Distributed Accelerated Proximal Optimization Algorithms with Variance Reduction [*]

March 2018

Abstract

The proximal stochastic gradient descent (ProxSGD) has been widely used to solve composite convex optimization problems. However, the random sampling in ProxSGD introduces noisy gradient updates with high variance, which causes to use a vanishing step size and so slows down the convergence as the iterates approach the optimum. In addition, although ProxSGD with variance-reduction enjoys great success in applications at small and moderate scales, but distributed versions of these algorithms are crucially demanded for larger-scale training datasets. In this paper, we propose a synchronous method, Sync-AcPSVRG, and an asynchronous method, Async-AcPSVRG, which integrate variance-reduction and momentum acceleration techniques in a distributed manner to speed up ProxSGD updates and can achieve significant speedup with the number of workers without hurting the convergence rate. Both Sync-AcPSVRG and Async-AcPSVRG enjoy lower iteration complexity than existing accelerated stochastic variance reduction methods, which need more updates per iteration. Furthermore, we allow the number of updates to increase with the epochs to secure sparse communications between workers and master. We compare distributed AcPSVRG with existing parallel stochastic proximal optimization methods on a few datasets. The empirical results verify our theoretical results and indicate that our proposed distributed accelerated methods with variance reduction are more efficient and effective than other ProxSGD variants.

1 Introduction

In this paper, we consider the following composite optimization problem

$$\min_{x\in\mathbb{R}^d} F(x) = h(x) + f(x) = h(x) + \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{1}$$

where $f_i(x)$, $i = 1,2,\ldots,n$, are smooth convex loss functions, and $h(x)$ is a non-smooth regularization term. Proximal stochastic gradient descent (ProxSGD) [**?**] is a general

[*]

approach for solving minimization problem in (1), which employs the fact that the objective function decomposes into a sum over many terms.

For problems where each f_i in (1) corresponds to a single data observation, ProxSGD selects a single data index i_k on each iteration k, and then performs update by solving the proximal optimization subproblem

$$x^{k+1} = \mathrm{Prox}_{\eta,h}\{x^k - \eta \nabla f_{i_k}(x_k)\},$$

where the proximal mapping is defined as

$$\mathrm{Prox}_{\eta,h}(y) = \ \mathrm{argmin}_{x\in\mathbb{R}^d}\{\frac{1}{2\eta}\,\|x-y\|^2 + h(x)\}.$$

In the above notation $\|\cdot\|$ is the L_2-norm and η is the step size. Typically, i_k is chosen uniformly at random from $\{1,2,\ldots,n\}$ on each iteration k, thus making the gradient approximation unbiased. However, because random sampling in ProxSGD introduces variance, vanishing step sizes are needed to guarantee the algorithm's convergence, and the convergence rate is only sublinear [**? ?**]. In [**?**], a variance reduction technique for proximal algorithms was introduced and it is proved that it can achieve linear convergence. The convergence rate of ProxSGD with variance reduction can be further improved with the so-called momentum acceleration technique [**?**].

Another major drawback of ProxSGD is the sequential nature of algorithm. For massive datasets or training datasets distributed over a cluster of multiple nodes, parallel or distributed algorithms are sorely needed, making more practical impacts on parallel SGD variants to solve large-scale or distributed problems. There is quite a bit of recent works in the area of distributed optimization that have been successfully applied to accelerate many optimization algorithms including SGD [**? ? ?**] and ProxSGD [**? ?**].

In this paper, we propose distributed algorithms to enhance the scalability of proximal algorithms using variance reduction and momentum acceleration techniques, yielding ProxSGD methods that scale near linearly and can provide better convergence rate.

2 Backgrounds

Since SGD estimates the gradient from only one or a few samples, the variance of the stochastic gradient estimator may be large [**? ?**], which leads to slowdown and poor performance. Recently there has been a lot of interest in methods which reduce the variance incurred due to stochastic gradients. Variance reduction (VR) methods [**? ?**] reduce the variance in the stochastic gradient estimates, and are able to alleviate the effects of vanishing step sizes that usually hit SGD, and yield methods that improve convergence rates both theoretically and empirically. Inspired by the success of these methods in reducing the gradient noise in stochastic gradient based optimization, recently many variants of variance reduction methods have been proposed [**? ? ? ? ?**]. Variance reduction methods begin with an initial estimate $\widetilde{x}$, and then generate a sequence of iterates x_k from

$$x_k = x_{k-1} - \eta\left[\nabla f_{i_k}(x_{k-1}) - \nabla f_{i_k}(\widetilde{x}) + \widetilde{\nabla} f(\widetilde{x})\right],$$

where η is the step size, $\widetilde{\nabla} f(\widetilde{x})$ is an approximation of the true gradient, and $\widetilde{x}$ is chosen to be a recent iterate from algorithm history. In particular, an error correction term is subtracted from regular update rules in stochastic optimization to reduce the variance of gradients in order to deal with the problems of instability and slower convergence hit SGD.

Recently several acceleration techniques based on momentum compensations were introduced to further speed up the VR methods mentioned above [**?** **?** **?** **?**]. The momentum acceleration technique, which is basically proposed by Nesterov [**?**] for gradient methods, introduces one or multiple auxiliary variables to record the momentum, and updates the parameters according to the gradient at the auxiliary variable. To be specific, with momentum acceleration, the update rule becomes

$$\begin{aligned} x_{k+1} &= z_k - \eta \nabla f_{i_k}(z_k), \\ z_{k+1} &= x_{k+1} + \beta (x_{k+1} - x_k), \end{aligned} \tag{2}$$

where β is the momentum weight. It can be proven that the convergence rate of SGD is improved by using the Nesterov acceleration technique [**?**]. After that, many accelerated algorithms have been designed to achieve faster convergence rates for stochastic optimization methods [**?** **?** **?** **?**].

Although these acceleration techniques have great value in general, for large-scale problems, however, with the availability of very big training data, sequential SGD is very inefficient. Therefore, integrating the VR algorithms and acceleration techniques to distributed settings remain indispensable. In [**?** **?**] SVRG is extended to the parallel asynchronous setting. In particular, in [**?**] a parallel algorithm mitigated by variance reduction, coordinate sampling, and Nesterov's method is introduced. But the scalability of algorithm is only proven for sparse datasets and under appropriate parameter setting. An asynchronous implementation of VR method with acceleration is presented in [**?**], however it requires learning rate to decrease with the square of the upper bound for delays, i.e. τ^2, and thereby an asynchronous speedup is not shown.

In this paper, we introduce distributed synchronous and asynchronous SGD-based algorithms with variance reduction integrated with acceleration techniques, which have not been well studied in the literature to the best of our knowledge. We prove that the proposed distributed algorithms have desirable convergence property, show considerable speedup. The parallel algorithms are highly scalable, uses a constant learning rate for convex functions, and converges linearly to the optimal solution in the strongly convex case.

3 Our Approaches

In this work, we attempt to introduce momentum to accelerate distributed variance reduction SGD algorithms for convex problems. We demonstrate that one step of momentum with only one weight provides a faster convergence rate in expectation, while we assume an upper bound for delay τ between delayed gradient and the latest one.

By applying this technique in distributed stochastic algorithms, we allow many local nodes to run simultaneously, while communicating with the server node through the exchange of updates. This new algorithm allows many processes to work towards a central solution which are faster by order than existing variance-reduced ProxSGD algorithms.

This work has three main contributions:

1. We propose distributed algorithms for convex problems adopted with variance reduction methods and momentum acceleration techniques with only one momentum weight. Our asynchronous algorithms have a low storage requirement, and local nodes only need to store the averaged sample gradient which is more suitable for optimization with a large number of variables.

2. We theoretically study the convergence for general proximal algorithm and obtain improved convergence rate for convex and strongly convex functions. We will show our proposed distributed algorithms with acceleration and variance reduction, can speed up on sparse or dense data. It is scalable and uses a constant learning rate for convex functions. To the best of our knowledge it is the first analyses to achieve speedup with no assumptions on sparsity and the number of workers. The algorithms uses a constant learning rate for convex functions and the learning rate has to decrease only with τ.

3. Finally, we present several empirical results over several distributed VR algorithms to demonstrate that the new accelerated algorithm can lead to better performance improvements and promising speedup than competing options, which agrees with our theory.

