Asynchronous Stochastic Proximal Optimization Algorithms with Variance Reduction

Qi Meng1, Wei Chen2, Jingcheng Yu3, Taifeng Wang2, Zhi-Ming Ma4, Tie-Yan Liu2

1 School of Mathematical Sciences, Peking University, qimeng13@pku.edu.cn
2 Microsoft Research, {wche, taifengw, tie-yan.liu}@microsoft.com
3 Fudan University, JingchengYu.94@gmail.com
4 Academy of Mathematics and Systems Science, Chinese Academy of Sciences, mazm@amt.ac.cn

Abstract

Regularized empirical risk minimization (R-ERM) is an important branch of machine learning, since it constrains the capacity of the hypothesis space and guarantees the generalization ability of the learning algorithm. Two classic proximal optimization algorithms, i.e., proximal stochastic gradient descent (ProxSGD) and proximal stochastic coordinate descent (ProxSCD) have been widely used to solve the R-ERM problem. Recently, variance reduction technique was proposed to improve ProxSGD and ProxSCD, and the corresponding ProxSVRG and ProxSVRCD have better convergence rate. These proximal algorithms with variance reduction technique have also achieved great success in applications at small and moderate scales. However, in order to solve large-scale R-ERM problems and make more practical impacts, the parallel version of these algorithms are sorely needed. In this paper, we propose asynchronous ProxSVRG (Async-ProxSVRG) and asynchronous ProxSVRCD (Async-ProxSVRCD) algorithms, and prove that Async-ProxSVRG can achieve near linear speedup when the training data is sparse, while Async-ProxSVRCD can achieve near linear speedup regardless of the sparse condition, as long as the number of block partitions are appropriately set. We have conducted experiments on a regularized logistic regression task. The results verified our theoretical findings and demonstrated the practical efficiency of the asynchronous stochastic proximal algorithms with variance reduction.

1 Introduction

In this paper, we focus on the regularized empirical risk minimization (R-ERM) problem, whose objective is a finite sum of smooth convex loss functions $f_i(x)$ plus a non-smooth regularization term $R(x)$, i.e.,

$$
\min_{x \in \mathbb{R}^d} P(x) = F(x) + R(x) = \frac{1}{n} \sum_{i=1}^n f_i(x) + R(x).
$$

In particular, in the context of machine learning, $f_i(x)$ and $R(x)$ are defined as follows. Suppose we are given a collection of training data $(a_1, b_1), \ldots, (a_n, b_n)$, where each $a_i \in \mathbb{R}^d$ is an input feature vector and $b_i \in \mathbb{R}$ is the output variable. The loss function $f_i(x)$ measures the fitness of the model $x$ on training data $(a_i, b_i)$. Different learning tasks may use different loss functions, such as the least square loss $\frac{1}{2}(a_i^T x - b_i)^2$ for regression and the logistic loss $\log(1 + \exp(-b_i a_i^T x))$ for classification. The regularization term is used to constrain the capacity of the hypothesis space. For example, the non-smooth $L_1$ regularization term is widely used.

In order to solve the R-ERM problem, the proximal stochastic gradient descent method (ProxSGD) has been widely used, which exploits the additive nature of the empirical risk function and updates the model based on the gradient which is calculated at randomly sampled training data. However, the random sampling in ProxSGD introduces non-negligible variance, which makes that we need to use a decreasing step size (also known as learning rate) to guarantee the algorithm’s convergence, and the convergence rate is only sublinear (Langford, Li, and Zhang 2009; Rakhlin, Shamir, and Sridharan 2011). To tackle this problem, people have developed a set of new technologies. For example, in (Xiao and Zhang 2014), a variance reduction technique was introduced to improve ProxSGD and a new algorithm called ProxSVRG was proposed. It has been proven that even with a constant step size, ProxSVRG can achieve linear convergence rate.

Proximal stochastic coordinate descent (ProxSCD) is another method which is used to solve the R-ERM problem (Shalev-Shwartz and Tewari 2011). Since the variance introduced by the coordinate sampling asymptotically goes to zero, the ProxSCD attains linear convergence rate when the objective function $P(x)$ is strongly convex (Wright 2015). However, ProxSCD still requires that all component functions in the empirical risk are accessible in each iteration, which is time consuming. In (Zhao et al. 2014), a new algorithm called ProxSVRCD (also known as MRBCD) was proposed to improve ProxSCD. This algorithm, in addition to randomly samples a block of coordinates, also randomly samples training data in each iteration and uses the variance reduction technique. It has been proven that ProxSVRCD can achieve linear convergence rate and outperform ProxSCD by a lower iteration complexity.

While the aforementioned new algorithms (i.e., ProxSVRG and ProxSVRCD) have both good theoretical properties and empirical performances, the investigations on them were mainly conducted in the sequential (single-machine) setting. In this big data era, we usually need to deal with very large scale R-ERM problems. In this case, sequential algorithms usually cost too much time. To tackle the challenge, paral-
we conduct theoretical analysis for Async-ProxSVRG and Async-ProxSVRCD. According to our results: (1) Async-ProxSVRG and Async-ProxSVRCD can converge; (2) Async-ProxSVRCD also have good speedup, and is more highly efficient and scalable for practical use.

In this section, we will briefly introduce proximal algorithms with variance reduction, and then review the existing convergence analysis for asynchronous parallel algorithms.

**ProxSGD and ProxSCD**

At first, let us briefly introduce the standard stochastic proximal gradient algorithms, i.e., ProxSGD and ProxSCD. With ProxSGD, at iteration $k$, the solution to the R-ERM problem, i.e., Eqn (4) is as follows:

$$x_{k+1} = \text{prox}_{\eta_k R} \{ x_k - \eta_k \nabla f_{B_k}(x_k) \},$$  

where $\eta_k$ is the step size, $B_k$ is a mini-batch of randomly selected training data, $\nabla f_{B_k}(x_k) = \frac{1}{|B_k|} \sum_{i \in B_k} \nabla f_i(x_k)$ and the proximal mapping is defined as $\text{prox}_R(y) = \arg\min_{x \in \mathbb{R}^d} \{ \frac{1}{2} \| x - y \|^2 + R(x) \}$.

ProxSGD exploits the block separability of the regularization term $R$ in the R-ERM problem, i.e., $R(x) = \sum_{j=1}^m R_j(x,C_j)$, where $x,C_j$ is the $j$-th coordinate block of $x$. For example, for the L1-norm regularizer, $\{C_j ; j = 1, \ldots, m\}$ is a partition of $\{1, \ldots, d\}$ with $m = \text{block size}$, and $R_j(x,C_j) = \sum_{i \in C_j} |x_i|$. ProxSCD randomly selects a coordinate block and update the coordinates in that block based on their gradients while keep the value of the other coordinates unchanged, i.e.,

$$x_{k+1,C_jk} = \text{prox}_{\eta_j R_j} \left\{ x_{k,C_jk} - \eta \nabla C_j F(x_{k-1}) \right\}$$  

where $C_jk$ is the coordinate block sampled at iteration $k$ and $\nabla C_j F(x) = [\nabla F(x)]_{C_j}$.

