

$$\mathbb{E}[\|\mathbf{v}_\zeta\|^2] = \frac{1}{K}\sum_{i=0}^{K-1}\mathbb{E}[\|\mathbf{v}_i\|^2] \leq \frac{f(\mathbf{x}_0) - f^*}{K\beta_1} + \frac{\eta}{\beta_1}\epsilon_1^2. \tag{8}$$

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

$$\mathbb{E}\left[\|\nabla f(\mathbf{x}_\zeta)\|^2\right] \leq \left[\frac{2}{\beta_1}\left(\eta + 2L^2\left(\Delta^2 + \frac{q(\Delta+1)}{|S|}\right)\eta^3\right) + 4\right]\epsilon_1^2 + \left[\frac{4L^2\left(\Delta^2 + \frac{q(\Delta+1)}{|S|}\right)\eta^2}{K\beta_1} + \frac{2}{K\beta_1}\right](f(\mathbf{x}_0) - f^*). \tag{9}$$

Lastly, we choose the following parameter: $q = \sqrt{N}$, $S = \sqrt{N}$, $\eta \leq \frac{1}{4L(\Delta+1)}$. Plugging the above parameters in the definitions of β_1, we obtain that:

$$\beta_1 = \frac{\eta}{2} - \frac{L\eta^2}{2} - L^2\eta^3\left(\frac{q(\Delta+1)}{|S|} + \Delta^2\right) > 0. \tag{10}$$

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

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

Solving for K to to ensure that $\mathbb{E}\left[\|\nabla f(\mathbf{x}_\zeta)\|^2\right] \leq \epsilon^2$, we have

$$K = O\left(\frac{(f(\mathbf{x}_0) - f^*)(\Delta+1)}{\epsilon^2}\right).$$

This completes the first part of the theorem.

Lastly, to show the SFO complexity, 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 Performance Analysis

After studying the convergence performance of SYNTHESIS under the distributed memory architecture, in this subsection, we turn our attention to the generalization performance of SYNTHESIS under distributed memory, i.e., how accurate a model trained by SYNTHESIS 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 finite-sample empirical risk minimization (ERM). 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 $\mathcal{D}$. Let A represent an algorithm (A could potentially be randomized) and let S represent a training dataset used by A. Then, the generalization error of the algorithm A can be defined as the gap between the ERM problem and the population risk minimization problem: $|\mathbb{E}_{S,A}[F_N[A(S)] - F[A(S)]|$. Our generalization analysis is based on the notion of algorithmic stability [9], which is restated as follows:

Definition 1 (Algorithmic Stability [9]). *Let S and S' be two datasets that differ by at most one element. For $\epsilon' > 0$, 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 $\mathcal{D}$ is the distribution of data sample ξ.*

It is shown [4] that if an algorithm A is ϵ'-uniformly stable, then the average generalization error of A is bounded as $|\mathbb{E}_{S,A}[F_N[A(S)] - F[A(S)]]| \leq \epsilon'$. Here, our goal is to study the stability of SYNTHESIS under the distributed memory architecture. Recall that the update rule for SYNTHESIS is given by

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

Consider two datasets $S = \{\xi_1, \xi_2, ..., \xi_N\}$ and $S' = (\xi_1', \xi_2', ..., \xi_N')$ that differ by at most one sample. The next two theorems state our main results on algorithmic stability (or equivalently, generalization error) of SYNTHESIS for convex and non-convex loss problems, respectively. Due to space limitation, the proof details are provided in the online technical report of this paper [16].

Theorem 2 (Algorithm Stability). *Under assumptions 1-6, the distributed-memory-based SYNTHESIS is $(2\eta M^2K^2 + (2M^2K^2\eta)/N)$-uniformly stable. In addition to the above conditions, if $f(x;\xi_i)$ is quadratic and μ-strongly convex, the distributed-memory-based SYNTHESIS is $(2\eta M^2 K)/N$-uniformly stable.*

Remark 2. Two remarks on Theorems 2 are in order: i) Although seemingly ideal, the quadratic strongly convex and smooth setting remains of practical interest. For example, the square loss and linear learning model naturally fits 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 2. ii) The distributed-memory-based SYNTHESIS under both convexity settings generalizes better as the number of training data samples N increases or as the learning rate η decreases. Since a small learning rate implies slower convergence, there is a fundamental *trade-off between training convergence speed and generalization performance*. Furthermore, the stronger convexity condition leads to tighter stability bound (i.e., generalizes better).

We note that our generalization analysis offers the first theoretical understanding of generalization performance for semi-asynchronous variance-reduced learning algorithms. 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 the 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 obtain a tighter stability error bound in Theorem 2. Also, the variance reduction component, the asynchrony in the algorithm, and the non-convexity also create challenges in analyzing the stability performance of SYNTHESIS .

Proof Sketch of Theorem 2. Due to space limitation, we provide a proof sketch here and refer readers to our technical report [16] for details. Let $S = (\xi_1, \xi_2, ..., \xi_N)$ and $S' = (\xi_1', \xi_2', ..., \xi_N')$ be two adjacent datasets that differ by at most one element. Define $\delta_k \triangleq \mathbf{x}_k - \mathbf{x}_k'$. Next, we structure our proof in several major steps. We let $\mathbf{x}_0 = \mathbf{x}_0'$ and 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

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

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

$$\begin{aligned}\mathbb{E}[\|(\delta_{k+1})\|] &\leq \mathbb{E}\Bigg[\Bigg\|(\delta_k) - \eta\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau(k)},\xi_i) \\ &+ \eta\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau(k)}',\xi_i)\Bigg\|\Bigg] + \mathbb{E}\Bigg[\Bigg\|\frac{1}{N}\eta\nabla f(\mathbf{x}_{k-\tau(k)}',\xi_i) \\ &- \frac{1}{N}\eta\nabla f(\mathbf{x}_{k-\tau(k)}',\xi_i')\Bigg\|\Bigg] \overset{(a)}{\leq} \mathbb{E}[\|(\delta_k)\|] + 2\eta M + \frac{2\eta M}{N}, \qquad (11)\end{aligned}$$

where (a) follows from the bounded gradient assumption.

Step 2) Bounding the error $\mathbb{E}[\|\delta_{k+1}\|]$: 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 (11), 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:

$$\mathbb{E}[\|\delta_{k+1}\|] \leq 2\eta MK + \frac{2\eta MK}{N}. \tag{12}$$

Step 3) Bounding the ϵ'-stability of SYNTHESIS in the non-convex case: Lastly, it follows from the definition of algorithmic stability and the M-Lipschitz assumption of the loss function that our SYNTHESIS 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 of stability bound with the non-convex loss function of Theorem 2. Next, we will provide the proof of the condition with quadratic strongly convex loss function.

Step 4) 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:

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

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:

$$\begin{aligned}&\mathbb{E}(\delta_{k+1}) \leq \mathbb{E}\big[(\delta_k) - \eta\frac{1}{N}\sum_{i=1}^{N}\nabla f((\mathbf{x}_{k-\tau(k)}, \xi_i)) + \eta\frac{1}{N}\sum_{i=1}^{N} \\ &\nabla f((\mathbf{x}'_{k-\tau(k)}, \xi_i))\big] + \mathbb{E}\big[\frac{1}{N}\eta\nabla f((\mathbf{x}'_{k-\tau(k)}, \xi_i)) - \frac{1}{N}\eta \\ &\nabla f((\mathbf{x}'_{k-\tau(k)}, \xi'_i))\big] \leq \mathbb{E}\big[(\delta_k) - \eta A(\delta_{k-\tau(k)})\big] + \epsilon''.\end{aligned} \tag{13}$$

Step 5) Bounding $\mathbb{E}[\|\delta_{k+1}\|]$: A key novelty in theoretical analysis of this paper is that, 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 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:

$$\begin{bmatrix}\delta_{k+1}\\ \delta_k\\ \dots\\ \delta_{k-\tau(k)+1}\\ \delta_{k-\tau(k)}\end{bmatrix} = \underbrace{\begin{bmatrix}1 & 0 & \cdots & -\eta A & 0\\ 1 & 0 & \cdots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ \vdots & \vdots & & \ddots & \vdots\\ 0 & 0 & \cdots & 1 & 0\end{bmatrix}}_{\mathbf{Q}} \begin{bmatrix}\delta_k\\ \delta_{k-1}\\ \dots\\ \delta_{k-\tau(k)}\\ \delta_{k-\tau(k)-1}\end{bmatrix} + \begin{bmatrix}\epsilon''\\ 0\\ \dots\\ 0\\ 0\end{bmatrix}.$$

For convenience, we define the coefficient matrix as $\mathbf{Q}$. We then show that the maximum eigenvalue of $\mathbf{Q}$ satisfies $\max(\lambda_{\mathbf{Q}}|) = 1$ after some algebraic manipulations. Since $\|\epsilon''\| \leq \frac{2\eta M}{N}$, for $mod(k,q) \neq 0$ we have: $\mathbb{E}[\|\delta_{k+1}\|] \leq \mathbb{E}[\|\delta_k\|] + \frac{2\eta M}{N}$. For those k-values that satisfy $mod(k,q) = 0$, we can also show $\|\delta_{k+1}\| = \|\mathbf{x}_{k+1} - \mathbf{x}'_{k+1}\| \leq \|\delta_k\| + \frac{2\eta M}{N}$. By applying this bound inductively, we can bound δ_{k+1} using the total number K iterations as:

$$\mathbb{E}[\|\delta_{k+1}\|] \leq \frac{2\eta MK}{N}. \tag{14}$$

Step 6) Bounding the ϵ'-stability of SYNTHESIS in the quadratic strongly-convex case: Lastly, from the definition of algorithmic stability and the M-Lipschitz assumption of the loss function, our SYNTHESIS algorithm has the following stability bound:

$$\epsilon' \leq M \cdot \mathbb{E} \parallel \delta_{K+1} \parallel \leq \frac{2\eta M^2 K}{N}. \tag{15}$$

This completes the proof of Theorem 2. □

5 SYNTHESIS: SHARED MEMORY

In this section, we turn our attention to the SYNTHESIS algorithm for the shared memory architecture. We will 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 SYNTHESIS for the shared memory architecture in Section 5.3.