4 Algorithm Overview

We begin by proposing our new accelerated proximal variance reduction scheme with momentum acceleration (AcPSVRG), in the single-worker case. As we will see later, the proposed method has a natural generalization to the distributed setting that has low communication requirement. Similar to the epoch gradient descent algorithm [**?**], we increase the number of iterations by a constant γ for every epoch.

Our proposed VR scheme (AcPSVRG) is divided into S epochs, with m_s updates taking place in each epoch. Throughout the paper, we will use the superscript for the index of each epoch, and the subscript for the index of iterations within each epoch.

Let the iterates generated in the s-th epoch be written as $\{x^s_k\}_{k=0}^{m_s-1}$. The update rule for AcPSVRG can be formulated as follows:

In each iteration the following type of proximal subproblem is solved,

$$z^s_{k+1} = \text{Prox}_{\frac{\eta}{\beta_s}} h\{z^s_k - \frac{\eta}{\beta_s}\widetilde{\nabla}^s_k\}, \tag{3}$$

where

$$\widetilde{\nabla}^s_k = \nabla f_{i^s_k}(x^s_k) - \nabla f_{i^s_k}(\widetilde{x}^{s-1}) + \nabla f(\widetilde{x}^{s-1}). \tag{4}$$

We let snapshot $\widetilde{x}^s$ be a weighted average of x^s_k in the most recent epoch s, which is updated at the end of each epoch, i.e., after every m_s parameter updates. We compensate the momentum term and introduce a new extrapolation rule to update x_k^s,

$$x_{k+1}^s = \widetilde{x}^s + \beta_s (z_{k+1}^s - \widetilde{x}^s),$$

where $\beta_s \in [0,1]$ is the momentum weight. Indeed, z_{k+1}^s is a momentum which adds a weighted sum of gradient history into x_k^s. Comparably, this update has only one momentum weight β_s versus two weights τ_1 and τ_2 in [**?**] through introducing two momentum vectors. We increase the number of iterations by a constant γ for every epoch, i.e., $m_{s+1} = \gamma m_s$. The choice of growing the iterations per epoch can reduce the number of full gradient calculations and the frequency of communication between the server and the local nodes in the distributed setting, while it can speed up the convergence [**?**].

Note that if i_k^s in (4) is chosen uniformly at random from the set $\{1, 2, \ldots, n\}$ on each iteration k, then, conditioning on all history,

$$\mathbb{E}[\nabla f_{i_k^s}(\widetilde{x})] = \nabla f(\widetilde{x}).$$

Thus, the error correction term has expected value 0, and $\mathbb{E}[\widetilde{\nabla}_k^s] = \nabla f(x_k^s)$, i.e., the approximate gradient $\widetilde{\nabla}_k^s$ is unbiased.

5 Distributed Algorithms

We now consider the distributed setting, on a master-slave framework with a master machine and p local clients, each of which contains a portion of the data set. In this setting, the whole data Ω is decomposed into disjoint subsets $\{\Omega_k\}$, where k denotes a particular local worker, and $\cup_{l=1}^{p} \Omega_l = \Omega$ and if $i \neq j$, $\Omega_i \cap \Omega_j = \emptyset$. Different clients can not communicate with each other and the clients can only communicate with the central server. Our model of computation is similar to the ones used in Parameter Server [**?**] and Mllib [**?**]. Our goal is to derive stochastic algorithms that scale linearly to high p.

5.1 Synchronous Accelerated SGD

AcPSVRG naturally extends to the distributed synchronous setting, and is presented in Algorithm 1. To distinguish the algorithm from the single worker case, we call it Sync-AcPSVRG. Note, the number of updates per epoch m_s and $\widetilde{x}^s$ are initialized with m_1 and $\widetilde{x}^0$. At the beginning of each epoch we initialize $x_0^s = z_0^s = \widetilde{x}^{s-1}$, where $\widetilde{x}^{s-1}$ is the average of the past m_{s-1} stochastic iterations. On each epoch, the local nodes first retrieve a copy of the central iterate $\widetilde{x}^{s-1}$, compute the sum of the gradients on its local data, i.e., $\sum_{i \in \Omega_k} \nabla f_i(\widetilde{x}^{s-1})$ and send it to the master node. The accelerated VR method is then locally performed on each node by only using the local data, and each worker sends the most recent parameter denoted as x_k for k-th worker to the master. Then parameter $\widetilde{x}^s$ is updated by the master after all the locally updated parameters have been gathered. As we put now constraint on how training data are partitioned on different workers, by sharing full gradient $\nabla f(\widetilde{x})$ across nodes, we ensure that the local gradient updates utilize global gradient information from remote nodes. We ensure also local updates are not far away from global updates through including momentum.

This speeds up the convergence of stochastic optimization and controls the difference between the local update and global update, even if each local node runs for one whole epoch before communicating back with the central node.

```
**Algorithm 1** Sync-AcPSVRG
Input: The number of epochs $S$ and the step size $\eta$
Initialize: $\widetilde{x}^0$, $m_1$, $\theta_1$, and $\rho > 1$
  for Worker $l$ do
    for  $s = 1$ to $S$  do
      Receive $\widetilde{x} = \widetilde{x}^{s-1}$ from master node.
      Send $\nabla_l f(\widetilde{x}) = \sum_{i \in \Omega_l} \nabla f_i(\widetilde{x})$ to master node.
      Receive $\nabla f(\widetilde{x})$ from master node.
      $x^s_{l,0} = z^s_{l,0} = \widetilde{x}$
      for $k = 0$ to $m_s - 1$ do
        Pick $i^s_{l,k}$ uniformly at random from $\Omega_l$.
        $\widetilde{\nabla}^s_{l,k} = \nabla f_{i^s_{l,k}}(x^s_{l,k}) - \nabla f_{i^s_{l,k}}(\widetilde{x}) + \nabla f(\widetilde{x})$
        $\delta^s_{l,k} = \mathrm{argmin}_\delta \, h(z^s_{l,k} + \delta) + \langle \widetilde{\nabla}^s_{l,k}, \delta \rangle + \frac{\beta_s}{2\eta} \|\delta\|^2$
        $z^s_{l,k+1} = z^s_{l,k} + \delta^s_{l,k}$
        $x^s_{l,k+1} = \widetilde{x} + \beta_s (z^s_{l,k+1} - \widetilde{x})$
      end for
      $\widetilde{x}^s_l = \frac{1}{m_s} \sum_{k=0}^{m_s - 1} x^s_{l,k}$, $m_{s+1} = \gamma m_s$
      Send $\widetilde{x}^s_l$ to master node.
    end for
  end for
  Master Node:
        Average $\widetilde{x}_l$ received from workers.
        Broadcast averaged $\widetilde{x}$ to local workers.
        Average $\nabla f(\widetilde{x}_l)$ received from workers.
        Broadcast averaged $\nabla f(\widetilde{x})$ to local workers.
Output: $\widetilde{x}^S$
```

In Sync-AcPSVRG, as we mentioned earlier, we assume that each worker has access to a subset and not the entire data set and performs local updates for one epoch, or iterations, before communicating with the server. This is a rather low communication frequency compared to a mini-batch parameter server model such as parallel implementation of SVRG [**?**] or mS2GD [**?**] in which stochastic optimization is performed on the master node based on the whole dataset, and updates and gradients are frequently transferred between workers and master. This makes a significant difference in runtimes when the number of local nodes is large. We also increase the number of inner updates per epoch by a constant γ which reduces the communication rate between server and worker because there is no communication during the inner iterations of each epoch and yields fastest convergence. We also let local workers make updates according to the proximal solver.

5.2 Asynchronous Accelerated SGD

The synchronous algorithm could be extended to the asynchronous one called Async-AcPSVRG as shown in Algorithm 2. We adopt asynchronous update in each inner loop and there is synchronization operation after each epoch. In Async-AcPSVRG, the master node keeps a copy of the averaged x. We make workers do proximal mapping step, and server is responsible for element-wise addition operations, average x, aggregate full gradient and then broadcast it to workers at the end of each epoch.