**Proximal Algorithms with Variance Reduction**

For ProxSGD, the step size $\eta_k$ has to be decreasing in order to mitigate the variance introduced by random sampling, which usually leads to slow convergence. To tackle this problem, one of the most popular variance reduction techniques was proposed by Johnson and Zhang (Johnson and Zhang 2013). Xiao and Zhang applied this variance reduction technique to improve ProxSGD, and a new algorithm called ProxSVRG was proposed (Xiao and Zhang 2014).

The ProxSVRG algorithm divides the optimization process into multiple stages. At the beginning of stage $s$, ProxSVRG calculates the full gradient at the current solution $\hat{x}_{s-1}$, i.e., $\nabla F(\hat{x}_{s-1})$. Then, at iteration $k$ inside stage $s$, the solution is updated as follows:

$$v_k = \nabla f_{B_k}(x_k) - \nabla f_{B_k}(\hat{x}_{s-1}) + \nabla F(\hat{x}_{s-1})$$  

$$x_{k+1} = \text{prox}_{\eta_k R} \{ x_k - \eta_k v_k \},$$  

where $-\nabla f_{B_k}(\hat{x}_{s-1}) + \nabla F(\hat{x}_{s-1})$ is the variance reduction regularization term.

For ProxSCD, since the variance introduced by the block selection asymptotically goes to zero, it attains linear convergence rate. However, it still requires that all component functions are accessible within every iteration. Zhao et al. used variance reduction technique to improve ProxSCD with random training data sampling and a new algorithm called ProxSVRCD was proposed (Zhao et al. 2014).

This paper is organized as follows: in Section 2, we briefly introduce the stochastic proximal algorithms with variance reduction including ProxSVRG and ProxSVRCD, and then related works; in Section 3, we describe the asynchronous parallelization of these algorithms; in Section 4, we prove the convergence rates for Async-ProxSVRG and Async-ProxSVRCD; in Section 5, we report the experimental results and make discussions; finally, in the last section, we conclude the paper and present future research directions.


ProxSVRCD is similar to ProxSVRG, the update formula for iteration $k$ inside stage $s$ takes the following form:

$$
v_k = \nabla f_{B_k}(x_k) - \nabla f_{B_k}(\tilde{x}_{s-1}) + \nabla F(\tilde{x}_{s-1}),
$$

$$
x_{k+1,C_k} = \text{prox}_{\eta_k C_k} \left\{ x_k,C_k - \eta_k v_k,C_k \right\},
$$

$$
x_{k+1,C_k} \leftarrow x_{k,C_k},
$$

where $-\nabla f_{B_k}(\tilde{x}_{s-1}) + \nabla F(\tilde{x}_{s-1})$ is the variance reduction regularization term.

Existing Convergence Analysis of Asynchronous Parallel Algorithms

The asynchronous parallel methods have been successfully applied to accelerate many optimization algorithms including stochastic gradient descent (SGD) (Agarwal and Duchi 2011; Feyzmahdavian, Aytekin, and Johansson 2015; Recht et al. 2011; Mania et al. 2015), stochastic coordinate descent (SCD) (Liu et al. 2013; Liu and Wright 2015), stochastic dual coordinate ascent (SDCA) (Tran et al. 2015) and randomized Kaczmarz algorithm (Liu, Wright, and Sridhar 2014). However, to the best of our knowledge, the asynchronous parallel versions of ProxSVRG and ProxSVRCD are not well studied, as well as their theoretical properties.

We briefly review the works which are closely related to ours as follows. Reddi et al. studied asynchronous SVRG and proved that, asynchronous SVRG can achieve near linear speedup under some sparse condition (Reddi et al. 2015). Liu and Wright analyzed the asynchronous ProxSCD. They proved that the asynchronous ProxSCD can achieve near linear speedup if the delay is bounded by $O(d^2)$, where $d$ is the input dimension (Liu and Wright 2015).

However, to the best of our knowledge, there is no study on the asynchronous parallel versions of proximal algorithms with variance reduction, as well as their theoretical properties.

3 Asynchronous Proximal Algorithms with Variance Reduction

In this section, we describe our Async-ProxSVRG and Async-ProxSVRCD algorithms under the following asynchronous parallel architecture. Suppose there are $P$ local workers and one master. For local workers, each of them has full access to the training data and stores a non-overlapping partition $N_p$ ($p = 1, \ldots, P$) of the training data. Each local worker independently communicates with the master to pull the global parameters from the master, and it computes the stochastic gradients locally and then push the gradients to the master. For the master, it maintains the global model. It updates the model parameters with the gradient pushed by local workers and sends the model parameters to local workers when it receives the pull request. Master can control the access conflict based on different granularity. In Async-ProxSVRG, the local worker will access the entire model in every update. Therefore, we let master only response to one local worker’s request at one time, which means the global model is atomic for all workers. In Async-ProxSVRCD, the local worker will only access a coordinate block in every update and different workers might work on different blocks without interfering others. In this case, master will response to multiple local workers simultaneously if only they are not accessing the same coordinate block, which means the global model is atomic at coordinate block level.

With variance reduction technique, the optimization process is divided into multiple stages (i.e., outer loop: $s = 1, \ldots, S$). In each stage, there are two phases: full gradient computation and solution updates (i.e., inner loop: $k = 1, \ldots, K$).

**Full gradient computation:** the workers collectively compute the full gradient in parallel based on the entire training data. Specifically, each worker pulls the master parameter from the master, computes the gradients over one part of the training data, and pushes the sum of the gradients to the master. Then the master aggregates the gradients from the workers to obtain the full gradient, and broadcasts it to the workers.

**Solution updates:** the workers compute the VR-regularized stochastic gradient in an asynchronous way and the master makes updates according to the proximal algorithms. To be specific, at iteration $k$, one local worker (who just finished its local computation) pulls the master parameters from the master, computes the VR-regularized stochastic gradient according to Eqn (4) for ProxSVRG or Eqn (6) for ProxSVRCD, and then pushes it to the master without any synchronization with the other workers. After the master receives the VR-regularized gradient from this worker, it updates the master parameter according to Eqn (5) for ProxSVRG or Eqn (7) for ProxSVRCD. Then the global clock becomes $k + 1$, and the next iteration begins. Corresponding details can be found in Algorithm 1.

Please note that, the gradient pushed by a local worker to the master could be delayed. The reason is, when the worker is working on its own local computation, other workers might finish their computations and push their gradients to the master, and the master updates the master parameter accordingly.

As aforementioned, for Async-ProxSVRG, the whole model is atomic to each worker’s access. When the worker 0 is working on its own local computation, worker 1 and worker 2 might finish their computations, pushed their gradients to the master, and the master updates the master parameter accordingly. Thus, when worker 0 finish its computation and push it to the master, the global clock has already plus 2. Thus, the local gradients have delay=2 for the current master parameter. We use a random variable $\tau_k$ to denote the delay of local gradients received by the master at global clock $k$.