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 mini-batch S of samples and locally computes a stochastic gradient

$$\frac{1}{|S|}\sum_{i\in S}(\nabla f(\mathbf{x}_{new}, \xi_i) - \nabla f(\mathbf{x}_{old}, \xi_i) + \mathbf{v}_{old}).$$

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 turn out to be "delayed." To avoid race conditions in the shared memory, only a single coordinate of $\mathbf{x}$ can be updated at a time [20]. Let $[\mathbf{a}]_i$ represent the i-th index of vector $\mathbf{a}$. Hence, in each iteration, the update of parameter $\mathbf{x}$ can be written 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, which can also be written as following similar arguments as in Eq. (1):

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

Algorithm 3: The SYNTHESIS Algorithm for the Shared Memory Architecture.

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

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 $(\mathbf{x}_{k+1})_{m_k} = (\mathbf{x}_k)_{m_k} - \eta(\mathbf{v}_{old})_{m_k}$.

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

We illustrate the SYNTHESIS algorithm for the shared memory architecture in Algorithm 3. Note that, compared to the shared-memory SYNTHESIS 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 executes the code; and ii) only one coordinate can be updated at a time to avoid race conditions (cf. Step 13).

5.2 Convergence Performance Analysis

In this subsection, we first present the main convergence result of SYNTHESIS under the shared memory architecture in Theorem 3. Due to the similarity to the proof of Theorem 1 and space limitation, we omit the proof here and relegate the proof of Theorem 3 to our online technical report [16].

Theorem 3 (Convergence of SYNTHESIS for Shared Memory). *Let f^* denote the global optimal value. Under shared memory and assumptions 1-6, for some $\epsilon > 0$, by choosing parameters $q = |S| = \sqrt{N}$ and $\eta \leq \frac{1}{2L(\Delta+1)}$ for a given maximum delay $\Delta > 0$, the SYNTHESIS algorithm outputs an $\mathbf{x}_\zeta$ that satisfies $\mathbb{E}[\|\nabla f(\mathbf{x}_\zeta)\|^2] \leq \epsilon^2$ if $K = O\big(\frac{f(\mathbf{x}_0)-f^*}{\epsilon^2}\big)$, which also implies an $O(\sqrt{N}\epsilon^{-2}(\Delta+1)d + N)$ total SFO complexity.*

Remark 3. The computational complexity results in Theorem 3 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 in Theorem 3.

5.3 Generalization Performance Analysis

In this section, our goal is to study the algorithmic stability of SYNTHESIS under the shared memory architecture. Recall that the update rule of SYNTHESIS 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 the shared memory architecture, we obtain the following main result on the algorithmic stability (or equivalently, generalization error) for SYNTHESIS :

Theorem 4 (Algorithm Stability). *Under assumptions 1-6, the shared-memory version of the SYNTHESIS algorithm is $((2\eta M^2 K)/\sqrt{d} + (2M^2K\eta)/N\sqrt{d})$-uniformly stable. In addition to the above conditions, suppose that the loss function $f(x; \xi_i)$ is quadratic and μ -strongly convex. Then, the shared-memory version of the SYNTHESIS algorithm is $((2\eta M^2 K)/N\sqrt{d}))$-uniformly stable.*

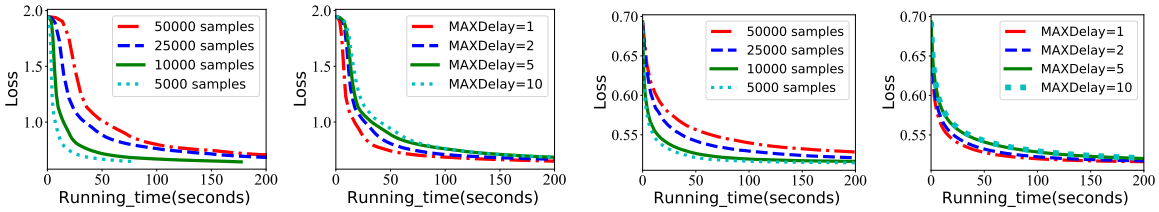
Remark 4. Theorems 4 provides the algorithmic stability results for both quadratic strongly-convex and non-convex settings. The proof techniques are similar to those of Theorem 2, and we omit the proofs here for brevity. We can see that the stability error bounds in Theorems 4 have an extra $(1/\sqrt{d})$-factor compared to those in Theorems 2. This is because the shared memory architecture only allows reading and writing a single coordinate of $\mathbf{x}$ at a time to avoid race conditions. Therefore, we have

$$\mathbb{E}[\|[\nabla f(\mathbf{x}'_{k-\tau(k)}, \xi_i)]_{m_k} - [\nabla f(\mathbf{x}'_{k-\tau(k)}, \xi'_i)]_{m_k}\|] \leq \frac{2M}{\sqrt{d}},$$

hence the existence of an extra $(1/\sqrt{d})$-factor. This extra $(1/\sqrt{d})$-factor in the denominator implies that, as the larger the dimensionality d increases, the generalization performance of our SYNTHESIS algorithm becomes better in the shared memory setting.

6 NUMERICAL RESULTS

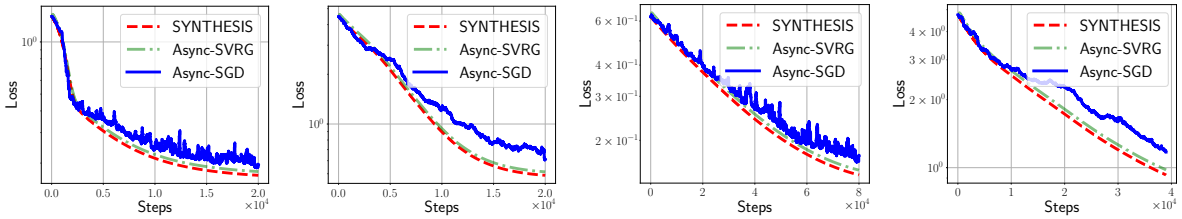
In this section, we conduct numerical experiments to verify our theoretical findings of the SYNTHESIS algorithms under distributed memory and shared memory architectures, respectively. First, we will test the convergence performance of SYNTHESIS under the distributed memory architecture and the shared memory architecture. Then, we will illustrate the generalization performance via simple logistic regression using the Breast-Cancer-Wisconsin dataset. In particular, we evaluate and compare the generalization and convergence performance metrics with two state-of-the-art algorithms in asynchronous first-order methods:



(a) Performance on the CIFAR-10 under distr.-mem. system.

(b) Performance on the MNIST under the shared-mem. system.

Figure 4: The convergence performance of SYNTHESIS .



(a) CIFAR-10 under distr.-mem. system. (b) MNIST under distr.-mem. system. (c) CIFAR-10 under shared-mem. system. (d) MNIST under shared-mem. system.

Figure 5: The convergence performance of different algorithms.

- *Async-SGD* [31]: This algorithm has a single-loop architecture, where each worker randomly picks a 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 maximum delay due to asynchrony is chosen to be the same Δ as that in the simulation of our SYNTHESIS algorithm.
- *Async-SVRG* [10]: 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.

1) Experiment Settings: For the distributed memory architecture, we conduct experiments on the Amazon Web Service (AWS) platform. There are four workers and one parameter server. For the shared memory architecture, we conduct experiments on a single machine with multiple cores. We leverage *OpenMPI* (Message Passing Interface) to facilitate the shared memory parallelism. There are four threads in the platform working in this experiment.

We use a pre-trained convolutional neural network (CNN) model with the CIFAR-10 dataset [1] and MNIST dataset [13]. We randomly select a subset of dataset with a varying size N as the training dataset and fix the step-size at $\eta = 0.1$ and choose the mini-batch size as $\lceil\sqrt{N}\rceil$. In the CIFAR-10 dataset, each sample is of dimension $d = 384$. Thus, we construct a fully connected three-layer ($384 \times 100 \times 10$) neural network with the ReLU activation function and the Softmax loss function while training CIFAR-10. In the MNIST dataset, each sample is a vector of dimension 784. Thus, we build a fully connected three-layer ($784 \times 100 \times 10$) neural network with the ReLU activation and the Softmax loss for MNIST.

2) Numerical Results: We first examine the convergence of the training loss value of the SYNTHESIS algorithms with different training dataset size N and different values of maximum delay Δ on each worker. Fig. 4(a) illustrates the convergence performance of training loss value with respect to the running time of the distributed-memory version of the SYNTHESIS algorithm with different N and Δ values, while Fig. 4(b) illustrates the performance of the shared-memory version of SYNTHESIS. It can be seen that as N or Δ decreases, the convergence rates of both distributed-memory and shared-memory versions of the SYNTHESIS algorithm increase. This is consistent with our theoretical algorithm complexity results. Note that the above algorithm complexity results imply 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 of Δ results in more stale stochastic gradient information, and thus inducing a negative impact on the convergence rate performance. Fig. 5 illustrate the algorithms comparison results under both distributed and shared-memory architectures. We plot and compare the performances of Async-SGD, Async-SVRG, and SYNTHESIS . We choose the same starting points for all algorithms, which are generated randomly from a normal distribution. Specifically, we choose a constant step-sizes 10^{-2}, training sample size $N = 5000$, and mini-batch S with $|S| = \lceil\sqrt{N}\rceil$. Fig. 5 show that SYNTHESIS has a faster convergence than both Async-SGD and Async-SVRG. In the CIFAR-10 dataset case under distributed memory system, we can see that SYNTHESIS needs around 1.5×10^4 steps to achieve the same loss as the Async-SVRG algorithm after running 2×10^4 steps. Since we use the same length of training

Table 1: Algorithms stability comparison.

System	Async-SGD	Async-SVRG	SYNTHESIS
Distributed Mem.	6.0×10^{-4}	7.6×10^{-4}	8.3×10^{-4}
Shared Mem.	5.1×10^{-4}	6.2×10^{-4}	6.7×10^{-4}

window and batch size in SYNTHESIS and Async-SVRG, SYNTHESIS has lower sample complexity than Async-SVRG. Similar results can be concluded in other cases in Fig. 5.