Algorithm 2 Async-AcPSVRG

```
Input: The number of epochs $S$ and the step size $\eta$.
Initialize: $\widetilde{x}^0$, $m_1$, $\theta_1$, and $\rho > 1$
  for $s = 1$ to $S$ do
    $t = 0$;
    Worker $l$
    Wait until it receives $\widetilde{x} = \widetilde{x}^{s-1}$ from master node.
    Send $\nabla_l f(\widetilde{x}) = \sum_{i \in \Omega_l} \nabla f_i(\widetilde{x})$ to master node.
    Wait until it receives $\nabla f(\widetilde{x})$ from master node.
    $x^s_{l,0} = z^s_{l,0} = \widetilde{x}$
    for $k = 0$ to $m_s - 1$ do
      Receive $x^s_{l,k-\tau_k} := x^s_t$ from master node.
      Pick $i^s_{l,k}$ uniformly at random from $\Omega_l$.
      $\widetilde{\nabla}^s_{l,k} = \nabla f_{i^s_{l,k}}(x^s_{l,k-\tau_k}) - \nabla f_{i^s_{l,k}}(\widetilde{x}) + \nabla f(\widetilde{x})$
      $\delta^s_{l,k} = \text{argmin}_\delta \, h(z^s_{l,k} + \delta) + \langle \widetilde{\nabla}^s_{l,k}, \delta \rangle + \frac{\beta_s}{2\eta} \|\delta\|^2$
      $z^s_{l,k+1} = z^s_{l,k} + \delta^s_{l,k}$
      Send $z^s_{l,k+1}$ to master node.
      Master Node:
      Receive $z^s_{l,k+1}$ from worker $l$.
      Update $x^s_{t+1} = \widetilde{x} + \beta_s(z^s_{l,k+1} - \widetilde{x})$, $t = t + 1$.
    end for
    Master Node:
    Calculate average $\widetilde{x}^s = \frac{1}{t} \sum_{i=1}^{t} x^s_i$ and broadcast averaged $\widetilde{x}^s$.
    Receive $\nabla_l f(\widetilde{x}^s)$ from workers and calculate average $\nabla f(\widetilde{x}^s)$.
    Broadcast averaged $\nabla f(\widetilde{x}^s)$ to local workers.
    $m_{s+1} = \gamma m_s$
  end for
Output: $\widetilde{x}^S$
```

The key idea for Async-AcPSVRG is that, once a local node solves the proximal update, it sends the update to the master. The master thread will use the update received by the local node to update x. Each processor repeatedly runs these procedures concurrently, without any synchronization. We use $k - \tau_k$ to denote the state of x at the reading time by the local worker. For asynchronous algorithms, the partial gradient calculated by the local worker will be delayed, and at iteration k the worker l could only obtain $\nabla f_{i_{l,k}^s}(x_{k-\tau_k}^s)$ instead of $\nabla f_{i_{l,k}^s}(x_k^s)$. We assume that the delay between the time of evaluation and updating is bounded by a non-negative integer τ, i.e., $\tau_k \le \tau$. The parameter τ captures the degree of delay. When there are more threads, the delay accumulates and results in larger τ. Furthermore, we also assume that the system is synchronized after every epoch.

For the purpose of our analysis, we assume a consistent read model, i.e., when x is read or updated in the central node, it will be locked. However, the proposed asynchronous algorithms can be easily implemented in a lock-free setting, leading to further speedups. We leave the analysis of inconsistent read (wild) model as future work.

6 Convergence Analysis

In this section, we provide convergence analysis for distributed AcPSVRG. For further analysis, throughout this paper, we make the following assumptions, which are commonly used in previous works [**? ?**].

Assumption 1 (Lipschitz Gradient). The components $f_i(x), i = 1, 2, \dots, n$ are differentiable and have Lipschitz continuous partial gradients, i.e., for $x, y \in \mathbb{R}^d$, we have,

$$\|\nabla f_i(x) - \nabla f_i(y)\| \le L \|x - y\| . \tag{5}$$

Assumption 2 (Convexity). The components $f_i(x), i = 1, 2, \dots, n$ and function $h(x)$ are convex. The objective function $F(x)$ could be strongly convex with parameter μ, i.e., $\forall x, y \in \mathbb{R}^d$

$$F(y) \ge F(x) + \langle \xi, y - x \rangle + \frac{\mu}{2} \|x - y\|^2 , \qquad \forall \xi \in \partial F(x), \tag{6}$$

and for non-strongly convex functions $\mu = 0$.

Assumption 3 (Bounded Delay). The delays $\tau_1, \tau_2, \dots$ are independent random variables, and $\tau_k \le \tau$ for all k. As we mentioned earlier, we use $k - \tau_k$ to denote the read state at iteration k in the asynchronous algorithm.

Let $p(w)$ be a convex function over a convex set X. Let $\hat{w} = \operatorname{argmin}_{w \in X} \{p(w) + \alpha \|w - \bar{y}\|^2\}$ for some $\bar{y} \in X$ and $\alpha \ge 0$. Due to the fact that the sum of a convex and a strongly convex function is also strongly convex, we have

$$p(w) + \alpha \|w - \bar{y}\|^2 \ge p(\hat{w}) + \alpha \|\hat{w} - \bar{y}\|^2 + \alpha \|w - \hat{w}\|^2 . \tag{7}$$

Lemma 6.1. *Under Assumptions 1-3, if x^* is the optimal solution of Problem (1), for the Algorithm 2 and $\eta < \min\{\frac{1}{4L\tau}, \frac{1}{3L}\}$, we have*

$$\begin{aligned}\mathbb{E}[F(\widetilde{x}^s) - F(x^*)] \le& (1 - \beta_s)[F(\widetilde{x}^{s-1}) - F(x^*)] \\ &+ \frac{\beta_s^2}{2\eta m_s} \mathbb{E}[\|z_0^s - x^*\|^2 - \|z_{m_s}^s - x^*\|^2].\end{aligned} \tag{8}$$

with $\lambda_s = \frac{(1-\beta_s+\alpha_s)}{1-\alpha_s}$, $\alpha_s = \frac{4(2+\theta_\eta^{-1})L^2\tau^2\eta^2}{1-2L^2\tau^2\eta^2}$, $\theta_\eta = \frac{1-\eta L}{2\eta L}$ and β_s is chosen such that $\beta_s <$ $1 - \theta_\eta^{-1}$.

Proof. See Appendix 9. □

We have the following convergence result.

Theorem 6.2. *Suppose the assumptions of Lemma 9.4 and further, β_s and η are chosen such that $\beta_s \le \frac{1}{s+2}$, $\lambda_s/\beta_s^2 \le 1/\beta_{s-1}^2$ and $\eta < \min\{\frac{1}{4L\tau}, \frac{1}{3L}\}$. Then, we have*

$$\begin{aligned}\mathbb{E}[F(\widetilde{x}^S) - F(x^*)] \le & \frac{1}{\beta_0^2(S+2)^2}[F(\widetilde{x}^0) - F(x^*)] \\ & + \frac{1}{\eta m_1 (S+2)^2}\mathbb{E}[\|z_0^1 - x_*\|^2].\end{aligned} \tag{9}$$

If function $F(x)$ is strongly convex with parameter μ, and η, β_s and m_s are chosen such that $\alpha_s \le \beta_s/4$ and

$$\lambda_s' := \lambda_s + \frac{2\beta_s^2}{\mu \eta m_s} < 1,$$

we have,

$$\mathbb{E}\left[F(\widetilde{x}^S) - F(x^*)\right] \le \lambda_{max}^S \left[F(\widetilde{x}^0) - F(x^*)\right], \tag{10}$$

where $\lambda_{max} = \max_s \lambda_s'$.