The delay equals to the number of updates that other workers have committed to the master between one particular worker pulls the master parameter at global clock $k - \tau_k$. We denote the finally pulled parameter as $\tilde{x}_k$, which can be represented as below:

$$
\tilde{x}_k = x_{k-\tau_k} + \sum_{h \in J(k)} (x_{k+1} - x_h),
$$

where $J(k) \subset \{k - \tau_k, \ldots, k - 1\}$. The $k$-th update can
be described as \( x_{k+1} = \text{prox}_{\eta B_k} \left\{ x_k, C_{j_k} - \eta u_{k,i} C_{j_k} \right\} \),
where \( u_k = \nabla f_{B_k} (\hat{x}_k) - \nabla f_{B_k} (\hat{x}) + \nabla F (\hat{x}) \). The delay \( \tau_k \) equals to the difference between the clock at which a local worker pulls the first coordinate block from the master and the clock at which the local worker pushes the gradients to the master.

We conduct theoretical analysis for Async-ProxSVRG and Async-ProxSVRCD based on the above setting in the next section. Like other asynchronous parallel algorithms, the delay also plays an important role in the convergence rate of asynchronous proximal algorithms with variance reduction.

**Algorithm 1** Async-ProxSVRG and Async-ProxSVRCD

Require: initial vector \( \hat{x}_0 \), step size \( \eta \), number of inner loops \( K \), size of mini-batch \( B \), number of coordinate blocks \( m \).

Ensure: \( \hat{x}_K \)

for \( s = 1, 2, ..., S \) do

\( \hat{x} = \hat{x}_{s-1}, x_0 = \hat{x} \)

For local worker \( p \): calculate \( \nabla F_p (\hat{x}) = \sum_{i \in N_p} \nabla f_i (\hat{x}) \) and send it to the master.

For master: calculate \( \nabla F (\hat{x}) = \frac{1}{n} \sum_{p=1}^{P} \nabla F_p (\hat{x}) \) and send it to each local worker.

for \( k = 1, ..., K \) do

1. Async-ProxSVRG: consistent read

   For local worker \( p \): randomly select a mini-batch \( B_k \) with \( |B_k| = B \).

   Pull current state \( x_{k-\tau_k} \) from the master.

   Compute \( u_k = \nabla f_{B_k} (x_{k-\tau_k}) - \nabla f_{B_k} (\hat{x}) + \nabla F (\hat{x}) \).

   Push \( u_k \) to the master.

   For master:

   Update \( x_{k+1} = \text{prox}_{\eta B_k} (x_k - \eta u_k) \).

2. Async-ProxSVRCD: inconsistent read

   For local worker \( p \): randomly select \( B_k \) with \( |B_k| = B \), and randomly select \( j_k \in [m] \).

   Pull current state \( \hat{x}_k \) from the master.

   Compute \( u_k = \nabla f_{B_k} (\hat{x}_k) - \nabla f_{B_k} (\hat{x}) + \nabla F (\hat{x}) \).

   Push \( u_k \) to the master.

   For master:

   Update \( x_{k+1} = \text{prox}_{\eta B_k} \left\{ x_k, C_{j_k} - \eta u_{k,i} C_{j_k} \right\} \;

   x_{k+1} \langle C_{j_k} \leftarrow x_k \langle C_{j_k} \;

end for

\( \hat{x}_K = \frac{1}{K} \sum_{k=1}^{K} x_k \)

end for

4 Convergence Analysis

In this section, we prove the convergence rates of the asynchronous parallel proximal algorithms with variance reduction introduced in the previous section.

Async-ProxSVRG

At first, we introduce the following assumptions, which are very common in the theoretical analysis for asynchronous parallel algorithms [Recht et al. 2011; Reddi et al. 2013].

**Assumption 1**: (Convexity) \( F(x) \) and \( R(x) \) are convex and \( R(x) \) is block separable. The objective function \( P(x) \) is \( \mu \)-strongly convex, i.e., \( \forall x, y \in \mathbb{R}^d \), we have,

\[ P(y) \geq P(x) + \langle \nabla P(x), y - x \rangle + \frac{\mu}{2} \| y - x \|^2, \forall x \in \partial P(x) \]

**Assumption 2**: (Smoothness) The components \( \{ f_i(x) \}; i \in \{ n \} \) of \( F(x) \) are differentiable and have Lipschitz continuous partial gradients and thus Lipschitz continuous gradients, i.e., \( \exists L > 0 \), such that \( \forall x, y \in \mathbb{R}^d \) with \( x_k \neq y_j \), we have

\[ \| \nabla_j f(x_k) - \nabla_j f(y) \| \leq T \| x_k - y_j \|, \forall i \in [n], j \in [d] \]

\[ \| \nabla_j f(x_k) - \nabla_j f(y) \| \leq L \| x_k - y_j \|, \forall i \in [n], j \in [d] \]

**Assumption 3**: (Bounded and Independent Delay) The random delay variables \( \tau_1, \tau_2, ..., \) in consistent read setting are independent of each other and independent of \( B_k \), and their expectations are upper bounded by \( \tau \), i.e., \( \mathbb{E} \tau_k \leq \tau \) for all \( k \).

**Assumption 4**: (Data Sparsity) The maximal frequency of a feature appearing in the dataset is upper bounded by \( \Delta \).

Based on these assumptions, we prove that Async-ProxSVRG has linear convergence rate.

**Theorem 4.1**: Suppose Assumptions 1-4 hold. If the step size \( \eta \) is \( \eta \) is \( \tau \) such that

\[ \rho = \frac{B}{\eta K (B - 8\eta L)} + \frac{8\eta L}{(B - 8\eta L)} < 1, \]

then Async-ProxSVRG has linear convergence rate in expectation:

\[ \mathbb{E} \| x_k - P(x^*) \| \leq \rho^k \| x_0 - P(x^*) \|, \]

where \( x^* = \text{argmin}_{x \in \mathbb{R}^d} P(x) \).

Due to space limitation, we only provide the proof sketch and put the proof details into supplementary materials.

**Proof Sketch of Theorem 4.1**

Firstly we introduce some notations. Let \( x_{k+1} = x_k - \eta g_k \; v_k = \nabla f_{B_k} (x_k) - \nabla f_{B_k} (\hat{x}) + \nabla F (\hat{x}), \) and \( u_k = \nabla f_{B_k} (x_{k-\tau_k}) - \nabla f_{B_k} (\hat{x}) + \nabla F (\hat{x}) \).

**Step 1**: The key for the proof is that by the sparsity condition, we have \( | F(x) \geq F(y) - \nabla F(x)(y - x) - \frac{L\Delta}{2} \| x - y \|^2 \).

**Step 2**: By using the convexity of \( F(x) \) and \( R (x) \), we have:

\[ P(x^*) \geq P(x_{k+1}) + \langle u_k - \nabla F (x_{k-\tau_k}) \rangle^T (x_{k+1} - x^*) \]

\[ + \eta \| g_k \|^2 - \frac{L\Delta}{2} \sum_{h=0}^{K} \| g_h \|^2 + g_h^T (x^* - x_{k+1}). \]

**Step 3**: We use Lemma 3 in [Xiao and Zhang 2014] to bound the term \( \mathbb{E} g_h^T (x_{k-\tau_k} - x^*) + \langle \eta - L\Delta^2 \| x_{k-\tau_k} \| \rangle \sum_{h=0}^{K} \mathbb{E} g_h^T (x_{k-\tau_k} - x^*) + \frac{8\eta L}{(B - 8\eta L)} (K + 1) (P(\hat{x}) - P(x^*)). \]

**Step 4**: Under the condition \( \eta < \min \left\{ \frac{2}{8\eta L}, \frac{K + 1}{8\eta L} \right\}, \) we have \( \eta - L\Delta^2 \| x_{k-\tau_k} \| \geq \frac{1}{2} \). Then following the proof of ProxSVRG, we can get the results.