For the algorithms generalization performance, we set datasets that differ only by one sample based on the definition of algorithmic stability. To do so, we first choose a subset S of samples from the entire dataset and 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 $\mathbf{x}'_k$. Table. 1 illustrates the algorithmic stability results of different algorithms with 10^3 training iterations. Table 1 shows a slightly worse stability performance (an approximate 10^{-4} gap with 10^3 iterations) than SGD and a similar stability performance as that of Aysnc-SVRG. The results show that Async-SGD has the smallest bound on the algorithmic stability, implying better generalization. This is consistent with the widely observed generalization robustness of SGD (see, e.g., [3]). Moreover, the performance of SYNTHESIS under distributed memory is worse than that of the shared memory architecture, which again confirms our theoretical findings.

7 CONCLUSION

In this paper, we proposed an algorithm called SYNTHESIS (semi-asynchronous path-integrated stochastic gradient search) for non-convex distributed learning under distributed memory and shared memory architectures in computing clusters. We showed that our SYNTHESIS 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 SYNTHESIS 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. Collectively, our results in this work advance the understanding of the convergence and generalization performances of semi-asynchronous distributed first-order variance-reduced methods.

ACKNOWLEDGMENTS

This work has been supported in part by NSF grants CAREER CNS-2110259, CNS-2112471, CNS-2102233, and CCF-2110252.

REFERENCES

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

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

[3] Azizan, N., Lale, S., and Hassibi, B. Stochastic mirror descent on overparameterized nonlinear models: Convergence, implicit regularization, and generalization. *arXiv:1906.03830* (2019).

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

[5] Defazio, A., Bach, F., and Lacoste-Julien, S. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems* (2014), pp. 1646–1654.

[6] Fang, C., Li, C. J., Lin, Z., and Zhang, T. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems* (2018), pp. 689–699.

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

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

[9] Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning* (2016), pp. 1225–1234.

[10] Huo, Z., and Huang, H. Asynchronous stochastic gradient descent with variance reduction for non-convex optimization. *arXiv preprint arXiv:1604.03584* (2016).

[11] Johnson, R., and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems* (2013), pp. 315–323.

[12] Johnson, R., and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems* (2013), pp. 315–323.

[13] LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. *Available: http://yann. lecun. com/exdb/mnist* (1998).

[14] Li, Z., Bao, H., Zhang, X., and Richtarik, P. Page: A simple and optimal probabilistic gradient estimator for nonconvex optimization. In *International Conference on Machine Learning* (2021), PMLR, pp. 6286–6295.

[15] Lian, X., Huang, Y., Li, Y., and Liu, J. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems* (2015), pp. 2737–2745.

[16] Liu, Z., Zhang, X., and Liu, J. Synthesis: A semi-asynchronous path-integrated stochastic gradient method for distributed learning in computing clusters. https://kevinliu-osu.github.io/publications/SYNTHESIS_TR.pdf.

[17] Nguyen, L. M., Liu, J., Scheinberg, K., and Takavc, M. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning* (2017), PMLR, pp. 2613–2621.

[18] Peng, Z., Xu, Y., Yan, M., and Yin, W. Arock: an algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing 38*, 5 (2016), A2851–A2879.

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

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

[21] Reddi, S. J., Hefny, A., Sra, S., Poczos, B., and Smola, A. J. On variance reduction in stochastic gradient descent and its asynchronous variants. In *Advances in Neural Information Processing Systems* (2015), pp. 2647–2655.

[22] Robbins, H., and Monro, S. A stochastic approximation method. *The annals of mathematical statistics* (1951), 400–407.

[23] Roux, N. L., Schmidt, M., and Bach, F. R. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in neural information processing systems* (2012), pp. 2663–2671.

[24] Schmidt, M., Le Roux, N., and Bach, F. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming 162*, 1-2 (2017), 83–112.

[25] Tang, L., Harrington, P., and Zhu, T. Providing personalized item recommendations using scalable matrix factorization with randomness, Feb. 19 2015. US Patent App. 13/970,271.

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

[27] Wang, Z., Ji, K., Zhou, Y., Liang, Y., and Tarokh, V. Spiderboost: A class of faster variance-reduced algorithms for nonconvex optimization. *arXiv preprint arXiv:1810.10690* (2018).

[28] Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems* (2017), pp. 1509–1519.

[29] Yao, Q., and Kwok, J. Scalable robust matrix factorization with nonconvex loss. In *Advances in Neural Information Processing Systems* (2018), pp. 5061–5070.

[30] Zhang, R., and Kwok, J. Asynchronous distributed admm for consensus optimization. In *International conference on machine learning* (2014), pp. 1701–1709.

[31] Zhang, X., Liu, J., and Zhu, Z. Taming convergence for asynchronous stochastic gradient descent with unbounded delay in non-convex learning. In *Proc. IEEE CDC* (2020).

[32] Zhao, S.-Y., and Li, W.-J. Fast asynchronous parallel stochastic gradient descent: A lock-free approach with convergence guarantee. In *Thirtieth AAAI conference on artificial intelligence* (2016).

A CONVERGENCE ANALYSIS OF SA-SPIDERBOOST FOR DISTRIBUTED MEMORY

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

Lemma 1. *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*

$$\beta_1 \triangleq \frac{\eta}{2} - \frac{L\eta^2}{2} - L^2\eta^3\big(\frac{q(\Delta+1)}{|S|} + \Delta^2\big) > 0, \tag{16}$$

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

$$\mathbb{E}\big\|\mathbf{v}_k - \nabla f(\mathbf{x}_k)\big\|^2 \le \epsilon_1^2, \tag{17}$$

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

$$\mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta)\big\|^2 \le \big[\frac{2}{\beta_1}(\eta + 2L^2(\Delta^2 + \frac{q(\Delta+1)}{|S|})\eta^3) + 4\big]\epsilon_1^2 + \big[\frac{4L^2(\Delta^2 + \frac{q(\Delta+1)}{|S|})\eta^2}{K\beta_1} + \frac{2}{K\beta_1}\big](f(\mathbf{x}_0) - f^*). \tag{18}$$

A.1 Proof of Lemma 1

Proof. In Lemma 1, 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:

$$\mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta)\big\|^2 = \mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta) - \mathbf{v}_\zeta + \mathbf{v}_\zeta\big\|^2 \le 2\mathbb{E}\big\|\mathbf{v}_\zeta\big\|^2 + 2\mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta) - \mathbf{v}_\zeta\big\|^2. \tag{19}$$

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 (19) 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:

$$\begin{aligned}
&\mathbb{E}\big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_k, \xi_i)\big\|^2 \\
&\overset{(a)}{\le} 2\mathbb{E}\big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i)\big\|^2 + 2\mathbb{E}\big\|\frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i) - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_k, \xi_i)\big\|^2 \\
&\overset{(b)}{\le} 2\mathbb{E}\big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i)\big\|^2 + 2L^2\mathbb{E}\big\|\mathbf{x}_{k-\tau(k)} - \mathbf{x}_k\big\|^2 \\
&= 2\mathbb{E}\big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i)\big\|^2 + 2L^2\mathbb{E}\big\|\sum_{j=k-\tau(k)}^{k-1}(\mathbf{x}_{j+1} - \mathbf{x}_j)\big\|^2 \\
&\overset{(c)}{\le} 2\mathbb{E}\big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i)\big\|^2 + 2L^2\Delta\sum_{j=k-\tau(k)}^{k-1}\mathbb{E}\big\|(\mathbf{x}_{j+1} - \mathbf{x}_j)\big\|^2 \\
&\overset{(d)}{=} 2\mathbb{E}\big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i)\big\|^2 + 2L^2\eta^2\Delta\sum_{j=k-\tau(k)}^{k-1}\mathbb{E}\big\|\mathbf{v}_j\big\|^2,
\end{aligned} \tag{20}$$

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 (20), which denotes the distance of the inner loop. Let $n_k = \lceil k/q \rceil$ such that $(n_k - 1)q \le k \le n_k q - 1$, we have

$$\begin{aligned}
&\mathbb{E}\big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^N \nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i)\big\|^2 \\
&\overset{(a)}{=} \mathbb{E}\big\|\frac{1}{|S|}\sum_{i\in S}\big[\nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i) - \nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}}, \xi_i) - \frac{1}{N}\sum_{j=1}^N \nabla f(\mathbf{x}_{k-\tau(k)}, \xi_j) \\
&\quad + \frac{1}{N}\sum_{j=1}^N \nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}}, \xi_j)\big]\big\|^2 + \mathbb{E}\big\|\frac{1}{|S|}\sum_{i\in S}\big[\mathbf{v}_{k-1-\tau(k)} - \frac{1}{N}\sum_{j=1}^N \nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}}, \xi_j)\big]\big\|^2 \\
&\overset{(b)}{\le} \frac{1}{|S|^2}\sum_{i\in S}\mathbb{E}\big\|\big[\nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i) - \nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}}, \xi_i)\big] - \frac{1}{N}\sum_{j=1}^N \nabla f(\mathbf{x}_{k-\tau(k)}, \xi_j)
\end{aligned}$$

$$
\begin{aligned}
&+\frac{1}{N}\sum_{j=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_j)\big\|^2+\mathbb{E}\big\|\mathbf{v}_{k-1-\tau(k)}-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_i)\big\|^2\\
&\overset{(c)}{\leq}\frac{1}{|S|^2}\sum_{i\in S}\mathbb{E}\big\|\nabla f(\mathbf{x}_{k-\tau(k)},\xi_i)-\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_i)\big\|^2\\
&\quad+\mathbb{E}\big\|\mathbf{v}_{k-1-\tau(k)}-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_i)\big\|^2\\
&\overset{(d)}{\leq}\frac{L^2}{|S|^2}\sum_{i\in S}\mathbb{E}\big\|\mathbf{x}_{k-\tau(k)}-\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}}\big\|^2+\mathbb{E}\big\|\mathbf{v}_{k-1-\tau(k)}-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_i)\big\|^2\\
&\overset{(e)}{=}\frac{L^2\eta^2}{|S|}\mathbb{E}\big\|\sum_{i=k-\tau(k)-1-\tau(k-\tau(k)-1)}^{k-\tau(k)-1}\mathbf{v}_i\big\|^2+\mathbb{E}\big\|\mathbf{v}_{k-1-\tau(k)}-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_i)\big\|^2\\
&\overset{(f)}{\leq}\frac{L^2\eta^2}{|S|}(\Delta+1)\sum_{i=k-\tau(k)-1-\tau(k-\tau(k)-1)}^{k-\tau(k)-1}\mathbb{E}\big\|\mathbf{v}_i\big\|^2+\mathbb{E}\big\|\mathbf{v}_{k-1-\tau(k)}-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_i)\big\|^2,
\end{aligned}
\tag{21}
$$