Proof. By selecting $\eta < \min\{\frac{1}{4L\tau}, \frac{1}{3L}\}$, we have $\alpha_s \le \frac{1}{2}$. From Lemma 9.4 we have,

$$\begin{aligned}\mathbb{E}[F(\widetilde{x}^s) - F(x^*)] \le & (1-\beta_s)[F(\widetilde{x}^{s-1}) - F(x^*)] \\ & + \frac{\beta_s^2}{2\eta m_s}\mathbb{E}[\|z_0^s - x^*\|^2 - \|z_{m_s}^s - x^*\|^2].\end{aligned} \tag{11}$$

Dividing both sides of the above inequality by β_s^2, and using the fact $(1-\beta_s)/\beta_s^2 \le 1/\beta_{s-1}^2$ we have

$$\begin{aligned}\frac{\mathbb{E}[F(\widetilde{x}^s) - F(x^*)]}{\beta_s^2} \le & \frac{(1-\beta_s)}{\beta_s^2}[F(\widetilde{x}^{s-1}) - F(x^*)] \\ & + \frac{1}{2\eta m_s}\mathbb{E}[\|z_0^s - x^*\|^2 - \|z_{m_s}^s - x^*\|^2] \\ \overset{a}{\le} & \frac{1}{\beta_{s-1}^2}[F(\widetilde{x}^{s-1}) - F(x^*)] \\ & + \frac{1}{2\eta m_s}\mathbb{E}[\|z_0^s - x^*\|^2 - \|z_{m_s}^s - x^*\|^2],\end{aligned}$$

where in inequality $\overset{a}{\le}$ we used $\frac{\lambda_s}{\beta_s^2} \le \frac{1}{\beta_{s-1}^2}$. By using $z_0^{s+1} = z_{m_s}^s$, and adding the above inequality from $s = 1$ to S, we have

$$\frac{\mathbb{E}[F(\widetilde{x}^S) - F(x^*)]}{\beta_S^2} \le \frac{1}{\beta_0^2}[F(\widetilde{x}^0) - F(x^*)] + \frac{1}{2\eta m_1}\mathbb{E}\left[\|z_0^1 - x_*\|^2\right],$$

where we used $m_1 \leq m_s$ and $\alpha_s \leq \frac{1}{2}$. Then we have

$$\begin{aligned}
&\mathbb{E}[F(\widetilde{x}^S) - F(x^*)] \\
\leq &\frac{\beta_S^2}{\beta_0^2}[F(\widetilde{x}^0) - F(x^*)] + \frac{\beta_S^2}{2\eta m_1}\mathbb{E}[\left\|z_0^1 - x_*\right\|^2 \\
\leq &\frac{1}{\beta_0^2(S+2)^2}[F(\widetilde{x}^0) - F(x^*)] + \frac{1}{2\eta m_1(S+2)^2}\mathbb{E}[\left\|z_0^1 - x_*\right\|^2].
\end{aligned}$$

Now suppose $F(x)$ is μ-strongly convex. Since x^* is the optimal solution, by inequality (6) we have

$$\nabla F(x^*) = 0, \qquad F(x) - F(x^*) \geq \frac{\mu}{2}\left\|x - x^*\right\|^2. \tag{12}$$

Using the inequality in (12) and $z_0^s = \widetilde{x}^{s-1}$, we have

$$\begin{aligned}
&\mathbb{E}[F(\widetilde{x}^s) - F(x^*)] \\
&\overset{a}{\leq} (1-\beta_s)[F(\widetilde{x}^{s-1}) - F(x^*)] + \frac{\beta_s^2}{2\eta m_s}\mathbb{E}[\|z_0^s - x^*\|^2 - \left\|z_{m_s}^s - x^*\right\|^2] \\
&\overset{b}{\leq} (1-\beta_s)\mathbb{E}[F(\widetilde{x}^{s-1}) - F(x^*)] + \frac{\beta_s^2}{2\mu\eta m_s}\mathbb{E}[F(\widetilde{x}^{s-1}) - F(x^*)] \\
&= ((1-\beta_s) + \frac{\beta_s^2}{2\mu\eta m_s})\mathbb{E}[F(\widetilde{x}^{s-1}) - F(x^*)] \\
&= \alpha_s\mathbb{E}[F(\widetilde{x}^{s-1}) - F(x^*)]
\end{aligned} \tag{13}$$

where the inequality $\overset{a}{\leq}$ follows from Lemma 9.4, and $\overset{b}{\leq}$ is due to the inequality (12) and the inequality $\alpha_s \leq \frac{1}{2}$. □

As a corollary, we immediately obtain an expected linear rate of convergence for the synchronous algorithm Sync-AcPSVRG for the strongly convex case.

Corollary 6.3. *Suppose function F is strongly convex and β_s, m_s and step size η are chosen such that the following condition is satisfied*

$$\lambda_s' := 1 - \beta_s + \frac{2\beta_s^2}{\eta\mu m_s} < 1,$$

Then, for iterates of algorithm Sync-AcPSVRG, we have

$$\mathbb{E}\left[F(\widetilde{x}^S) - F(x^*)\right] \leq \lambda_{max}^S\left[F(\widetilde{x}^0) - F(x^*)\right], \tag{14}$$

with $\lambda_{max} = \max_s \lambda_s'$. The optimal value of λ is $\lambda_{max} = 1 - \frac{\mu m_s \eta}{8}$ which is obtained by choosing $\beta_s = \frac{\mu m_s \eta}{4}$.

Proof. For the synchronous algorithm, $\tau = 0$, and consequently $\alpha_s = 0$. Then, from Theorem 6.2, we have $\lambda_s = 1 - \beta_s$ and

$$\lambda_s' = 1 - \beta_s + \frac{2\beta_s^2}{\mu\eta m_s}.$$

The minimal value of λ_{max} is obtained by minimizing λ_s' with respect to β_s. □

Remark 6.4. Compared to Sync-AcPSVRG, due to the delay the convergence rate for Async-AcPSVRG is slower and thereby we prove an asynchronous speedup. In order to obtain speedup for Async-AcPSVRG, the order of inner loop m_s cannot be τ times larger than Async-AcPSVRG. From Corollary 6.3 for Sync-AcPSVRG, the inner loop size in the order of $\mathscr{O}(L/\mu)$ makes $\lambda_s' < 1$. In Theorem 6.2, the condition $\lambda_s/\beta_s^2 \leq 1/\beta_{s-1}^2$ can be satisfied by choosing $\beta_{s-1}/\sqrt{2} \leq \beta_s < \beta_{s-1}$. Further, for inequality $\alpha_s \leq \beta_s/4$, it is sufficient to have

$$\begin{aligned}
\alpha_s &\leq \beta_s/4 \\
\alpha_s &= \frac{4(2+\theta_\eta^{-1})L^2\tau^2\eta^2}{1-2L^2\tau^2\eta^2} \\
\eta &\leq \frac{\sqrt{\beta_s}}{4L\tau}. \qquad (15)
\end{aligned}$$

where we use $\theta_\eta^{-1} \leq 1$ which is from $\eta < \frac{1}{3L}$. Hence from Theorem 6.2, the stepsize η_s satisfies $\eta_s < \min\{\frac{\sqrt{\beta_s}}{4L\tau}, \frac{1}{4L\tau}, \frac{1}{3L}\}$. By having $\lambda_s' < 1$, we obtain

$$\begin{aligned}
\lambda_s' := \lambda_s + \frac{2\beta_s^2}{\mu\eta m_s} &< 1, \\
(1-\beta_s+\alpha_s) + \frac{2(1-\alpha_s)\beta_s^2}{\mu\eta m_s} &< 1-\alpha_s \\
\frac{2(1-\alpha_s)\beta_s^2}{\mu\eta m_s} &\leq \beta_s - 2\alpha_s = \frac{\beta_s}{2} \\
\frac{4(1-\alpha_s)\beta_s}{\mu\eta} &\leq m_s \\
\frac{16(1-\alpha_s)L\tau\beta_s}{\mu\sqrt{\beta_s}} &\leq m_s \\
\frac{16(1-\alpha_s)L\tau\sqrt{\beta_s}}{\mu} &\leq m_s \qquad (16)
\end{aligned}$$

$$\begin{aligned}
(1-\beta_s+\alpha_s) + \frac{2(1-\alpha_s)\beta_s^2}{\mu\eta m_s} &< (1-\frac{\beta_s}{2})(1-\alpha_s) \\
(1-\beta_s+\alpha_s) + \frac{2(1-\alpha_s)\beta_s^2}{\mu\eta m_s} &< (1-\frac{\beta_s}{2})(1-\frac{\beta_s}{4}) \\
\frac{2(1-\alpha_s)\beta_s^2}{\mu\eta m_s} &\leq \frac{\beta_s}{4} \\
\frac{8(1-\alpha_s)\beta_s}{\mu\eta} &\leq m_s \\
\frac{32(1-\alpha_s)L\tau\beta_s}{\mu\sqrt{\beta_s}} &\leq m_s \\
\frac{32(1-\alpha_s)L\tau\sqrt{\beta_s}}{\mu} &\leq m_s \qquad (17)
\end{aligned}$$

Hence, by setting $\sqrt{\beta_s} \leq \frac{1}{\tau}$ the inner loop size m_s should be $\mathscr{O}(L/\mu)$ to make $\lambda_s' < 1$ which is the same as Sync-AcPSVRG. Thus one we can achieve linear speedup for Async-AcPSVRG using adaptive stepsizes.