**Remark**: Theorem 4.1 actually shows that Async-ProxSVRG can achieve linear speedup when \( \Delta \) is small and \( \tau \leq \sqrt{8/B^2\Delta}. \) For sequential ProxSVRG, with stop size
\( \eta = 0.1B/L \), the inner loop size \( K \) should be in the same order of \( O(L/B\mu) \) to make \( \rho < 1 \). The computation complexity (number of gradients need to calculate) for the inner loop is in the same order of \( O(L/\mu) \). For the Async-ProxSVRG, with \( \eta = \min \{ \frac{2}{5L^2B^2\Delta^2}, \frac{0.05B}{L} \} \), the inner loop size \( K \) should be in the same order of \( O(L/B\mu + B\Delta^2 L/\mu) \) to make \( \rho < 1 \). For the case \( \tau < \sqrt{8/\Delta^2} \) (i.e., \( \frac{0.05B}{L} < \frac{2}{5L^2B^2\Delta^2} \)), by setting \( \eta = \frac{0.05B}{L} \), the order of inner loop size \( K \) is \( O(L/B\mu) \) and the corresponding computation complexity is \( O(L/\mu) \), which is the same as the sequential ProxSVRG. Therefore, Async-ProxSVRG can achieve nearly the same performance as the sequential version, but \( \tau \) times faster since we are running the algorithm asynchronously, and thus we achieve "linear speedup". For the case \( \tau \geq \sqrt{8/\Delta^2} \), the inner loop size \( K \) should be in the same order of \( O(B\Delta^2 L/\mu) \). Compared with the sequential ProxSVRG with \( K = O(L/B\mu) \), Async-ProxSVRG can not obtain linear speedup but still have a theoretical speedup of \( 1/B^2 \Delta \) if \( B^2\Delta < 1 \).

According to Theorem 4.3 and the above discussions, we provide the following corollary for a simple setup of the parameters in Async-ProxSVRG which can achieve near linear speedup.

**Corollary 4.2** Suppose Assumptions 1-4 hold. If we set \( B = (\frac{1}{3})^{\frac{1}{4}} \), \( \eta \leq \sqrt{8/\Delta^2} \), \( \eta = \frac{0.05\Delta^2}{L} \), and \( K = 2000\Delta^\frac{1}{2} / \mu \), then Async-ProxSVRG has the following linear convergence rate:

\[
\mathbb{E}P(\tilde{x}_a) - P(x^*) \leq \left( \frac{5}{6} \right)^a \left[ \mathbb{E}P(\tilde{x}_0) - P(x^*) \right],
\]

where \( x^* = \text{argmin}_x P(x) \).

**Async-ProxSVRCD**

In this section, we present Theorem 4.3 which states the convergence rate of Async-ProxSVRCD, as well as the conditions for them to achieve near linear speedup.

**Assumption 3′:** (Bounded and Independent Delay) The random delay variables \( \tau_1, \tau_2, \ldots \) in inconsistent read setting in Eqn. 3′ are independent of each other and independent of \( B_k \), and their expectations are upper bounded by \( \tau \).

**Theorem 4.3** Suppose Assumptions 1, 2, and 3′ hold. In addition, we assume that the mini-batch size \( B \geq L/\tau \), the step size \( \eta \) and the coordinate block number \( m \) satisfies \( \eta \leq \min \{ \frac{1}{3}, \frac{4m^2}{m^2 + 3m^2 \tau + \tau^2}, \frac{\mu \sqrt{m}}{2\tau} \} \), and the inner loop size \( K \) is sufficiently large so that

\[
\rho = \left( \frac{\eta K(1 - \frac{4\tau}{\sqrt{m}})}{\nu \sqrt{m}} - 4\eta T \right) + \frac{4\eta T K(1 + 1)}{(1 - \frac{4\tau}{\sqrt{m}} - 4\eta T) K} < 1,
\]

then Async-ProxSVRCD has linear convergence in expectation:

\[
\mathbb{E}P(\tilde{x}_a) - P(x^*) \leq \rho^a \left[ \mathbb{E}P(\tilde{x}_0) - P(x^*) \right],
\]

where \( x^* = \text{argmin}_x P(x) \).

**Proof Sketch of Theorem 4.3:**

**Step 1:** By the convexity of \( F(x) \) and \( R(x) \), we have

\[
P(x^*) \geq m\mathbb{E}_{j_k} P(x_{k+1}) - (m-1)P(x_k) + (\eta - \frac{T\eta^2}{2})m\mathbb{E}_{j_k} \| g_k \|^2 + (v_k - \nabla F(x_k))^T (\tilde{x}_{k+1} - x^*) + m\mathbb{E}_{j_k} (g_k)^T (x^* - x_k) + (\nabla f_{B_k}(\tilde{x}_k) - \nabla f_{B_k}(x_k))^T (\tilde{x}_{k+1} - x^*).
\]

**Step 2:** We decompose the term \((\nabla f_{B_k}(\tilde{x}_k) - \nabla f_{B_k}(x_k))^T (\tilde{x}_{k+1} - x^*)\) by using Assumption 2 as below:

\[
- \frac{1}{T}(\nabla f_{B_k}(\tilde{x}_k) - \nabla f_{B_k}(x_k))^T (\tilde{x}_{k+1} - x^*) \leq \sum_{k=0}^{m-1} \sum_{a=k+\tau}^{k-1} \| x_{a+1} - x_a \| \| \tilde{x}_{k+1} - x_k \| + \sum_{a=k+\tau}^{k-1} \| x_{a+1} - x_a \| \| x_a - x^* \|
\]

\[
+ \sum_{a=k-\tau}^{k-1} \sum_{a=k+\tau}^{k-1} \| x_{a+1} - x_a \| \| x_{b+1} - x_b \|.
\]

By taking expectation w.r.t \( j_1, \ldots, j_k \) gradually, we can bound the three terms on the right side. This is a key step for the proof and please see the details in the supplementary materials. Thus we can get

\[
\sum_{k=0}^{m-1} \mathbb{E}(g_k)^T (x_k - x^*) + A(\eta)\mathbb{E}\| g_k \|^2
\]

\[
\leq \sum_{k=0}^{m-1} (\mathbb{E}(P(x_{k+1}) - P(x^*))) - \frac{1}{m} (v_k - \nabla F(x_k))^T (\tilde{x}_{k+1} - x^*)
\]

\[
+ \sum_{k=0}^{m-1} (\frac{m-1}{m}) + \frac{T\eta T}{\mu m^2} \| \mathbb{E}(P(x_k) - P(x^*)) \|
\]

where \( A(\eta) = \left( \eta \left(1 - \frac{T\eta}{2m^2} \right) - T\eta^2 \left( \frac{1}{2} + \frac{T\eta}{2m^2} + \frac{T\eta}{m^2} \right) \right) \).