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 [6], (b) and (c) use in [6, 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:

$$
\begin{aligned}
\mathbb{E}\big\|\mathbf{v}_k-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau(k)},\xi_i)\big\|^2\leq&\frac{L^2\eta^2(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k-1-\tau(k)}\mathbb{E}\big\|\mathbf{v}_j\big\|^2\\
&+\mathbb{E}\big\|\mathbf{v}_{(n_k-1)q}-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{(n_k-1)q},\xi_i)\big\|^2.
\end{aligned}
\tag{22}
$$

By combining (20) and (22), we arrive at:

$$
\begin{aligned}
&\mathbb{E}\big\|\mathbf{v}_k-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_k,\xi_i)\big\|^2\\
&\leq\frac{2L^2\eta^2(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k-1-\tau(k)}\mathbb{E}\big\|\mathbf{v}_j\big\|^2+2\mathbb{E}\big\|\mathbf{v}_{(n_k-1)q}-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{(n_k-1)q},\xi_i)\big\|^2\\
&+2L^2\eta^2\Delta\sum_{j=k-\tau(k)}^{k-1}\mathbb{E}\big\|\mathbf{v}_j\big\|^2.
\end{aligned}
\tag{23}
$$

Next, we continue to bound the other term $\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2$ on the right-hand-side of (19). By Assumption 5, the entire objective function f is $L-$smooth, which further implies

$$
\begin{aligned}
f(\mathbf{x}_{k+1})&\leq f(\mathbf{x}_k)+\langle\nabla f(\mathbf{x}_k),\mathbf{x}_{k+1}-\mathbf{x}_k\rangle+\frac{L}{2}\big\|\mathbf{x}_{k+1}-\mathbf{x}_k\big\|^2\\
&\overset{(a)}{=}f(\mathbf{x}_k)-\eta\langle\nabla f(\mathbf{x}_k),\mathbf{v}_k\rangle+\frac{L\eta^2}{2}\big\|\mathbf{v}_k\big\|^2\\
&=f(\mathbf{x}_k)-\eta\langle\nabla f(\mathbf{x}_k)-\mathbf{v}_k,\mathbf{v}_k\rangle-\eta\big\|\mathbf{v}_k\big\|^2+\frac{L\eta^2}{2}\big\|\mathbf{v}_k\big\|^2\\
&\overset{(b)}{\leq}f(\mathbf{x}_k)+\frac{\eta}{2}\big\|\mathbf{v}_k-\nabla f(\mathbf{x}_k)\big\|^2-(\frac{\eta}{2}-\frac{L\eta^2}{2})\big\|\mathbf{v}_k\big\|^2,
\end{aligned}
\tag{24}
$$

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

$$\begin{aligned}
\mathbb{E}f(\mathbf{x}_{k+1}) \leq & \mathbb{E}f(\mathbf{x}_k) + \frac{\eta}{2}\mathbb{E}\left\|\mathbf{v}_k - \nabla f(\mathbf{x}_k)\right\|^2 - (\frac{\eta}{2} - \frac{L\eta^2}{2})\mathbb{E}\left\|\mathbf{v}_k\right\|^2 \\
\overset{(a)}{\leq} & \frac{\eta}{2}\Big(\frac{2L^2\eta^2(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k-1-\tau(k)}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 + 2\mathbb{E}\left\|\mathbf{v}_{(n_k-1)q} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{(n_k-1)q}, \xi_i)\right\|^2 \\
& + 2L^2\eta^2\Delta\sum_{j=k-\tau(k)}^{k-1}\mathbb{E}\left\|\mathbf{v}_j\right\|^2\Big) - (\frac{\eta}{2} - \frac{L\eta^2}{2})\mathbb{E}\left\|\mathbf{v}_k\right\|^2 + \mathbb{E}f(\mathbf{x}_k) \\
\overset{(b)}{\leq} & \mathbb{E}f(\mathbf{x}_k) - (\frac{\eta}{2} - \frac{L\eta^2}{2})\mathbb{E}\left\|\mathbf{v}_k\right\|^2 + \eta\epsilon_1^2 + \frac{L^2\eta^3(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k-1-\tau(k)}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 + L^2\eta^3\Delta\sum_{j=k-\tau(k)}^{k-1}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 \\
\leq & \mathbb{E}f(\mathbf{x}_k) - (\frac{\eta}{2} - \frac{L\eta^2}{2})\mathbb{E}\left\|\mathbf{v}_k\right\|^2 + \eta\epsilon_1^2 + \frac{L^2\eta^3(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 + L^2\eta^3\Delta\sum_{j=k-\tau(k)}^{k-1}\mathbb{E}\left\|\mathbf{v}_j\right\|^2, \qquad (25)
\end{aligned}$$

where (a) follows from (20), (b) 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 (25) over k from $(n_k-1)q$ to k, where $k \leq n_k q - 1$, we have:

$$\begin{aligned}
\mathbb{E}f(\mathbf{x}_{k+1}) \overset{(a)}{\leq} & \mathbb{E}f(\mathbf{x}_{(n_k-1)q}) - (\frac{\eta}{2} - \frac{L\eta^2}{2})\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + \sum_{i=(n_k-1)q}^{k}\eta\epsilon_1^2 \\
& + \frac{L^2\eta^3(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k}\sum_{i=(n_k-1)q}^{j}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + L^2\eta^3\Delta\sum_{j=(n_k-1)q}^{k}\sum_{i=j-\tau^j}^{j-1}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 \\
\overset{(b)}{\leq} & \mathbb{E}f(\mathbf{x}_{(n_k-1)q}) - (\frac{\eta}{2} - \frac{L\eta^2}{2})\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + \sum_{i=(n_k-1)q}^{k}\eta\epsilon_1^2 \\
& + \frac{L^2\eta^3(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k}\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + L^2\eta^3\Delta^2\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2, \qquad (26)
\end{aligned}$$

where (a) follows from (25), (b) extends the summation of second term from j to k. Further relaxing (26) yields:

$$\begin{aligned}
\mathbb{E}f(\mathbf{x}_{k+1}) \overset{(a)}{\leq} & \mathbb{E}f(\mathbf{x}_{(n_k-1)q}) - (\frac{\eta}{2} - \frac{L\eta^2}{2})\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + \sum_{i=(n_k-1)q}^{k}\eta\epsilon_1^2 \\
& + \frac{qL^2\eta^3(\Delta+1)}{|S|}\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 + L^2\eta^3\Delta^2\sum_{j=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 \\
= & \mathbb{E}f(\mathbf{x}_{(n_k-1)q}) + \sum_{i=(n_k-1)q}^{k}\eta\epsilon_1^2 - \Big[\frac{\eta}{2} - \frac{L\eta^2}{2} - L^2\eta^3\big(\frac{q(\Delta+1)}{|S|} + \Delta^2\big)\Big]\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 \\
= & \mathbb{E}[f(\mathbf{x}_{(n_k-1)q})] - \sum_{i=(n_k-1)q}^{k}(\beta_1\mathbb{E}\left\|\mathbf{v}_i\right\|^2 - \eta\epsilon_1^2), \qquad (27)
\end{aligned}$$

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

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

$$= -\sum_{i=0}^{K-1}(\beta_1\mathbb{E}\|\mathbf{v}_i\|^2) + K\eta\epsilon_1^2, \tag{28}$$

where (a) follows from (27). Since $\mathbb{E}f(\mathbf{x}_K) \geq f(x^*)$, then we have:

$$\mathbb{E}f(\mathbf{x}^*) - \mathbb{E}f(\mathbf{x}_0) \leq -\sum_{i=0}^{K-1}(\beta_1\mathbb{E}\|\mathbf{v}_i\|^2) + K\eta\epsilon_1^2. \tag{29}$$

By rearranging (29), we have:

$$\mathbb{E}\|\mathbf{v}_\zeta\|^2 = \frac{1}{K}\sum_{i=0}^{K-1}\mathbb{E}\|\mathbf{v}_i\|^2 \leq \frac{f(\mathbf{x}_0)-f^*}{K\beta_1} + \frac{\eta}{\beta_1}\epsilon_1^2, \tag{30}$$