Corollary 6.5. *Suppose function F is strongly convex and let $\eta_s = \frac{\sqrt{\beta_s}}{4L\tau}$, $\beta_s < \min\{\frac{1}{\tau^2}, \frac{1}{s+2}\}$, and $m_s = \frac{32L}{\mu}$, then,*

$$\lambda_s' := \lambda_s + \frac{2\beta_s^2}{\mu \eta m_s} < 1 - \frac{\beta_s}{2},$$

and for iterates of algorithm Async-AcPSVRG, we have

$$\mathbb{E}\left[F(\tilde{x}^S) - F(x^*)\right] \leq \lambda_{max}^S \left[F(\tilde{x}^0) - F(x^*)\right], \tag{18}$$

with $\lambda_{max} = \max_s \lambda_s'$.

Remark 6.6. For convex functions, we set $\eta = \frac{1}{5L\tau}$, the number of epochs S is in the same order of $\mathscr{O}(\sqrt{\frac{L\tau}{m}}\frac{1}{\sqrt{\varepsilon}})$. Compared with Sync-AcPSVRG, it is at most $\sqrt{\tau}$ larger but the computing time is τ times smaller. The computation can be distributed to multiple workers to get τ times speedup which is in proportion to the number of local workers P. Hence, Async-AcPSVRG can obtain at least $\sqrt{P}$ times speedup.

7 Experiments

8 Conclusion

In this paper, we have studied the distributed proximal stochastic gradient algorithms by integrating with variance reduction and momentum acceleration techniques. Using momentum acceleration rate, we have proved their convergence rates, discussed their speedups, and empirically verified our theoretical findings. Our distributed proximal algorithms can achieve nearly linear speedup. The proposed algorithms can reduce the communication cost significantly by increasing the length of epoch by a constant after each epoch.

As for future work, we will extend the study in this paper to the non-convex case, and investigate AcPSVRG in shared memory architectures, both theoretically and empirically.

9 Proof of Lemma 9.4

Because proximal step needs to compute the gradient at each iteration, it cannot be applied to solve the problems, where the explicit gradient of function $f(x)$ is not available. For example, in the black-box machine learning model, only function values (e.g., prediction results) are available [**?**]. To avoid computing explicit gradient, we use the zeroth-order gradient estimators [**? ? ?**] to estimate the gradient only by function values.

- Specifically, we use the Gaussian Smoothing Gradient Estimator (GauSGE) [**? ? ? ? ?**] to estimate the gradients as follows:

$$\hat{\nabla} f_i(x) = \frac{f_i(x+\mu u_i) - f_i(x)}{\mu} u_i, \qquad i \in [n], \tag{19}$$

  where μ is a smoothing parameter, and $\{u_i\}_{i=1}^n$ denote *i.i.d.* random directions drawn from a zero-mean isotropic multivariate Gaussian distribution $\mathcal{N}(0, I)$.

- Moreover, to obtain better estimated gradient, we can use the Coordinate Smoothing Gradient Estimator (CooSGE) [**? ? ? ? ? ?**] to estimate the gradients as follows:

$$\hat{\nabla} f_i(x) = \sum_{j=1}^{d} \frac{f_i(x+\mu_j e_j) - f_i(x-\mu_j e_j)}{2\mu_j} e_j, \qquad i \in [n], \tag{20}$$

  where μ_j is a coordinate-wise smoothing parameter, and e_j is a standard basis vector with 1 at its j-th coordinate, and 0 otherwise. Although the CooSGE need more function queries than the GauSGE, it can get better estimated gradient, and even can make the algorithms to obtain a faster convergence rate.

Lemma 9.1. *Assume that the function $f(x)$ is L-Lipschitz. Let $\hat{\nabla} f(x)$ denote the estimated gradient defined by ZO-SGD. Define $f_{\mu_j} = \mathbb{E}_{u\sim U[-\mu_j,\mu_j]} f(x+ue_j)$, where $U[-\mu_j, \mu_j]$ denotes the uniform distribution at the interval $[-\mu_j, \mu_j]$. Then we have*
1) f_{μ_j} is L-smooth and

$$\hat{\nabla} f(x) = \sum_{j=1}^{d} \frac{\partial f_{\mu_j}(x)}{\partial x_j} e_j, \tag{21}$$

where $\frac{\partial f}{\partial x_j}$ denotes the partial derivatives with respect to jth coordinate.
2) For $j \in [d]$,

$$\begin{aligned} \left| f_{\mu_j}(x) - f(x) \right| &\le \frac{L\mu_j^2}{2}, \\ \left| \frac{\partial f_{\mu_j}(x)}{\partial x_j} \right| &\le \frac{L\mu_j^2}{2}. \end{aligned} \tag{22}$$

3) If $\mu = \mu_j$ for $j \in [d]$, then

$$\left\| \hat{\nabla} f(x) - \nabla f(x) \right\|^2 \le \frac{L^2 d^2 \mu^2}{4}. \tag{23}$$

Lemma 9.2. *Assume that the function $f(x)$ is L-smooth. Let $\hat{\nabla} f(x)$ denote the estimated gradient defined by Zo-SGD. Define $f_\mu(x) = \mathbb{E}_{u\sim\mathcal{N}(0,1)}[f(x+\mu u)]$. Then we have*
1) For any $x \in \mathbb{R}^d$, $\nabla f_\mu(x) = \mathbb{E}_u[\hat{\nabla} f(x)]$.

2) For any $x \in \mathbb{R}^d$,

$$\begin{aligned}
\left| f_\mu(x) - f(x) \right| &\leq \frac{Ld\mu^2}{2}, \\
\left| \nabla f_\mu(x) - \nabla f(x) \right| &\leq \frac{L\mu(d+3)^{3/2}}{2}, \\
\mathbb{E}_u \left\| \hat{\nabla} f(x) \right\|^2 &\leq 2(d+4) \left\| \nabla f(x) \right\|^2 + \frac{\mu^2 L^2 (d+6)^3}{2}.
\end{aligned} \tag{24}$$

3) For any $x \in \mathbb{R}^d$,

$$\mathbb{E}_u \left\| \hat{\nabla} f(x) - \nabla f(x) \right\|^2 \leq 2(2d+9) \left\| \nabla f(x) \right\|^2 + \mu^2 L^2 (d+6)^3. \tag{25}$$

Lemma 9.3. *In Algorithm* **??** *using CooSGD, given the estimated gradient $\widetilde{\nabla}_k^s$, then the following inequality holds*

$$\mathbb{E} \left\| \widetilde{\nabla}_k^s - \nabla f(x_k^s) \right\|^2 \leq 2L^2 d \mathbb{E} \left\| x_k^s - \widetilde{x}^{s-1} \right\|^2 + \frac{L^2 d^2 \mu^2}{2}. \tag{26}$$

Lemma 9.4. *Under Assumptions 1-3, if x^* is the optimal solution of Problem (1), for the Algorithm 2 and $\eta < \min\{\frac{1}{4L\tau}, \frac{1}{3L}\}$, we have*

$$\begin{aligned}
\mathbb{E}[F(\widetilde{x}^s) - F(x^*)] \leq &(1-\beta_s)[F(\widetilde{x}^{s-1}) - F(x^*)] \\
&+ \frac{\beta_s^2}{2\eta m_s} \mathbb{E}[\left\| z_0^s - x^* \right\|^2 - \left\| z_{m_s}^s - x^* \right\|^2].
\end{aligned} \tag{27}$$

with $\lambda_s = \frac{(1-\beta_s+\alpha_s)}{1-\alpha_s}$, $\alpha_s = \frac{4(2+\theta_\eta^{-1})L^2\tau^2\eta^2}{1-2L^2\tau^2\eta^2}$, $\theta_\eta = \frac{1-\eta L}{2\eta L}$ and β_s is chosen such that $\beta_s < 1 - \theta_\eta^{-1}$.