**Step 3:** With the assumption \( \eta < \frac{1}{2} \frac{m^2 - 3m^2 \tau + \tau^2}{m^2 + 3m^2 \tau + \tau^2} \), we have \( A(\eta) > \frac{m}{2} \). Then by following the proof of ProxSVRCD, we can get the results.

**Remark:** Theorem 4.3 actually shows that when \( m \) is large (equivalent the block size is small) and \( \tau \leq \min \{ \sqrt{m}, 4\rho \sqrt{m}, m^2/2T \} \), Async-ProxSVRCD can achieve linear speedup. For the sequential ProxSVRCD, Corollary 4.3 in [Zhou et al. 2014] set \( \eta = 1/16T \), \( B = L/T \) and the inner loop size \( K \) in the same order of \( O(mT/\mu) \) to make \( \rho < 1 \). For Async-ProxSVRCD, if \( m \) is sufficiently large so that the delay satisfies \( \tau \leq \min \{ \sqrt{m}, 4\rho \sqrt{m}, m^2/2T \} \), we can set \( \eta = 1/24T \) which guarantees the condition \( \rho < \min \{ \frac{1}{2} \frac{m^2 - 3m^2 \tau + \tau^2}{m^2 + 3m^2 \tau + \tau^2}, \frac{1}{4\rho \sqrt{m}}, \frac{1}{2T} \} \). Thus, the inner loop size \( K \) should be \( O(mT/\mu) \) to make \( \rho < 1 \), which is the same as sequential ProxSVRCD. Therefore, Async-ProxSVRCD can achieve near linear speedup. If we consider the indicative case [Shamir, Srebro, and Zhang 2014] in which \( L/\mu = \sqrt{n}, L = O(1) \) and \( \mu = O(\sqrt{1/n}) \). The condition for the linear speedup can be simplified to \( \tau \leq 4\sqrt{m/n} \).

Even if \( 4\sqrt{m/n} < \tau \leq \sqrt{m/n} \), Async-ProxSVRCD still...
have a speedup of $O(\sqrt{m/n})$ by setting $\eta = \frac{\mu \sqrt{m}}{2T \sqrt{\tau}} = \frac{\sqrt{m}}{2T \sqrt{\mu}}$.

Since $\frac{m \tau}{2 T \sqrt{m}} > \frac{m^2 \tau}{2 T \sqrt{\mu}}$.

According to Theorem 4.3 and the above discussions, we provide the following corollary for a simple setup of the parameters in Async-ProxSVRCD which can achieve near linear speedup.

**Corollary 4.4** Suppose Assumptions 1, 2, and 3 hold and the delay bound satisfies $\tau \leq \min \left\{ \sqrt{m}, 4\mu \sqrt{m}, \frac{m \tau}{2T} \right\}$. Let $\eta = \frac{1}{24T}$, $B = \frac{L}{T}$ and $K = \frac{16mT}{\mu}$, then Async-ProxSVRCD has the following linear convergence rate:

$$\mathbb{E}P(\tilde{x}_s) - P(x^*) \leq \left( \frac{5}{6} \right)^s [P(\tilde{x}_0) - P(x^*)],$$

where $x^* = \arg\min_x P(x)$.

By comparing the conditions of the linear speedup for asynchronous Proximal algorithms, we have the following findings: (1) Async-ProxSVRG relies on the data sparsity to alleviate the negative impact of communication delay $\tau$; (2) Async-ProxSVRCD does not rely on the sparsity condition, however, it requires the block size is small or the input dimension is large, since in this way, the block-wise updates will become frequent and can also alleviate the delay of the whole parameter vector.

To sum up, in this section, based on a few widely used assumptions, we have proven the convergence properties of the asynchronous parallel implementations of ProxSVRG, and ProxSVRCD, and discussed the conditions for them to achieve near linear speedups as compared to their sequential (single-machine) counterparts. In the next section, we will report the results of our experiments to verify these theoretical findings.

## 5 Experiments

In this section, we report our experimental results on the efficiency of the asynchronous proximal algorithms with variance reduction. In particular, we conducted binary classifications on three benchmark datasets: rcv1, real-sim, news20 (Reddi et al., 2015), new20 is the densest one with a much higher dimension and rcv1 is the sparsest one. The detailed information about the three data sets is given in Table 1. We use the logistic loss function with both $L_1$ and $L_2$ regularizations with weight $\lambda_1$ and $\lambda_2$ respectively.

| Table 1: Experimental Datasets |
|--------------------------------|
| Dataset | rcv1 | real-sim | news20 |
| Data size $n$ | 20242 | 72309 | 19996 |
| Feature size $d$ | 47236 | 20958 | 1355919 |
| $\lambda_1$, $\lambda_2$ | $10^{-5}$, $10^{-4}$ | $10^{-4}$, $10^{-4}$ | $10^{-6}$, $10^{-6}$ |

Following the practices in (Xiao and Zhang, 2014), we normalized the input vector of each data set before feeding it into the classifier, which leads to an upper bound of $0.25$ for the Lipschitz constant $L$. The stopping criterion for all the algorithms under investigation is the optimization error smaller than $10^{-10}$ (i.e., $P(\tilde{x}_S) - P(x^*) < 10^{-10}$). For Async-ProxSVRG, we set step size $\eta = 0.04$, the mini-batch size $B = 200$, and the inner loop size $K = 2n$, where $n$ is the data size. For Async-ProxSVRCD, we set step size $\eta = 0.04$, the number of block partitions $m = \frac{d}{100}$, the mini-batch size $B = 200$, and a larger inner loop size $K = 2nm$. We implement Async-ProxSVRNG and Async-ProxSVRCD in the consistent read setting and the inconsistent read setting, respectively.

The speedups of Async-ProxSVRG and Async-ProxSVRCD are shown in Figures 1(a) and 1(b). From the figures, we have the following observations. (1) On all the three datasets, Async-ProxSVRNG has near linear speedup compared to its sequential counterpart. The speedup on rcv1 is the largest, while that on news20 is the smallest. This observation is consistent with our theoretical findings that Async-ProxSVRNG has better performance on sparser data. (2) Async-ProxSVRCD also achieves nice speedup. The speedup is more significant for news20 than that for the other two data sets. This is consistent with our theoretical discussions - the sufficient condition for the linear speedup of Async-ProxSVRCD is easier to be satisfied for high-dimensional datasets. As literature also reported other asynchronous algorithms, such as Async-ProxSGD and Async-ProxSCD, we also compare with them to test the performance of our algorithms. Our algorithms actually converge faster than those without variance reduction, which means asynchronization can work together with VR techniques smoothly and enhances the model’s convergence speed. For saving space, we put the detailed results in the supplementary materials.

In summary, our experimental results well validate our theoretical findings, and indicate that the asynchronous proximal algorithms with variance reduction are very efficient and could have good applications in practice.