It then follows that:

$$\begin{aligned}
\mathbb{E}\|\mathbf{v}_\zeta - \nabla f(\mathbf{x}_\zeta)\|^2 &\overset{(a)}{\leq} \mathbb{E}\frac{2L^2\eta^2(\Delta+1)}{|S|}\sum_{j=(n_\zeta-1)q}^{\zeta-1-\tau^\zeta}\mathbb{E}\|\mathbf{v}_j\|^2 + 2\mathbb{E}\|\mathbf{v}_{(n_\zeta-1)q} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{(n_\zeta-1)q}, \xi_i)\|^2 \\
&\quad + 2L^2\eta^2\Delta\mathbb{E}\sum_{j=\zeta-\tau^\zeta}^{\zeta-1}\mathbb{E}\|\mathbf{v}_j\|^2 \\
&\overset{(b)}{\leq} \frac{2L^2\eta^2(\Delta+1)}{|S|}\mathbb{E}\sum_{j=(n_\zeta-1)q}^{\zeta-1-\tau^\zeta}\mathbb{E}\|\mathbf{v}_j\|^2 + 2\epsilon_1^2 + 2L^2\eta^2\Delta\mathbb{E}\sum_{i=\zeta-\tau^\zeta}^{\zeta-1}\mathbb{E}\|\mathbf{v}_i\|^2 \\
&\leq \frac{2L^2\eta^2(\Delta+1)}{|S|}\mathbb{E}\sum_{j=(n_\zeta-1)q}^{\xi}\mathbb{E}\|\mathbf{v}_j\|^2 + 2\epsilon_1^2 + 2L^2\eta^2\Delta\mathbb{E}\sum_{i=\zeta-\tau^\zeta}^{\zeta-1}\mathbb{E}\|\mathbf{v}_i\|^2 \\
&\overset{(c)}{\leq} \frac{2L^2\eta^2(\Delta+1)}{|S|}\mathbb{E}\sum_{i=(n_\zeta-1)q}^{min\{(n_\zeta)q-1,K-1\}}\mathbb{E}\|\mathbf{v}_i\|^2 + 2\epsilon_1^2 + 2L^2\eta^2\Delta\mathbb{E}\sum_{i=\zeta-\tau^\zeta}^{\zeta-1}\mathbb{E}\|\mathbf{v}_i\|^2 \\
&\overset{(d)}{\leq} \frac{q}{K}\sum_{i=0}^{K-1}\frac{2L^2\eta^2(\Delta+1)}{|S|}\mathbb{E}\|\mathbf{v}_i\|^2 + 2\epsilon_1^2 + \frac{\Delta}{K}\sum_{i=0}^{K-1}2L^2\eta^2\Delta\mathbb{E}\|\mathbf{v}_i\|^2 \\
&= (\frac{2L^2\eta^2q(\Delta+1)}{|S|} + 2L^2\eta^2\Delta^2)\frac{1}{K}\sum_{i=0}^{K-1}\mathbb{E}\|\mathbf{v}_i\|^2 + 2\epsilon_1^2 \\
&\overset{(e)}{\leq} \Big[\frac{2L^2(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^3}{\beta_1} + 2\Big]\epsilon_1^2 + 2L^2(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^2\Big[\frac{f(\mathbf{x}_0)-f^*}{K\beta_1}\Big],
\end{aligned} \tag{31}$$

where (a) follows from (23), (b) follows from (17), (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 (30).

By combining (30) and (31), we arrive at:

$$\begin{aligned}
&\mathbb{E}\|\nabla f(\mathbf{x}_\zeta)\|^2 \leq 2\mathbb{E}\|\nabla f(\mathbf{x}_\zeta) - \mathbf{v}_\zeta\|^2 + 2\mathbb{E}\|\mathbf{v}_\zeta\|^2 \\
&\leq 2\Big\{\Big[\frac{2L^2(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^3}{\beta_1} + 2\Big]\epsilon_1^2 + 2L^2(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^2\Big[\frac{f(\mathbf{x}_0)-f^*}{K\beta_1}\Big]\Big\} \\
&\quad + 2[\frac{f(\mathbf{x}_0)-f^*}{K\beta_1} + \frac{\eta}{\beta_1}\epsilon_1^2] \\
&= \Big[\frac{2}{\beta_1}(\eta + 2L^2(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^3) + 4\Big]\epsilon_1^2 + \Big[\frac{4L^2(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^2}{K\beta_1} + \frac{2}{K\beta_1}\Big](f(\mathbf{x}_0)-f^*).
\end{aligned} \tag{32}$$

This completes the proof of Lemma 1. □

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

$$q = \sqrt{N}, \quad S = \sqrt{N}, \quad \eta = \frac{1}{4L(\Delta+1)}, \tag{33}$$

we obtain:

$$\beta_1 \triangleq \frac{\eta}{2} - \frac{L\eta^2}{2} - L^2\eta^3\big(\frac{q(\Delta+1)}{|S|} + \Delta^2\big) = \frac{1}{8L(\Delta+1)} \cdot \frac{14\Delta^2 + 26\Delta + 10}{16(\Delta+1)^2} > 0. \tag{34}$$

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

$$\mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta)\big\|^2 \leq 16L(\Delta+1)\frac{9\Delta^2 + 17\Delta + 9}{K \cdot (7\Delta^2 + 13\Delta + 5)}(f(\mathbf{x}_0) - f^*). \tag{35}$$

To ensure $\mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta)\big\| \leq \epsilon$, it suffices to ensure$\mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta)\big\|^2 \leq \epsilon^2$. 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: $\lceil \frac{K}{q} \rceil N + K \cdot S \leq \frac{K+q}{q} N + K\sqrt{N} = K\sqrt{N} + N + K\sqrt{N} = O(\sqrt{N}\epsilon^{-2}(\Delta+1) + N)$. This completes the proof.

B GENERALIZATION ANALYSIS OF SA-SPIDERBOOST FOR DISTRIBUTED MEMORY

B.1 Proofs of Theorem 2

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

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \frac{1}{|S|} \sum_{i \in S} (\nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i) - \nabla f(\mathbf{x}_{k-\tau(k)-1-\tau(k-\tau(k)-1)}, \xi_i) + \mathbf{v}_{k-1-\tau(k)}). \tag{36}$$

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

$$\mathbb{E}(\delta_{k+1}) = \mathbb{E}(\mathbf{x}_{k+1}) - \mathbb{E}(\mathbf{x}_{k+1}') = \mathbb{E}(\delta_k) - \eta[\mathbb{E}(\mathbf{v}_k) - \mathbb{E}(\mathbf{v}_k')]. \tag{37}$$

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

$$\mathbb{E}(\delta_{k+1}) = \mathbb{E}(\delta_k) - \eta[\mathbb{E}(\nabla f(\mathbf{x}_{k-\tau(k)})) - \mathbb{E}(\nabla f(\mathbf{x}_{k-\tau(k)}'))]. \tag{38}$$

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:

$$\begin{aligned}
\mathbb{E}(\delta_{k+1}) \leq & \mathbb{E}\big[(\delta_k) - \eta\frac{1}{N}\sum_{i=1}^{N}\nabla f((\mathbf{x}_{k-\tau(k)}, \xi_i)) + \eta\frac{1}{N}\sum_{i=1}^{N}\nabla f((\mathbf{x}_{k-\tau(k)}', \xi_i))\big] \\
& + \mathbb{E}\big[\frac{1}{N}\eta\nabla f((\mathbf{x}_{k-\tau(k)}', \xi_i)) - \frac{1}{N}\eta\nabla f((\mathbf{x}_{k-\tau(k)}', \xi_i'))\big] \\
\overset{(a)}{\leq} & \mathbb{E}\big[(\delta_k) - \eta A(\delta_{k-\tau(k)})\big] + \epsilon'',
\end{aligned} \tag{39}$$

where (a) follows from f is a quadratic convex function. We note that $\|\epsilon''\| \leq \frac{2\eta M}{N}$ since we have the bounded gradient Assumption 4. Then, we have

$$\begin{bmatrix} \delta_{k+1} \\ \delta_k \\ \vdots \\ \delta_{k-\tau(k)+1} \\ \delta_{k-\tau(k)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & -\eta A & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} \delta_k \\ \delta_{k-1} \\ \vdots \\ \delta_{k-\tau(k)} \\ \delta_{k-\tau(k)-1} \end{bmatrix} + \begin{bmatrix} \epsilon'' \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}. \tag{40}$$

Let Matrix

$$\mathbf{Q} \triangleq \begin{bmatrix} 1 & 0 & \cdots & -\eta A & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}. \tag{41}$$

Consider the characteristic polynomial

$$\begin{bmatrix} 1 & 0 & \cdots & -\eta A & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \vdots \\ \mathbf{v}_{\tau(k)+2} \end{bmatrix} = \lambda_{\mathbf{Q}} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \vdots \\ \mathbf{v}_{\tau(k)+2} \end{bmatrix}, \tag{42}$$

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, we have:

$$\big(-\lambda_{\mathbf{Q}}^{\tau^k+2} + \lambda_{\mathbf{Q}}^{\tau^k+1} + \lambda_{\mathbf{Q}}[-\eta A]\big)\mathbf{v}_{\tau(k)+2} = 0. \tag{43}$$

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:

$$\mathbf{U}\big(-\lambda_{\mathbf{Q}}^{\tau^k+2} + \lambda_{\mathbf{Q}}^{\tau^k+1} + \lambda_{\mathbf{Q}}[-\eta \mathbf{A}]\big)\mathbf{U}^\top = 0. \tag{44}$$

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

$$\lambda_{\mathbf{Q}}^{\tau_k+1} \cdot \big[-\lambda_{\mathbf{Q}} + 1\big] = \lambda_{\mathbf{Q}}[\eta\lambda_i]. \tag{45}$$

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, $\max(|\lambda_{\mathbf{Q}}|) = 1$. Thus, for $mod(k, q) \neq 0$ we have

$$\mathbb{E}\|\delta_{k+1}\| \leq \mathbb{E}\|\delta_k\| + \frac{2\eta M}{N}. \tag{46}$$

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 (46), 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:

$$\| \delta_{k+1} \| \leq \frac{2\eta MK}{N}. \tag{47}$$

Lastly, it follows from the definition of algorithm stability and the M-Lipschitz Assumption 3 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 2

Recall that update rule for SpiderBoost is given by:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \frac{1}{|S|} \sum_{i \in S} (\nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i) - \nabla f(\mathbf{x}_{k-\tau(k)-1-\tau(k-\tau(k)-1)}, \xi_i) + \mathbf{v}_{k-1-\tau(k)}). \tag{48}$$

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

$$\mathbb{E}(\delta_{k+1}) = \mathbb{E}(\mathbf{x}_{k+1}) - \mathbb{E}(\mathbf{x}'_{k+1}) = \mathbb{E}(\delta_k) - \eta[\mathbb{E}(\mathbf{v}_k) - \mathbb{E}(\mathbf{v}'_k)]. \tag{49}$$

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

$$\mathbb{E}(\delta_{k+1}) = \mathbb{E}(\delta_k) - \eta[\mathbb{E}(\nabla f(\mathbf{x}_{k-\tau(k)})) - \mathbb{E}(\nabla f(\mathbf{x}'_{k-\tau(k)}))]. \tag{50}$$

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

$$\begin{aligned}
\mathbb{E}\|(\delta_{k+1})\| \leq & \mathbb{E}\|(\delta_k) - \eta(\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i)) + \eta(\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}'_{k-\tau(k)}, \xi_i))\| \\
& + \frac{1}{N}\eta\|\nabla f(\mathbf{x}'_{k-\tau(k)}, \xi_i)) - \nabla f(\mathbf{x}'_{k-\tau(k)}, \xi'_i))\| \\
\overset{(a)}{\leq} & \mathbb{E}\|(\delta_k) - \eta(\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i)) + \eta(\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}'_{k-\tau(k)}, \xi_i))\| + \frac{2\eta M}{N} \\
\overset{(b)}{\leq} & \mathbb{E}\|(\delta_k)\| + \eta\mathbb{E}\|(\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i)) - (\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}'_{k-\tau(k)}, \xi_i))\| + \frac{2\eta M}{N} \\
\overset{(c)}{\leq} & \mathbb{E}\|(\delta_k)\| + 2\eta M + \frac{2\eta M}{N},
\end{aligned} \tag{51}$$

where (a) and (b) follows from the triangle inequality, (c) follows from the bounded gradient Assumption 4. 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 4.