Proof. By choosing $\eta < \frac{1}{3L}$ we have,

$$\begin{aligned}
f(x_{k+1}^s) &\leq f(x_k^s) + \left\langle \nabla f(x_k^s), x_{k+1}^s - x_k^s \right\rangle + \frac{L}{2} \left\| x_{k+1}^s - x_k^s \right\|^2 \\
&= f(x_k^s) + \left\langle \nabla f(x_k^s), x_{k+1}^s - x_k^s \right\rangle \\
&\quad + \frac{1}{2\eta} \left\| x_{k+1}^s - x_k^s \right\|^2 - \theta_\eta L \left\| x_{k+1}^s - x_k^s \right\|^2 \\
&\overset{a}{=} f(x_k^s) + \left\langle \widetilde{\nabla}_k^s, x_{k+1}^s - x_k^s \right\rangle \\
&\quad + \frac{1}{2\eta} \left\| x_{k+1}^s - x_k^s \right\|^2 + \left\langle \nabla f(x_k^s) - \widetilde{\nabla}_k^s, x_{k+1}^s - x_k^s \right\rangle \\
&\quad - \theta_\eta L \left\| x_{k+1}^s - x_k^s \right\|,
\end{aligned} \tag{28}$$

where the inequality follows from Lipschitz continuous nature of the gradient of function f. In $\overset{a}{=}$, we add and subtract $\left\langle \widetilde{\nabla}_k^s - \nabla f(x_{k-\tau_k}^s), x_{k+1}^s - x_k^s \right\rangle$. From Cauchy-Schwarz inequality, we also have,

$$\begin{aligned}
&\mathbb{E}\left\langle \nabla f(x_k^s) - \nabla f(x_{k-\tau_k}^s), x_{k+1}^s - x_k^s \right\rangle \\
&\le \frac{1}{2\theta_\eta L}\left\|\nabla f(x_k^s) - \nabla f(x_{k-\tau_k}^s)\right\|^2 + \frac{L\theta_\eta}{2}\left\|x_{k+1}^s - x_k^s\right\|^2 . \qquad (29)
\end{aligned}$$

To bound the last term in equation (28), we obtain,

$$\begin{aligned}
&\left\langle \nabla f(x_k^s) - \widetilde{\nabla}_k^s, x_{k+1}^s - x_k^s \right\rangle \\
&\overset{a}{\le} \mathbb{E}\left[\frac{1}{2\theta_\eta L}\left\|\nabla f(x_k^s) - \widetilde{\nabla}_k^s\right\|^2 + \frac{\theta_\eta L}{2}\left\|x_{k+1}^s - x_k^s\right\|^2\right] \qquad (30)
\end{aligned}$$

where inequality $\overset{a}{\le}$ follows from the Cauchy-Schwarz inequality, and inequality $\overset{b}{\le}$ follows from Lemma **??**.

Substituting the inequalities (29) and (30) in (28), and taking expectation over i_k^s, we obtain,

$$\begin{aligned}
&\mathbb{E}\left[F(x_{k+1}^s) - f(x_k^s)\right] \\
&\le \mathbb{E}\left[h(x_{k+1}^s) + \left\langle \widetilde{\nabla}_k^s, x_{k+1}^s - x_k^s \right\rangle + \left(\frac{1}{2\eta} - \frac{\theta_\eta L}{2}\right)\left\|x_{k+1}^s - x_k^s\right\|^2\right] \\
&\quad + \mathbb{E}\left[\frac{1}{2\theta_\eta L}\left\|\nabla f(x_k^s) - \widetilde{\nabla}_k^s\right\|^2\right] \\
&\overset{a}{\le} \mathbb{E}\left[\left\langle \beta_s \widetilde{\nabla}_k^s, z_{k+1}^s - z_k^s \right\rangle\right] + \left(\frac{\beta_s^2}{2\eta}\right)\left\|z_{k+1}^s - z_k^s\right\|^2 - \frac{\theta_\eta L}{2}\left\|x_{k+1}^s - x_k^s\right\|^2 \\
&\quad + \mathbb{E}\left[\beta_s h(z_{k+1}^s) + (1-\beta_s) h(\widetilde{x}^{s-1})\right] \\
&\quad + \frac{1}{2\theta_\eta L}\mathbb{E}\left[\left\|\nabla f(x_k^s) - \widetilde{\nabla}_k^s\right\|^2\right], \qquad (31)
\end{aligned}$$

where in inequality $\overset{a}{\le}$ we use the update $x_k^s = \beta_s z_k^s + (1-\beta_s)\widetilde{x}^{s-1}$ and convexity of h. Since $p(\delta) = h(z_k^s + \delta) + \langle \widetilde{\nabla}_k^s, \delta \rangle$ is a convex function, using inequality (7) with $w = x^* - z_k^s$, $\bar{y} = 0$, it follows

$$\begin{aligned}
h(z_{k+1}^s) + \left\langle \widetilde{\nabla}_k^s, z_{k+1}^s - z_k^s \right\rangle + \frac{\beta_s}{2\eta}\left\|z_{k+1}^s - z_k^s\right\|^2 \\
\le h(x^*) + \left\langle \widetilde{\nabla}_k^s, x^* - z_k^s \right\rangle \\
+ \frac{\beta_s}{2\eta}\left(\|z_k^s - x^*\|^2 - \|z_{k+1}^s - x^*\|^2\right). \qquad (32)
\end{aligned}$$

By replacing the above inequality in (31) we have,

$$
\begin{aligned}
&\mathbb{E}\left[F(x_{k+1}^s)-f(x_k^s)\right]\\
\leq &\mathbb{E}\left[\left\langle \beta_s(\widetilde{\nabla}_k^s-\nabla f(x_k^s)+\nabla f(x_k^s)), x^*-z_k^s\right\rangle+\frac{\beta_s^2}{2\eta}(\|z_k^s-x^*\|^2-\|z_{k+1}^s-x^*\|^2)\right]\\
&+\mathbb{E}\left[\beta_s h(x^*)+(1-\beta_s)h(\widetilde{x}^{s-1})\right]\\
&+\frac{1}{2\theta_\eta L}\mathbb{E}\left[\left\|\nabla f(x_k^s)-\widetilde{\nabla}_k^s\right\|^2\right]-\frac{\theta_\eta L}{2}\left\|x_{k+1}^s-x_k^s\right\|^2\\
\overset{a}{=}&\mathbb{E}\left[\frac{\beta_s^2}{2\eta}(\|z_k^s-x^*\|^2-\|z_{k+1}^s-x^*\|^2)+\beta_s h(x^*)\right]\\
&+\mathbb{E}\left[\left\langle \nabla f(x_k^s), \beta_s x^*+(1-\beta_s)\widetilde{x}^{s-1}-x_k^s\right\rangle\right]\\
&+\mathbb{E}\left[\left\langle -\nabla f_{i_k^s}(\widetilde{x}^{s-1})+\nabla f(\widetilde{x}^{s-1}), \beta_s x^*+(1-\beta_s)\widetilde{x}^{s-1}-x_k^s\right\rangle\right]\\
&+\frac{1}{2\theta_\eta L}\mathbb{E}\left[\left\|\nabla f(x_k^s)-\widetilde{\nabla}_k^s\right\|^2\right]-\frac{\theta_\eta L}{2}\left\|x_{k+1}^s-x_k^s\right\|^2\\
&+(1-\beta_s)\mathbb{E}[h(\widetilde{x}^{s-1})]+\mathbb{E}\left[\left\langle \beta_s(\widetilde{\nabla}_k^s-\nabla f(x_k^s)), x^*-z_k^s\right\rangle\right]
\end{aligned}
\tag{33}
$$