## 6 Conclusion

In this paper, we have studied the asynchronous parallelization of two widely used proximal gradient algorithms with variance reduction, i.e., ProxSVRG and ProxSVRCD. We have proved their convergence rates, discussed their speedups, and verified our theoretical findings through experiments. Overall speaking, these asynchronous proximal algorithms can achieve linear speedup under certain conditions, and can be highly efficient when being used to solve large scale R-
ERM problems. As for future work, we plan to make the following explorations. First, we will extend the study in this paper to the non-convex case, both theoretically and experimentally. Second, we will study the asynchronous parallelization of more proximal algorithms.

References

[Agarwal and Duchi 2011] Agarwal, A., and Duchi, J. C. 2011. Distributed delayed stochastic optimization. In NIPS, 873–881.

[Dean et al. 2012] Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Mao, M.; Senior, A.; Tucker, P.; Yang, K.; Le, Q. V.; et al. 2012. Large scale distributed deep networks. In NIPS, 1223–1231.

[Feyzmahdavian, Aytekin, and Johansson 2015] Feyzmahdavian, H. R.; Aytekin, A.; and Johansson, M. 2015. An asynchronous mini-batch algorithm for regularized stochastic optimization. arXiv preprint arXiv:1505.04824.

[Johnson and Zhang 2013] Johnson, R., and Zhang, T. 2013. Accelerating stochastic gradient descent using predictive variance reduction. In NIPS, 315–323.

[Langford, Li, and Zhang 2009] Langford, J.; Li, L.; and Zhang, T. 2009. Sparse online learning via truncated gradient. In NIPS, 905–912.

[Lian et al. 2015] Lian, X.; Huang, Y.; Li, Y.; and Liu, J. 2015. Asynchronous parallel stochastic gradient for nonconvex optimization. In NIPS, 2719–2727.

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

[Liu et al. 2013] Liu, J.; Wright, S. J.; Ré, C.; Bittorf, V.; and Sridhar, S. 2013. An asynchronous parallel stochastic coordinate descent algorithm. arXiv preprint arXiv:1311.1873.

[Liu, Wright, and Sridhar 2014] Liu, J.; Wright, S. J.; and Sridhar, S. 2014. An asynchronous parallel randomized kaczmarz algorithm. arXiv preprint arXiv:1401.4780.

[Mania et al. 2015] Mania, H.; Pan, X.; Papailiopoulos, D.; Recht, B.; Ramchandran, K.; and Jordan, M. I. 2015. Perturbed iterate analysis for asynchronous stochastic optimization. arXiv preprint arXiv:1507.06970.

[Rakhlin, Shamir, and Sridharan 2011] Rakhlin, A.; Shamir, O.; and Sridharan, K. 2011. Making gradient descent optimal for strongly convex stochastic optimization. arXiv preprint arXiv:1109.5647.

[Recht et al. 2011] Recht, B.; Re, C.; Wright, S.; and Niu, F. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In NIPS, 693–701.

[Reddi et al. 2015] Reddi, S. J.; Hefny, A.; Sra, S.; Póczos, B.; and Smola, A. J. 2015. On variance reduction in stochastic gradient descent and its asynchronous variants. In NIPS, 2629–2637.

[Shalev-Shwartz and Tewari 2011] Shalev-Shwartz, S., and Tewari, A. 2011. Stochastic methods for 1 1-regularized loss minimization. The Journal of Machine Learning Research 12:1865–1892.

[Shamir, Srebro, and Zhang 2014] Shamir, O.; Srebro, N.; and Zhang, T. 2014. Communication-efficient distributed optimization using an approximate newton-type method. In Proceedings of the 31st International Conference on Machine Learning (ICML-14), 1000–1008.

[Tran et al. 2015] Tran, K.; Hosseini, S.; Xiao, L.; Finley, T.; and Bilenko, M. 2015. Scaling up stochastic dual coordinate ascent. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1185–1194. ACM.

[Wright 2015] Wright, S. J. 2015. Coordinate descent algorithms. Mathematical Programming 151(1):3–34.

[Xiao and Zhang 2014] Xiao, L., and Zhang, T. 2014. A proximal stochastic gradient method with progressive variance reduction. SIAM Journal on Optimization 24(4):2057–2075.

[Zhao et al. 2014] Zhao, T.; Yu, M.; Wang, Y.; Arora, R.; and Liu, H. 2014. Accelerated mini-batch randomized block coordinate descent method. In NIPS, 3329–3337.
7 Appendices

Proof of Theorem 4.1

Firstly we introduce some notations. Let $x_{k+1} - x_k = -\eta g_k, v_k = \nabla f_{B_k}(x_k) - \nabla f_{B_k}(\tilde{x}) + \nabla F(\tilde{x}),$ and $u_k = \nabla f_{B_k}(x_k - r_k) - \nabla f_{B_k}(\tilde{x}) + \nabla F(\tilde{x}).$

Since the update formula is

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left\{ \frac{1}{2} \|x - x_k - \eta u_k\|^2 + \eta R(x) \right\},$$

the associated optimality condition states that there is a $\xi_{k+1} \in \partial R(x_{k+1})$ such that $x_{k+1} - (x_k - \eta u_k) + \eta \xi_{k+1} = 0.$ By the smoothness assumption of $f_i(x)$ and the sparseness assumption 4, we have: $\forall x, y \in \mathbb{R}^d$ which are independent with $B_k,$

$$F(x) = E_{B_k} f_{B_k}(x)$$

$$\geq E_{B_k} f_{B_k}(y) - E_{B_k} \nabla f_{B_k}(x)(y - x) - \frac{L}{2} E_{B_k} \|x - y\|_{B_k}^2$$

$$\geq F(y) - \nabla F(x)(y - x) - \frac{L B \Delta}{2} \|x - y\|^2,$$ (10)

where Ineq. (10) is established by $E_{B_k} \|x\|^2_{B_k} \leq B \Delta \|x\|^2,$ which comes from $E_k \|x\|_{g_k}^2 \leq \Delta \|x\|^2.$

By the convexity of $F(x)$ and $R(x),$ we have:

$$P(x^*) = F(x^*) + R(x^*)$$

$$\geq F(x_{k+1} - r_k) + \nabla F(x_{k+1} - r_k)^T (x^* - x_{k+1} - r_k) + R(x_{k+1}) + \xi_{k+1}^T (x^* - x_{k+1})$$

$$\geq F(x_{k+1}) - \nabla F(x_{k+1} - r_k)^T (x_{k+1} - x_{k-1}) - \frac{L B \Delta}{2} \|x_{k+1} - x_{k-1}\|^2 + \nabla F(x_{k+1} - r_k)^T (x^* - x_{k+1})$$

$$+ R(x_{k+1}) + \xi_{k+1}^T (x^* - x_{k+1})$$

$$= P(x_{k+1}) + (u_k - \nabla F(x_{k+1} - r_k))^T (x_{k+1} - x^*) + g_k^T (x^* - x_{k+1}) + \eta \|g_k\|^2 - \frac{L B \Delta}{2} \|x_{k+1} - x_{k-1}\|^2$$

$$= P(x_{k+1}) + (u_k - \nabla F(x_{k+1} - r_k))^T (x_{k+1} - x^*) + \eta \|g_k\|^2 - \frac{L \eta^2 B \Delta r_k}{2} \sum_{h=k-r_i}^k \|g_h\|^2 + g_k^T (x^* - x_{k+1}),$$ (11)

where the second inequality is established by Ineq. (10).