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 (51), 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:

$$\mathbb{E} \parallel \delta_{k+1} \parallel \leq 2\eta MK + \frac{2\eta MK}{N}. \tag{52}$$

Lastly, it follows from the definition of algorithm stability and the M-Lipschitz Assumption 3 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 2. *Let all assumptions hold and apply SA-SpiderBoost in Algorithm 3, if the parameters η, q and S are chosen such that*

$$\beta_1 \triangleq \frac{\eta}{2d} - \frac{L\eta^2}{2d} - \frac{L^2\eta^3}{d^2}\big(\frac{q(\Delta+1)}{|S|} + \Delta^2\big) > 0, \tag{53}$$

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

$$\mathbb{E}\big\|\mathbf{v}_k - \nabla f(\mathbf{x}_k)\big\|^2 \leq \epsilon_1^2, \tag{54}$$

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

$$\mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta)\big\|^2 \leq \big[\frac{2}{\beta_1 d}\big(\eta + \frac{2L^2}{d}(\Delta^2 + \frac{q(\Delta+1)}{|S|})\eta^3\big)\big]\epsilon_1^2 + \big[\frac{4L^2(\Delta^2 + \frac{q(\Delta+1)}{|S|})\eta^2}{K\beta_1 d} + \frac{2}{K\beta_1}\big](f(\mathbf{x}_0) - f^*). \tag{55}$$

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

$$\mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta)\big\|^2 = \mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta) - \mathbf{v}_\zeta + \mathbf{v}_\zeta\big\|^2 \leq 2\mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta) - \mathbf{v}_\zeta\big\|^2 + 2\mathbb{E}\big\|\mathbf{v}_\zeta\big\|^2. \tag{56}$$

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 (56) 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 $\mathbb{E}\big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i)\big\|^2$. Let $n_k = \lceil k/q \rceil$ such that $(n_k - 1)q \leq k \leq n_k q - 1$, we have:

$$\begin{aligned}
& \mathbb{E}\big\|\mathbf{v}_k - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i)\big\|^2 \\
\overset{(a)}{=} & \mathbb{E}\big\|\frac{1}{|S|}\sum_{i\in S}\big[\nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i) - \nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}}, \xi_i) - \frac{1}{N}\sum_{j=1}^{N}\nabla f(\mathbf{x}_{k-\tau(k)}, \xi_j)
\end{aligned}$$

$$
\begin{aligned}
&+\frac{1}{N}\sum_{j=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_j)\big]\big\|^2+\mathbb{E}\Big\|\frac{1}{|S|}\sum_{i\in S}\big[\mathbf{v}_{k-1-\tau(k)}-\frac{1}{N}\sum_{j=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_j)\big]\Big\|^2\\
&\overset{(b)}{\leq}\frac{1}{|S|^2}\sum_{i\in S}\mathbb{E}\Big\|\big[\nabla f(\mathbf{x}_{k-\tau(k)},\xi_i)-\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_i)\big]-\frac{1}{N}\sum_{j=1}^{N}\nabla f(\mathbf{x}_{k-\tau(k)},\xi_j)\\
&\quad+\frac{1}{N}\sum_{j=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_j)\Big\|^2+\mathbb{E}\Big\|\mathbf{v}_{k-1-\tau(k)}-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_i)\Big\|^2\\
&\overset{(c)}{\leq}\frac{1}{|S|^2}\sum_{i\in S}\mathbb{E}\Big\|\nabla f(\mathbf{x}_{k-\tau(k)},\xi_i)-\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_i)\Big\|^2\\
&\quad+\mathbb{E}\Big\|\mathbf{v}_{k-1-\tau(k)}-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_i)\Big\|^2\\
&\overset{(d)}{\leq}\frac{L^2}{|S|^2}\sum_{i\in S}\mathbb{E}\Big\|\mathbf{x}_{k-\tau(k)}-\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}}\Big\|^2+\mathbb{E}\Big\|\mathbf{v}_{k-1-\tau(k)}-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_i)\Big\|^2\\
&\overset{(e)}{=}\frac{L^2\eta^2}{|S|d}\mathbb{E}\Big\|\sum_{i=k-\tau(k)-1-\tau(k-\tau(k)-1)}^{k-\tau(k)-1}\mathbf{v}_i\Big\|^2+\mathbb{E}\Big\|\mathbf{v}_{k-1-\tau(k)}-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_i)\Big\|^2\\
&\overset{(f)}{\leq}\frac{L^2\eta^2}{|S|d}(\Delta+1)\sum_{i=k-\tau(k)-1-\tau(k-\tau(k)-1)}^{k-\tau(k)-1}\mathbb{E}\big\|\mathbf{v}_i\big\|^2+\mathbb{E}\Big\|\mathbf{v}_{k-1-\tau(k)}-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-1-\tau(k)-\tau^{k-1-\tau(k)}},\xi_i)\Big\|^2,
\end{aligned}
\tag{57}
$$

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 [6], (b) and (c)use A.3 in [6], (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:

$$
\begin{aligned}
\mathbb{E}\Big\|\mathbf{v}_k-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau(k)},\xi_i)\Big\|^2\leq\frac{L^2\eta^2(\Delta+1)}{|S|}\sum_{j=(n_k-1)q}^{k-1-\tau(k)}\mathbb{E}\big\|\mathbf{v}_j\big\|^2\\
+\mathbb{E}\Big\|\mathbf{v}_{(n_k-1)q}-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{(n_k-1)q},\xi_i)\Big\|^2.
\end{aligned}
\tag{58}
$$

Next, we can conclude that,

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

$$+\frac{2L^2\eta^2\Delta}{d}\sum_{j=k-\tau(k)}^{k-1}\mathbb{E}\left\|\mathbf{v}_j\right\|^2, \tag{59}$$

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 (58).

Next, we continue to bound the other term $\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2$ on the right-hand-side of (56). To evaluate $\mathbb{E}\left\|\mathbf{v}_\zeta\right\|^2$, we start from the iteration relationship of our SA-SpiderBoost algorithm. By Assumption 5, the entire objective function f is $L-$smooth, which further implies

$$\begin{aligned} f(\mathbf{x}_{k+1}) &\le f(\mathbf{x}_k)+\langle\nabla f(\mathbf{x}_k),\mathbf{x}_{k+1}-\mathbf{x}_k\rangle+\frac{L}{2d}\left\|\mathbf{x}_{k+1}-\mathbf{x}_k\right\|^2 \\ &\overset{(a)}{\le} f(\mathbf{x}_k)+\frac{\eta}{2d}\left\|\mathbf{v}_k-\nabla f(\mathbf{x}_k)\right\|^2-(\frac{\eta}{2d}-\frac{L\eta^2}{2d})\left\|\mathbf{v}_k\right\|^2, \end{aligned} \tag{60}$$

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

$$\begin{aligned} &\mathbb{E}f(\mathbf{x}_{k+1})\le\mathbb{E}f(\mathbf{x}_k)+\frac{\eta}{2}\mathbb{E}\left\|\mathbf{v}_k-\nabla f(\mathbf{x}_k)\right\|^2-(\frac{\eta}{2}-\frac{L\eta^2}{2})\mathbb{E}\left\|\mathbf{v}_k\right\|^2 \\ &\overset{(a)}{\le}\frac{\eta}{2d}\Big(\frac{2L^2\eta^2(\Delta+1)}{|S|d}\sum_{j=(n_k-1)q}^{k-1-\tau(k)}\mathbb{E}\left\|\mathbf{v}_j\right\|^2+2\mathbb{E}\left\|\mathbf{v}_{(n_k-1)q}-\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{(n_k-1)q},\xi_i)\right\|^2\Big) \\ &\quad+\frac{\eta}{2d}\Big(\frac{2L^2\eta^2\Delta}{d}\sum_{j=k-\tau(k)}^{k-1}\mathbb{E}\left\|\mathbf{v}_j\right\|^2\Big)-(\frac{\eta}{2d}-\frac{L\eta^2}{2d})\mathbb{E}\left\|\mathbf{v}_k\right\|^2+\mathbb{E}f(\mathbf{x}_k) \\ &\overset{(b)}{\le}\mathbb{E}f(\mathbf{x}_k)-(\frac{\eta}{2d}-\frac{L\eta^2}{2d})\mathbb{E}\left\|\mathbf{v}_k\right\|^2+\frac{\eta}{d}\epsilon_1^2+\frac{L^2\eta^3(\Delta+1)}{|S|d^2}\sum_{j=(n_k-1)q}^{k-1-\tau(k)}\mathbb{E}\left\|\mathbf{v}_j\right\|^2+\frac{L^2\eta^3\Delta}{d^2}\sum_{j=k-\tau(k)}^{k-1}\mathbb{E}\left\|\mathbf{v}_j\right\|^2 \\ &\le\mathbb{E}f(\mathbf{x}_k)-(\frac{\eta}{2d}-\frac{L\eta^2}{2d})\mathbb{E}\left\|\mathbf{v}_k\right\|^2+\frac{\eta}{d}\epsilon_1^2+\frac{L^2\eta^3(\Delta+1)}{|S|d^2}\sum_{j=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_j\right\|^2+\frac{L^2\eta^3\Delta}{d^2}\sum_{j=k-\tau(k)}^{k-1}\mathbb{E}\left\|\mathbf{v}_j\right\|^2, \end{aligned} \tag{61}$$

where $\overset{(a)}{\le}$ follows from (59), $\overset{(b)}{\le}$ follows from the $\mathbb{E}\left\|\mathbf{v}_{(n_k-1)q}-\nabla f(\mathbf{x}_{(n_k-1)q})\right\|^2\le\epsilon_1^2$.