The equality $\overset{a}{=}$ is obtained by definition of $\widetilde{\nabla}_k^s$ and rearranging terms. By using

$$
\mathbb{E}\left\langle -\nabla f_{i_k^s}(\widetilde{x}^{s-1})+\nabla f(\widetilde{x}^{s-1}), \beta_s x^*+(1-\beta_s)\widetilde{x}^{s-1}-x_k^s\right\rangle=0,
$$

in (33), we have

$$
\begin{aligned}
&\mathbb{E}\left[F(x_{k+1}^s)-f(x_k^s)\right]\\
\leq &\mathbb{E}\left[\frac{\beta_s^2}{2\eta}(\|z_k^s-x^*\|^2-\|z_{k+1}^s-x^*\|^2)\right]\\
&+\mathbb{E}\left[\beta_s h(x^*)+(1-\beta_s)h(\widetilde{x}^{s-1})\right]\\
&+\mathbb{E}\left[\left\langle \nabla f(x_k^s), \beta_s x^*+(1-\beta_s)\widetilde{x}^{s-1}-x_k^s\right\rangle\right]\\
&+\frac{1}{2\theta_\eta L}\mathbb{E}\left[\left\|\nabla f(x_k^s)-\widetilde{\nabla}_k^s\right\|^2\right]-\frac{\theta_\eta L}{2}\left\|x_{k+1}^s-x_k^s\right\|^2\\
&+\mathbb{E}\left[\left\langle \beta_s(\widetilde{\nabla}_k^s-\nabla f(x_k^s)), x^*-z_k^s\right\rangle\right].
\end{aligned}
\tag{34}
$$

Additionally, we have the following,

$$
\begin{aligned}
&\left\langle \nabla f(x_k^s), \beta_s x^*+(1-\beta_s)\widetilde{x}^{s-1}-x_k^s\right\rangle\\
&\overset{a}{\leq} \beta_s f(x^*)+(1-\beta_s)f(\widetilde{x}^{s-1})-f(x_k^s)-\frac{\nu}{2}\|x^*-z_k^s\|^2,
\end{aligned}
\tag{35}
$$

where in inequalities $\overset{a}{\leq}$ and $\overset{b}{\leq}$ we used the strong convexity of f.

$$
\mathbb{E}\left[\left\langle \beta_s(\widetilde{\nabla}_k^s-\nabla f(x_k^s)), x^*-z_k^s\right\rangle\right]\leq\frac{\beta_s^2}{2\nu}\left\|\widetilde{\nabla}_k^s-\nabla f(x_k^s)\right\|+\frac{\nu}{2}\|x^*-z_k^s\|^2.
\tag{36}
$$

Since z^s_{k+1} is the optimal solution of proximal subproblem in the Algorithm 2, there exists $\xi^s_{k+1} \in \partial h(z^s_{k+1})$ satisfying

$$\beta_s(z^s_{k+1} - z^s_k) + \eta\, \widetilde{\nabla}^s_k + \eta\, \xi^s_{k+1} = 0. \tag{37}$$

We obtain from (34)

$$\begin{aligned}\mathbb{E}[F(x^s_{k+1})] \leq & \beta_s F(x^*) + (1-\beta_s)F(\widetilde{x}^{s-1}) \\ & + \frac{\beta_s^2}{2\eta}\mathbb{E}[\|z^s_k - x^*\|^2 - \left\|z^s_{k+1} - x^*\right\|^2] \\ & + (\frac{1}{2\theta_\eta L} + \frac{\beta_s^2}{2\nu})\mathbb{E}\left[\left\|\nabla f(x^s_k) - \widetilde{\nabla}^s_k\right\|^2\right] - \frac{\theta_\eta L}{2}\left\|x^s_{k+1} - x^s_k\right\|^2.\end{aligned} \tag{38}$$

Equivalently, we have the following

$$\begin{aligned}\mathbb{E}[F(x^s_{k+1}) - F(x^*)] \leq & (1-\beta_s)[F(\widetilde{x}^{s-1}) - F(x^*)] \\ & + \frac{\beta_s^2}{2\eta}\mathbb{E}[\|z^s_k - x^*\|^2 - \left\|z^s_{k+1} - x^*\right\|^2] \\ & + (\frac{1}{2\theta_\eta L} + \frac{\beta_s^2}{2\nu})\mathbb{E}\left[\left\|\nabla f(x^s_k) - \widetilde{\nabla}^s_k\right\|^2\right] - \frac{\theta_\eta L}{2}\left\|x^s_{k+1} - x^s_k\right\|^2.\end{aligned} \tag{39}$$

From convexity of function F and the definition $\widetilde{x}^s = \frac{1}{m_s}\sum_{k=0}^{m_s-1} x^s_{k+1}$, we get

$$\begin{aligned} & F(\widetilde{x}^s) = F(\frac{1}{m_s}\sum_{k=0}^{m_s-1} x^s_{k+1}) \leq \frac{1}{m_s}\sum_{k=0}^{m_s-1} F(x^s_{k+1}) \\ & \widetilde{x}^s = x^s_{m_s}\end{aligned} \tag{40}$$

Using the above inequality and (39) we obtain

$$\begin{aligned}\mathbb{E}[F(\widetilde{x}^s) - F(x^*)] \leq & (1-\beta_s)[F(\widetilde{x}^{s-1}) - F(x^*)] \\ & + \frac{\beta_s^2}{2\eta m_s}\mathbb{E}[\|z^s_0 - x^*\|^2 - \left\|z^s_{m_s} - x^*\right\|^2] \\ & + \frac{1}{2\theta_\eta L m_s}\sum_{k=0}^{m_s-1}\mathbb{E}\left[\left\|\nabla f(x^s_k) - \widetilde{\nabla}^s_k\right\|^2\right] - \frac{\theta_\eta L}{2m_s}\sum_{k=0}^{m_s-1}\left\|x^s_{k+1} - x^s_k\right\|^2.\end{aligned} \tag{41}$$

We define $\psi_k^s = F(x_k^s) - F(x^*)$. We have

$$\begin{aligned}
\sum_{k=0}^{m_s-1} \mathbb{E}[\psi_{k+1}^s - \psi_k^s] \leq & -\beta_s[F(\widetilde{x}^{s-1}) - F(x^*)] \\
& + \frac{\beta_s^2}{2\eta}\mathbb{E}[\|z_0^s - x^*\|^2 - \left\|z_{m_s}^s - x^*\right\|^2] \\
& + (\frac{1}{2\theta_\eta L} + \frac{\beta_s^2}{2\nu})\sum_{k=0}^{m_s-1} \mathbb{E}\left[\left\|\nabla f(x_k^s) - \widetilde{\nabla}_k^s\right\|^2\right] - \frac{\theta_\eta L}{2}\sum_{k=0}^{m_s-1}\left\|x_{k+1}^s - x_k^s\right\|^2 \\
\leq & -\beta_s[F(\widetilde{x}^{s-1}) - F(x^*)] \\
& + \frac{\beta_s^2}{2\eta}\mathbb{E}[\|z_0^s - x^*\|^2 - \left\|z_{m_s}^s - x^*\right\|^2] \\
& + (\frac{Ld}{\theta_\eta} + \frac{L^2 d\beta_s^2}{\nu})\sum_{k=0}^{m_s-1} \mathbb{E}\left\|x_k^s - \widetilde{x}^{s-1}\right\|^2 + \frac{Ld^2\mu^2 m_s}{4\theta_\eta} + \frac{L^2 d^2\mu^2\beta_s^2}{\nu} \\
& - \frac{\theta_\eta L}{2}\sum_{k=0}^{m_s-1}\left\|x_{k+1}^s - x_k^s\right\|^2
\end{aligned} \tag{42}$$