By rearranging Ineq. (11), we have:

$$- g_k^T (x_{k+1} - x^*) + (\eta - \frac{L \eta^2 B \Delta r_k}{2}) \|g_k\|^2 - \frac{L \eta^2 B \Delta r_k}{2} \sum_{h=k-r_i}^k \|g_h\|^2$$

$$\leq P(x^*) - P(x_{k+1}) - (u_k - \nabla F(x_{k+1} - r_k))^T (x_{k+1} - x^*).$$ (12)

According to the proof of Lemma 3 in [Xiao and Zhang 2014], we can get:

$$- E_{B_k} (u_k - \nabla F(x_{k+1} - r_k))^T (x_{k+1} - x^*)$$

$$= \eta E_{B_k} \|u_k - \nabla F(x_{k+1} - r_k)\|^2$$

$$\leq 2 \eta E_{B_k} \|u_k - \nabla F(x_{k+1} - r_k)\| \|u_k - \nabla F(x_{k+1})\|^2 + 2 \eta E_{B_k} \|u_k - \nabla F(x_{k+1})\|^2$$

$$= 2 \eta E_{B_k} \|u_k - \nabla F(x_{k+1} - r_k)\|^2 - 2 \eta \|\nabla F(x_{k+1} - r_k) - \nabla F(x_{k+1})\|^2 + 2 \eta E_{B_k} \|u_k - \nabla F(x_{k+1})\|^2$$

$$\leq 2 \eta E_{B_k} \|u_k - \nabla F(x_{k+1} - r_k)\|^2 + 2 \eta E_{B_k} \|u_k - \nabla F(x_{k+1})\|^2$$

$$\leq 2 \eta L^2 E_{B_k} \|x_{k+1} - x_k\|^2 + 2 \eta E_{B_k} \|u_k - \nabla F(x_{k+1})\|^2$$

$$\leq 2 \eta \sum_{h=k-r_i}^{k-1} \|g_h\|^2 + 2 \eta E_{B_k} \|u_k - \nabla F(x_{k+1})\|^2$$

$$\leq 2 \eta \sum_{h=k-r_i}^{k-1} \|g_h\|^2 + 2 \eta E_{B_k} \|u_k - \nabla F(x_{k+1})\|^2$$

$$\leq 2 \eta L^2 B \Delta r_k \sum_{h=k-r_i}^{k-1} \|g_h\|^2 + 2 \eta E_{B_k} \|u_k - \nabla F(x_{k+1})\|^2$$

$$\leq 2 \eta L^2 B \Delta r_k \sum_{h=k-r_i}^{k-1} \|g_h\|^2 + \frac{8 \eta L B}{P(x_k) - P(x^*) + P(\tilde{x}) - P(x^*).}$$

The $^{(1)}$ is established based on Corollary 3 in [Xiao and Zhang 2014].
Then by taking expectation on both sides of Ineq. (13) with respect to $B_k$ and $\tau_k$, and by using Assumption 3, we obtain:

$$
- \mathbb{E}_B g_k^T (x_{k+1} - x^*) + (\eta - L\eta^2 B \Delta \tau^2) \mathbb{E}_B \| g_k \|^2 - \left( \frac{L\eta^2 B \Delta \tau}{2} + 2\eta^3 B^2 \Delta \tau \right) \sum_{k=0}^{K-1} \mathbb{E}_B \| g_k \|^2 \\
\leq (P(x^*) - \mathbb{E}_B P(x_{k+1})) + \frac{8L\eta}{B} \left( \sum_{k=0}^{K-1} (P(x_k) - P(x^*) + \mathbb{E}(P(x_k) - P(x^*)) \right)
$$

(13)

Summing both sides of Ineq. (13) from $k = 0$ to $K - 1$, and taking expectations with respect to $B_{k-1}, \ldots, B_1$ gradually, we can get:

$$
- \sum_{k=0}^{K-1} \mathbb{E}_B g_k^T (x_{k+1} - x^*) + (\eta - L\eta^2 B \Delta \tau^2 (1 + 2L\eta)) \sum_{k=0}^{K-1} \mathbb{E}_B \| g_k \|^2 \\
\leq \left( \frac{8L\eta}{B} - 1 \right) \sum_{k=0}^{K-1} (P(x_{k+1}) - P(x^*)) + \frac{8L\eta}{B} (K+1)(P(\hat{x}) - P(x^*)).
$$

By rearranging the above inequality, we can get:

$$
- \sum_{k=1}^{K-1} \mathbb{E}_B g_k^T (x_k - x^*) + (\eta - L\eta^2 B \Delta \tau^2 (1 + 2L\eta)) \sum_{k=0}^{K-1} \mathbb{E}_B \| g_k \|^2 \\
\leq \left( \frac{8L\eta}{B} - 1 \right) \sum_{k=0}^{K-1} (P(x_{k+1}) - P(x^*)) + \frac{8L\eta}{B} (K+1)(P(\hat{x}) - P(x^*)).
$$

Under the condition $\eta < \min \left\{ \frac{2}{5LB \Delta \tau^2}, \frac{B}{16L} \right\}$, we have $\eta - L\eta^2 B \Delta \tau^2 (1 + 2L\eta) \geq \frac{\eta}{2}$. Then we can get:

$$
\mathbb{E}\|x_K - x^*\|^2 = \mathbb{E}\|x_K - x^*\|^2 - 2\eta \mathbb{E}g_{K-1}^T (x_K - x^*) + \eta^2 \mathbb{E}\|g_{K-1}\|^2 \\
\leq \mathbb{E}\|\hat{x} - x^*\|^2 - 2\eta \sum_{k=0}^{K-1} \mathbb{E}g_k^T (x_k - x^*) + \eta^2 \sum_{k=0}^{K-1} \mathbb{E}\|g_k\|^2 \\
\leq \mathbb{E}\|\hat{x} - x^*\|^2 + 2\eta \frac{8L\eta}{B} (K+1)(P(\hat{x}) - P(x^*)) \\
\leq \frac{2}{\mu} \mathbb{E}(P(\hat{x}) - P(x^*)) + 2\eta \frac{8L\eta}{B} (K+1)(P(\hat{x}) - P(x^*))
$$

(14)

where the last inequality follows by the strongly convexity assumption.