Next, telescoping (61)over k from $(n_k-1)q$ to k where $k\le n_kq-1$, we have

$$\begin{aligned} \mathbb{E}f(\mathbf{x}_{k+1}) &\overset{(a)}{\le}\mathbb{E}f(\mathbf{x}_{(n_k-1)q})-(\frac{\eta}{2d}-\frac{L\eta^2}{2d})\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2+\sum_{i=(n_k-1)q}^{k}\frac{\eta}{d}\epsilon_1^2 \\ &\quad+\frac{L^2\eta^3(\Delta+1)}{|S|d^2}\sum_{j=(n_k-1)q}^{k}\sum_{i=(n_k-1)q}^{j}\mathbb{E}\left\|\mathbf{v}_i\right\|^2+\frac{L^2\eta^3\Delta}{d^2}\sum_{j=(n_k-1)q}^{k}\sum_{i=j-\tau^j}^{j-1}\mathbb{E}\left\|\mathbf{v}_i\right\|^2 \\ &\overset{(b)}{\le}\mathbb{E}f(\mathbf{x}_{(n_k-1)q})-(\frac{\eta}{2d}-\frac{L\eta^2}{2d})\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2+\sum_{i=(n_k-1)q}^{k}\frac{\eta}{d}\epsilon_1^2 \\ &\quad+\frac{L^2\eta^3(\Delta+1)}{|S|d^2}\sum_{j=(n_k-1)q}^{k}\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2+\frac{L^2\eta^3\Delta^2}{d^2}\sum_{i=(n_k-1)q}^{k}\mathbb{E}\left\|\mathbf{v}_i\right\|^2, \end{aligned} \tag{62}$$

where (a) follows from (61), (b) extends the summation of second term form j to k. It then follows that:

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

$$=\mathbb{E}[f(\mathbf{x}_{(n_k-1)q})] - \sum_{i=(n_k-1)q}^{k} (\beta_1 \mathbb{E}\|\mathbf{v}_i\|^2 - \frac{\eta}{d}\epsilon_1^2), \tag{63}$$

where (a) follows from (62). Then, we can further derive:

$$\begin{aligned}
&\mathbb{E}f(\mathbf{x}_K) - \mathbb{E}f(\mathbf{x}_0) = (\mathbb{E}f(\mathbf{x}_q) - \mathbb{E}f(\mathbf{x}_0)) + (\mathbb{E}f(\mathbf{x}_{2q}) - \mathbb{E}f(\mathbf{x}_q) + ... + (\mathbb{E}f(\mathbf{x}_K) - \mathbb{E}f(\mathbf{x}_{(n_k-1)q})))\\
&\overset{(a)}{\leq} -\sum_{i=0}^{q-1}(\beta_1 \mathbb{E}\|\mathbf{v}_i\|^2 - \frac{\eta}{d}\epsilon_1^2) - \sum_{i=q}^{2q-1}(\beta_1 \mathbb{E}\|\mathbf{v}_i\|^2 - \frac{\eta}{d}\epsilon_1^2) - ... - \sum_{(n_K-1)q}^{K-1}(\beta_1 \mathbb{E}\|\mathbf{v}_i\|^2 - \frac{\eta}{d}\epsilon_1^2)\\
&= -\sum_{i=0}^{K-1}(\beta_1 \mathbb{E}\|\mathbf{v}_i\|^2 - \frac{\eta}{d}\epsilon_1^2) = -\sum_{i=0}^{K-1}(\beta_1 \mathbb{E}\|\mathbf{v}_i\|^2) + \frac{K\eta}{d}\epsilon_1^2,
\end{aligned} \tag{64}$$

where (a) follows from eq.(63),. Since $\mathbb{E}f(\mathbf{x}_K) \geq f(x^*)$, then we have

$$\mathbb{E}f(x^*) - \mathbb{E}f(\mathbf{x}_0) \leq -\sum_{i=0}^{K-1}(\beta_1 \mathbb{E}\|\mathbf{v}_i\|^2) + \frac{K\eta}{d}\epsilon_1^2. \tag{65}$$

By rearranging (65), we have:

$$\mathbb{E}\|\mathbf{v}_\zeta\|^2 = \frac{1}{K}\sum_{i=0}^{K-1}\mathbb{E}\|\mathbf{v}_i\|^2 \leq \frac{f(\mathbf{x}_0) - f^*}{K\beta_1} + \frac{\eta}{d\beta_1}\epsilon_1^2. \tag{66}$$

It then follow that:

$$\begin{aligned}
\mathbb{E}\|\mathbf{v}_\zeta - \nabla f(\mathbf{x}_\zeta)\|^2 &\overset{(a)}{\leq} \mathbb{E}\frac{2L^2\eta^2(\Delta+1)}{|S|d}\sum_{j=(n_\zeta-1)q}^{\zeta-1-\tau^\zeta}\mathbb{E}\|\mathbf{v}_j\|^2 + 2\mathbb{E}\|\mathbf{v}_{(n_k-1)q} - \frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{(n_k-1)q}, \xi_i)\|^2\\
&+ \frac{2L^2\eta^2\Delta}{d}\mathbb{E}\sum_{j=\zeta-\tau^\zeta}^{\zeta-1}\mathbb{E}\|\mathbf{v}_j\|^2\\
&\overset{(b)}{\leq} \frac{2L^2\eta^2(\Delta+1)}{|S|d}\mathbb{E}\sum_{j=(n_\zeta-1)q}^{\zeta-1-\tau^\zeta}\mathbb{E}\|\mathbf{v}_j\|^2 + 2\epsilon_1^2 + \frac{2L^2\eta^2\Delta}{d}\mathbb{E}\sum_{i=\zeta-\tau^\zeta}^{\zeta-1}\mathbb{E}\|\mathbf{v}_i\|^2\\
&\leq \frac{2L^2\eta^2(\Delta+1)}{|S|d}\mathbb{E}\sum_{j=(n_\zeta-1)q}^{\xi}\mathbb{E}\|\mathbf{v}_j\|^2 + 2\epsilon_1^2 + \frac{2L^2\eta^2\Delta}{d}\mathbb{E}\sum_{i=\zeta-\tau^\zeta}^{\zeta-1}\mathbb{E}\|\mathbf{v}_i\|^2\\
&\overset{(c)}{\leq} \frac{2L^2\eta^2(\Delta+1)}{|S|d}\mathbb{E}\sum_{i=(n_\zeta-1)q}^{min\{(n_\zeta)q-1,K-1\}}\mathbb{E}\|\mathbf{v}_i\|^2 + 2\epsilon_1^2 + \frac{2L^2\eta^2\Delta}{d}\mathbb{E}\sum_{i=\zeta-\tau^\zeta}^{\zeta-1}\mathbb{E}\|\mathbf{v}_i\|^2\\
&\overset{(d)}{\leq} \frac{q}{K}\sum_{i=0}^{K-1}\frac{2L^2\eta^2(\Delta+1)}{|S|d}\mathbb{E}\|\mathbf{v}_i\|^2 + 2\epsilon_1^2 + \frac{\Delta}{K}\sum_{i=0}^{K-1}\frac{2L^2\eta^2\Delta}{d}\mathbb{E}\|\mathbf{v}_i\|^2\\
&= (\frac{2L^2\eta^2 q(\Delta+1)}{|S|d} + \frac{2L^2\eta^2\Delta^2}{d})\frac{1}{K}\sum_{i=0}^{K-1}\mathbb{E}\|\mathbf{v}_i\|^2 + 2\epsilon_1^2\\
&\overset{(e)}{\leq} \Big[\frac{2L^2(\Delta^2 + \frac{q(\Delta+1)}{|S|})\eta^3}{\beta_1 d^2} + 2\Big]\epsilon_1^2 + \frac{2L^2(\Delta^2 + \frac{q(\Delta+1)}{|S|})\eta^2}{d}\Big[\frac{f(\mathbf{x}_0) - f^*}{K\beta_1}\Big],
\end{aligned} \tag{67}$$

where (a) follows from (55), (b) follows from (61), (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 (68). Then we can obtain

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

$$=\Big[\frac{2}{\beta_1 d}\big(\eta+\frac{2L^2}{d}(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^3\big)+4\Big]\epsilon_1^2+\Big[\frac{4L^2(\Delta^2+\frac{q(\Delta+1)}{|S|})\eta^2}{K\beta_1 d}+\frac{2}{K\beta_1}\Big](f(\mathbf{x}_0)-f^*), \tag{68}$$

where (a) follows the (67) and (68). This completes the proof. □

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

$$q=\sqrt{N}, S=\sqrt{N}, \quad \eta=\frac{1}{2L(\Delta+1)}. \tag{69}$$

Then, we obtain:

$$\beta_1=\frac{\eta}{2d}-\frac{L\eta^2}{2d}-\frac{L^2\eta^3}{d^2}\big(\frac{q(\Delta+1)}{|S|}+\Delta^2\big)=\frac{1}{4Ld(\Delta+1)}\cdot\big(1-\frac{1}{2(\Delta+1)}-\frac{(1+\Delta+\Delta^2)}{2d(\Delta+1)^2}\big)>0 \tag{70}$$