Next, we define an useful Lyapunov function as follows:

$$R_k^s = \mathbb{E}\left[\psi_k^s + c_k\left\|x_k^s - \widetilde{x}^s\right\|^2\right], \tag{43}$$

where $\{c_k\}$ is a nonnegative sequence. Considering the upper bound of $\left\|x_{k+1}^s - \widetilde{x}^s\right\|^2$, we have

$$\begin{aligned}
\left\|x_{t+1}^s - \widetilde{x}^{s-1}\right\|^2 &= \left\|x_{k+1}^s - x_k^s + x_k^s - \widetilde{x}^{s-1}\right\|^2 \\
&= \left\|x_{k+1}^s - x_k^s\right\|^2 + 2\left\langle x_{k+1}^s - x_k^s, x_k^s - \widetilde{x}^s\right\rangle + \left\|x_k^s - \widetilde{x}^{s-1}\right\|^2 \\
&\leq \left\|x_{k+1}^s - x_k^s\right\|^2 + 2(\frac{1}{2\beta}\left\|x_{k+1}^s - x_k^s\right\|^2 + \frac{\beta}{2}\left\|x_k^s - \widetilde{x}^{s-1}\right\|^2) + \left\|x_k^s - \widetilde{x}^{s-1}\right\|^2 \\
&= (1 + \frac{1}{\beta})\left\|x_{k+1}^s - x_k^s\right\|^2 + (1+\beta)\left\|x_k^s - \widetilde{x}^{s-1}\right\|^2,
\end{aligned} \tag{44}$$

where $\beta > 0$. Then we have

$$\begin{aligned}
&\sum_{k=0}^{m_s-1}\mathbb{E}[R^s_{k+1}-\psi^s_k] = \sum_{k=0}^{m_s-1}\mathbb{E}\left[\psi^s_{k+1}+c_{k+1}\left\|x^s_{k+1}-\tilde{x}^{s-1}\right\|^2-\psi^s_k\right]\\
&\le \sum_{k=0}^{m_s-1}\mathbb{E}\left[\psi^s_{k+1}+c_{k+1}(1+\frac{1}{\beta})\left\|x^s_{k+1}-x^s_k\right\|^2+c_{k+1}(1+\frac{1}{\beta})\left\|x^s_k-\tilde{x}^{s-1}\right\|^2-\psi^s_k\right]\\
&\le \sum_{k=0}^{m_s-1}\mathbb{E}\left[\psi^s_{k+1}+c_{k+1}(1+\frac{1}{\beta})\left\|x^s_{k+1}-x^s_k\right\|^2+c_{k+1}(1+\beta)\left\|x^s_k-\tilde{x}^{s-1}\right\|^2-\psi^s_k\right]\\
\le &-\beta_s[F(\tilde{x}^{s-1})-F(x^*)]\\
&+\frac{\beta_s^2}{2\eta}\mathbb{E}[\|z^s_0-x^*\|^2-\left\|z^s_{m_s}-x^*\right\|^2]\\
&+\sum_{k=0}^{m_s-1}(c_{k+1}(1+\beta)+\frac{Ld}{\theta_\eta}+\frac{L^2d\beta_s^2}{\nu})\mathbb{E}\left\|x^s_k-\tilde{x}^{s-1}\right\|^2+\frac{Ld^2\mu^2m_s}{4\theta_\eta}\\
&+\sum_{k=0}^{m_s-1}(c_{k+1}(1+\frac{1}{\beta})-\frac{\theta_\eta L}{2})\left\|x^s_{k+1}-x^s_k\right\|^2\\
\le &-\beta_s[F(\tilde{x}^{s-1})-F(x^*)]\\
&+\frac{\beta_s^2}{2\eta}\mathbb{E}[\|z^s_0-x^*\|^2-\left\|z^s_{m_s}-x^*\right\|^2]\\
&+\sum_{k=0}^{m_s-1}c_k\mathbb{E}\left\|x^s_k-\tilde{x}^{s-1}\right\|^2+\frac{Ld^2\mu^2m_s}{4\theta_\eta}+\frac{L^2d^2\mu^2\beta_s^2}{\nu}\\
&+\sum_{k=0}^{m_s-1}(c_{k+1}(1+\frac{1}{\beta})-\frac{\theta_\eta L}{2})\left\|x^s_{k+1}-x^s_k\right\|^2
\end{aligned}\tag{45}$$

where $c_k = c_{k+1}(1+\frac{1}{\beta})+\frac{Ld}{\theta_\eta}$. Let $c_m = 0$, $\beta = \frac{1}{m}$ and $\eta =$. Recursing on k, we have

$$\begin{aligned}
c_k &= c_{k+1}(1+\beta)+\frac{Ld}{\theta_\eta}+\frac{L^2d\beta_s^2}{\nu}\\
c_t &= (\frac{Ld}{\theta_\eta}+\frac{L^2d\beta_s^2}{\nu})\frac{(1+\beta)^{m-k}-1}{\beta}\\
&= (\frac{Ldm}{\theta_\eta}+\frac{L^2d\beta_s^2m}{\nu})((1+\frac{1}{m})^m-1)\\
&\le (\frac{Ldm}{\theta_\eta}+\frac{L^2d\beta_s^2m}{\nu})(e-1)\\
&\le 2(\frac{Ldm}{\theta_\eta}+\frac{L^2d\beta_s^2m}{\nu})
\end{aligned}\tag{46}$$

It follows that

$$\begin{aligned} c_{k+1}(1+\frac{1}{\beta}) &\leq 2(\frac{Ldm}{\theta_\eta}+\frac{L^2 d\beta_s^2 m}{\nu})(1+m) \\ &\leq 4\frac{Ldm^2}{\theta_\eta} \leq \frac{\theta_\eta L}{2} - 4\frac{L^2 d\beta_s^2 m^2}{\nu} \end{aligned} \tag{47}$$

It is sufficient to have

$$\begin{aligned} 8dm^2 + 8\frac{\theta_\eta Ldm^2}{\nu} &\leq \frac{\theta_\eta^2}{2} \\ 1+\frac{\theta_\eta L}{\nu} &\leq \frac{\theta_\eta^2}{8dm^2} \end{aligned} \tag{48}$$

So, it is sufficient

$$\theta_\eta \geq 4dm^2\left[\frac{L}{\nu}+\sqrt{\frac{L^2}{\nu^2}+\frac{1}{2dm^2}}\right]$$

Then we have

$$\begin{aligned} &\sum_{k=0}^{m_s-1}\mathbb{E}[R_{k+1}^s-\psi_k^s] \\ \leq &-\beta_s[F(\widetilde{x}^{s-1})-F(x^*)] \\ &+\frac{\beta_s^2}{2\eta}\mathbb{E}[\|z_0^s-x^*\|^2-\left\|z_{m_s}^s-x^*\right\|^2] \\ &+c_k\sum_{k=0}^{m_s-1}\mathbb{E}\,\|x_k^s-\widetilde{x}^s\|^2+\frac{Ld^2\mu^2 m_s}{4\theta_\eta}+\frac{L^2d^2\mu^2\beta_s^2}{\nu} \end{aligned} \tag{49}$$

Equivalently, we obtain

$$\begin{aligned} &\sum_{k=0}^{m_s-1}\mathbb{E}[R_{k+1}^s-R_k^s]=\psi_{m_s}^s-\psi_0^s+\sum_{k=1}^{m_s-1}(c_{k-1}-c_k)E\left\|x_k^s-\widetilde{x}^{s-1}\right\|^2 \\ &+c_{m_s}\left\|x_{m_s}^s-\widetilde{x}^{s-1}\right\|^2-c_0\left\|x_0^s-\widetilde{x}^{s-1}\right\|^2 \\ \leq &-\beta_s[F(\widetilde{x}^{s-1})-F(x^*)] \\ &+\frac{\beta_s^2}{2\eta}\mathbb{E}[\|z_0^s-x^*\|^2-\left\|z_{m_s}^s-x^*\right\|^2] \\ &+\frac{Ld^2\mu^2 m_s}{4\theta_\eta}+\frac{L^2d^2\mu^2\beta_s^2}{\nu} \end{aligned} \tag{50}$$

After rearranging, we can derive

$$
\begin{aligned}
[F(\widetilde{x}^s) - F(x^*)] \leq & (1-\beta_s)[F(\widetilde{x}^{s-1}) - F(x^*)] \\
& + \frac{\beta_s^2}{2\eta}\mathbb{E}[\|z_0^s - x^*\|^2 - \left\|z_{m_s}^s - x^*\right\|^2] + \frac{Ld^2\mu^2 m_s}{4\theta_\eta} + \frac{L^2 d^2 \mu^2 \beta_s^2}{\nu} \\
\leq & (1-\beta_s)[F(\widetilde{x}^{s-1}) - F(x^*)] \\
& + \frac{\beta_s^2}{2\eta}\mathbb{E}[\|z_0^s - x^*\|^2 - \left\|z_{m_s}^s - x^*\right\|^2] + \frac{Ld^2\mu^2 m_s}{4\theta_\eta} + \frac{L^2 d^2 \mu^2 \beta_s^2}{\nu}. \quad (51)
\end{aligned}
$$

□