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

$$\mathbb{E}\big\|\nabla f(\mathbf{x}_\zeta)\big\|^2\le\frac{8Ld(\Delta+1)}{K}\frac{2(\Delta+1)^2d+\Delta^2+\Delta+1}{2d(\Delta+1)^2-d(\Delta+1)-(\Delta^2+\Delta+1)}(f(\mathbf{x}_0)-f^*). \tag{71}$$

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

$$K=\frac{8Ld(\Delta+1)}{\epsilon^2}\frac{2(\Delta+1)^2d+\Delta^2+\Delta+1}{2d(\Delta+1)^2-d(\Delta+1)-(\Delta^2+\Delta+1)}(f(\mathbf{x}_0)-f^*)=O\big(\frac{f(\mathbf{x}_0)-f^*}{\epsilon^2}\big). \tag{72}$$

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 4

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

$$(\mathbf{x}_{k+1})_{m_k}=(\mathbf{x}_k)_{m_k}-\eta\frac{1}{|S|}\sum_{i\in S}(\nabla f(\mathbf{x}_{k-\tau(k)},\xi_i)-\nabla f(\mathbf{x}_{k-\tau(k)-1-\tau(k-\tau(k)-1)},\xi_i)+\mathbf{v}_{k-1-\tau(k)})_{m_k}. \tag{73}$$

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

$$\mathbb{E}(\delta_{k+1})_{m_k}=\mathbb{E}(\mathbf{x}_{k+1})_{m_k}-\mathbb{E}(\mathbf{x}_{k+1}')_{m_k}=\mathbb{E}(\delta_k)_{m_k}-\eta\big[\mathbb{E}(\mathbf{v}_k)_{m_k}-\mathbb{E}(\mathbf{v}_k')_{m_k}\big]. \tag{74}$$

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

$$\mathbb{E}(\delta_{k+1})_{m_k}=\mathbb{E}(\delta_k)_{m_k}-\eta[\mathbb{E}(\nabla f(\mathbf{x}_{k-\tau(k)}))_{m_k}-\mathbb{E}(\nabla f(\mathbf{x}_{k-\tau(k)}'))_{m_k}]. \tag{75}$$

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

$$\begin{aligned}\mathbb{E}(\delta_{k+1})_{m_k}\le&\mathbb{E}\big[(\delta_k)_{m_k}-\eta\frac{1}{N}\sum_{i=1}^N\nabla f((\mathbf{x}_{k-\tau(k)},\xi_i))_{m_k}+\eta\frac{1}{N}\sum_{i=1}^N\nabla f((\mathbf{x}_{k-\tau(k)}',\xi_i))_{m_k}\big]\\&+\mathbb{E}\big[\frac{1}{N}\eta\nabla f((\mathbf{x}_{k-\tau(k)}',\xi_i))_{m_k}-\frac{1}{N}\eta\nabla f((\mathbf{x}_{k-\tau(k)}',\xi_i'))_{m_k}\big]\\\overset{(a)}{\le}&\mathbb{E}\big[(\delta_k)_{m_k}-\eta A(\delta_{k-\tau(k)})_{m_k}\big]+\epsilon'',\end{aligned} \tag{76}$$

where (a) follows from f is a quadratic convex function. We note that $\|\epsilon''\|\le\frac{2\eta M}{N\sqrt{d}}$ since we have the bounded gradient Assumption 4 and $m_k\in 1,2,...,d$ is the uniformly selected updated coordinate in x in iteration k. Then, we have

$$\begin{bmatrix}(\delta_{k+1})_{m_k}\\(\delta_k)_{m_k}\\\vdots\\\delta_{k-\tau(k)+1}\\\delta_{k-\tau(k)}\end{bmatrix}=\begin{bmatrix}1&0&\cdots&-\eta A&0\\1&0&\cdots&0&0\\\vdots&\vdots&\ddots&\vdots&\vdots\\\vdots&\vdots&&\ddots&\vdots\\0&0&\cdots&1&0\end{bmatrix}\begin{bmatrix}(\delta_k)_{m_k}\\(\delta_{k-1})_{m_k}\\\vdots\\(\delta_{k-\tau(k)})_{m_k}\\(\delta_{k-\tau(k)-1})_{m_k}\end{bmatrix}+\begin{bmatrix}\epsilon''\\0\\\vdots\\0\\0\end{bmatrix}. \tag{77}$$

Let Matrix

$$\mathbf{Q} \triangleq \begin{bmatrix} 1 & 0 & \cdots & -\eta A & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}. \tag{78}$$

Consider the characteristic polynomial

$$\begin{bmatrix} 1 & 0 & \cdots & -\eta A & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \vdots \\ \mathbf{v}_{\tau(k)+2} \end{bmatrix} = \lambda_{\mathbf{Q}} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \vdots \\ \mathbf{v}_{\tau(k)+2} \end{bmatrix}, \tag{79}$$

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

$$\big(-\lambda_{\mathbf{Q}}^{\tau^k+2} + \lambda_{\mathbf{Q}}^{\tau^k+1} + \lambda_{\mathbf{Q}}[-\eta A]\big)\mathbf{v}_{\tau(k)+2} = 0. \tag{80}$$

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

$$\mathbf{U}\big(-\lambda_{\mathbf{Q}}^{\tau^k+2} + \lambda_{\mathbf{Q}}^{\tau^k+1} + \lambda_{\mathbf{Q}}[-\eta A]\big)\mathbf{U}^\top = 0. \tag{81}$$

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

$$\lambda_{\mathbf{Q}}^{\tau_k+1} \cdot \big[-\lambda_{\mathbf{Q}} + 1\big] = \lambda_{\mathbf{Q}}[\eta\lambda_i]. \tag{82}$$

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, $max(|\lambda_{\mathbf{Q}}|) = 1$. Thus, for $mod(k, q) \neq 0$ we have

$$\mathbb{E}\|(\delta_{k+1})_{m_k}\| \leq \mathbb{E}\|(\delta_k)_{m_k}\| + \frac{2\eta M}{N\sqrt{d}}. \tag{83}$$

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 (83), 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:

$$\mathbb{E} \parallel \delta_{k+1} \parallel \leq \frac{2\eta MK}{N\sqrt{d}}. \tag{84}$$

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

$$\epsilon' \leq M \cdot \mathbb{E} \parallel \delta_{t+1} \parallel \leq \frac{2\eta M^2 K}{N\sqrt{d}}. \tag{85}$$

This completes the proof. □

D.2 Proofs of Theorem 4

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

$$(\mathbf{x}_{k+1})_{m_k} = (\mathbf{x}_k)_{m_k} - \eta\frac{1}{|S|}\sum_{i\in S}(\nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i) - \nabla f(\mathbf{x}_{k-\tau(k)-1-\tau(k-\tau(k)-1)}, \xi_i) + \mathbf{v}_{k-1-\tau(k)})_{m_k}. \tag{86}$$

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

$$\mathbb{E}(\delta_{k+1})_{m_k} = \mathbb{E}(\mathbf{x}_{k+1})_{m_k} - \mathbb{E}(\mathbf{x}'_{k+1})_{m_k} = \mathbb{E}(\delta_k)_{m_k} - \eta[\mathbb{E}(\mathbf{v}_k)_{m_k} - \mathbb{E}(\mathbf{v}'_k)_{m_k}]. \tag{87}$$

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

$$\mathbb{E}(\delta_{k+1})_{m_k} = \mathbb{E}(\delta_k)_{m_k} - \eta[\mathbb{E}(\nabla f(\mathbf{x}_{k-\tau(k)}))_{m_k} - \mathbb{E}(\nabla f(\mathbf{x}'_{k-\tau(k)}))_{m_k}]. \tag{88}$$

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

$$\begin{aligned}
\mathbb{E}\|(\delta_{k+1})_{m_k}\| \leq & \mathbb{E}\|(\delta_k)_{m_k} - \eta(\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i))_{m_k} + \eta(\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}'_{k-\tau(k)}, \xi_i))_{m_k}\| \\
& + \frac{1}{N}\eta\|\nabla f(\mathbf{x}'_{k-\tau(k)}, \xi_i))_{m_k} - \nabla f(\mathbf{x}'_{k-\tau(k)}, \xi'_i))_{m_k}\| \\
\overset{(a)}{\leq} & \mathbb{E}\|(\delta_k)_{m_k} - \eta(\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i))_{m_k} + \eta(\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}'_{k-\tau(k)}, \xi_i))_{m_k}\| + \frac{2\eta M}{N\sqrt{d}} \\
\overset{(b)}{\leq} & \mathbb{E}\|(\delta_k)_{m_k}\| + \eta\mathbb{E}\|(\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}_{k-\tau(k)}, \xi_i))_{m_k} - (\frac{1}{N}\sum_{i=1}^{N}\nabla f(\mathbf{x}'_{k-\tau(k)}, \xi_i))_{m_k}\| + \frac{2\eta M}{N\sqrt{d}} \\
\overset{(c)}{\leq} & \mathbb{E}\|(\delta_k)_{m_k}\| + \frac{2\eta M}{\sqrt{d}} + \frac{2\eta M}{N\sqrt{d}},
\end{aligned} \tag{89}$$

where (a) and (b) follows from the triangle inequality, (c) follows from the bounded gradient Assumption 4. 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 4 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

$$\mathbb{E}\|(\delta_{k+1})\| \leq \mathbb{E}\|(\delta_k)\| + \frac{2\eta M}{\sqrt{d}} + \frac{2\eta M}{N\sqrt{d}}. \tag{90}$$

Thus, total number K of iterations satisfies

$$\| \delta_{k+1} \| \leq \frac{2\eta MK}{\sqrt{d}} + \frac{2\eta MK}{N\sqrt{d}}. \tag{91}$$

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

$$\epsilon' \leq M \cdot \| \delta_{t+1} \| \leq \frac{2\eta M^2 K}{\sqrt{d}} + \frac{2\eta M^2 K}{N\sqrt{d}}. \tag{92}$$

This completes the proof. □