By rearranging the Ineq. (13), we get:

$$
2\eta (1 - \frac{8L\eta}{B}) \sum_{k=1}^{K} \mathbb{E}(P(x_k) - P(x^*)) \leq \frac{2}{\mu} \left( \frac{16L^2}{B} \right) (P(\hat{x}) - P(x^*)). 
$$

(15)

Dividing both sides of Ineq. (15) by $2\eta (1 - \frac{8L\eta}{B})$, we obtain:

$$
P(\hat{x}_k) - P(x^*) \leq \left( \frac{B}{\eta \mu K(B - 8\eta L)} + \frac{8\eta L}{(B - 8\eta L)} \right) \mathbb{E}[P(\hat{x}_{k+1}) - P(x^*)].
$$

Proof of Theorem 4.3

Let $x_{k+1} - x_k = -\eta g_k$, $v_k = \nabla f_{B_k}(x_k) - \nabla f_{B_k}(\hat{x}) + \nabla F(\hat{x})$, and $u_k = \nabla f_{B_k}(\hat{x}_k) - \nabla f_{B_k}(\hat{x}) + \nabla F(\hat{x})$. Let $\tilde{x}_{k+1} = \text{argmin}_{x \in \mathcal{R}} \left\{ \frac{1}{2} \| x - x_k - \eta u_k \|^2 + \eta R(x) \right\}$, and recall the following update rule for $x_k$:

$$
\begin{align*}
x_{k+1} & = \mu x_k + (1 - \mu) \mathbb{E} x_{k+1} \quad \text{if} \quad x_{k+1} \in C_k, \\
x_{k+1} & = x_k \quad \text{else}.
\end{align*}
$$

(15)
We take expectation with respect to $j_k$, and have,
\begin{align}
E_{j_k}(x_{k+1} - x_k) &= \frac{1}{m}(\bar{x}_{k+1} - x_k) \\
E_{j_k}\|x_{k+1} - x_k\|^2 &= \frac{1}{m}\|\bar{x}_{k+1} - x_k\|^2. 
\end{align}

We have the following derivation for $P(x^*)$,
\begin{align}
P(x^*) &= F(x^*) + R(x^*) \\
&\geq F(x_k) + \nabla F(x_k)^T(x^* - x_k) + R(\bar{x}_{k+1}) + \xi_{k+1}^T(x^* - \bar{x}_{k+1}) \\
&= mE_{j_k}P(x_{k+1}) - (m - 1)P(x_k) + (\eta - \frac{T\eta^2}{2})mE_{j_k}\|g_k\|^2 + (v_k - \nabla F(x_k))^T(\bar{x}_{k+1} - x_k) \\
&\quad + mE_{j_k}(g_k)^T(x^* - x_k) + (\nabla f_{B_k}(\hat{x}_k) - \nabla f_{B_k}(x_k))^T(\bar{x}_{k+1} - x^*).
\end{align}

The first "$\geq$" holds by the convexity of $F(x)$ and $R(x)$. The last "$=\" holds, by lemma B.1 in (Zhao et al. 2014).
Due to the delay, the Ineq. (18) has an extra term $(\nabla f_{B_k}(\hat{x}_k) - \nabla f_{B_k}(x_k))^T(\bar{x}_{k+1} - x^*)$ compared to lemma B.1 in (Zhao et al. 2014). Next we will show how to bound this term by using the separability of the coordinate blocks. Intuitively, each worker calculates a partial gradient at each iteration. When the the number of block partitions $m$ is sufficient large, different workers select the same block with low probability.
We decompose the term $-(\nabla f_{B_k}(\hat{x}_k) - \nabla f_{B_k}(x_k))^T(\bar{x}_{k+1} - x^*)$ by using the partial smoothness assumption and the bounded and independent delay assumption as below:
\begin{align}
&- (\nabla f_{B_k}(\hat{x}_k) - \nabla f_{B_k}(x_k))^T(\bar{x}_{k+1} - x^*) \\
&\leq \|\nabla f_{B_k}(\hat{x}_k) - \nabla f_{B_k}(x_k)\|\|\bar{x}_{k+1} - x^*\| \\
&\leq T\|\bar{x}_k - x_k\|\|\bar{x}_{k+1} - x^*\| \\
&\leq T\sum_{h \in J(k)}\|x_{h+1} - x_k\|\|\bar{x}_{k+1} - x^*\| \\
&\leq T\left\{\sum_{a=k-\tau}^{k-1}\|x_{a+1} - x_a\|\|\bar{x}_{k+1} - x_k\| + \sum_{a=k-\tau}^{k-1}\sum_{b=a}^{k-1}\|x_{a+1} - x_a\|\|x_{b+1} - x_b\| + \sum_{a=k-\tau}^{k-1}\|x_{a+1} - x_a\|\|x_a - x^*\|\right\}.
\end{align}

For the term (i), we have
\begin{align}
\sum_{a=k-\tau}^{k-1}\|x_{a+1} - x_a\|\|\bar{x}_{k+1} - x_k\| &\leq \sum_{a=k-\tau}^{k-1}\frac{1}{2}(\sqrt{m}\|x_{a+1} - x_a\|^2 + \frac{1}{\sqrt{m}}\|\bar{x}_{k+1} - x_k\|^2) \\
&\leq \sum_{a=k-\tau}^{k-1}\frac{1}{2}(\sqrt{m}\|x_{a+1} - x_a\|^2 + \sqrt{m}E_{j_k}\|x_{k+1} - x_k\|^2) \\
&\leq \frac{\eta^2\sqrt{m}}{2}\sum_{a=k-\tau}^{k-1}\|g_a\|^2 + \frac{\eta^2\tau\sqrt{m}}{2}E_{j_k}\|g_k\|^2. 
\end{align}

The first "$\leq$" holds by the AM-GM inequality. The second "$\leq$" holds by Eqn (17).
For the term (ii), we have
\begin{align}
\sum_{a=k-\tau}^{k-1}\sum_{b=a}^{k-1}\|x_{a+1} - x_a\|\|x_{b+1} - x_b\| &\leq \sum_{a=k-\tau}^{k-1}\eta^2\left(\|g_a\|^2 + \sum_{b=a+1}^{k-1}\|g_{j_b}\|\|g_b\|\right) \\
&\leq \sum_{a=k-\tau}^{k-1}\frac{\eta^2}\|g_a\|^2 + \sum_{a=k-\tau}^{k-1}\eta^2\sum_{b=a+1}^{k-1}\left(\frac{\sqrt{m}\|g_{j_b}\|^2}{2} + \frac{\|g_b\|^2}{2\sqrt{m}}\right). 
\end{align}
It is clear that, \( E_{j_k} \|g_a\|^2 = \frac{1}{m} \|g_a\|^2 \) for \( b > a \) since \( j_b \) is independent to \( g_a \). Therefore, the expectation of the Ineq. (20) inequality, we have the following derivation:

\[
E \left( \sum_{k=k-\tau}^{k-1} \sum_{b=a}^{k-1} \|x_{a+1} - x_a\| \|x_{b+1} - x_b\| \right) \leq E \left( \sum_{a=k-\tau}^{k-1} \eta^2 \|g_a\|^2 \right) + E \left( \sum_{a=k-\tau}^{k-1} \eta^2 \sum_{b=a+1}^{k-1} \left( \frac{\sqrt{m}E_{j_b} \|g_a\|^2}{2} + \|g_b\|^2 \frac{2}{2\sqrt{m}} \right) \right)
\]

\[
= E \left( \sum_{a=k-\tau}^{k-1} \eta^2 \|g_a\|^2 \right) + E \left( \sum_{a=k-\tau}^{k-1} \eta^2 \sum_{b=a+1}^{k-1} \left( \|g_a\|^2 \frac{2}{2\sqrt{m}} + \|g_b\|^2 \frac{2}{2\sqrt{m}} \right) \right)
\]

\[
= E \sum_{a=k-\tau}^{k-1} \eta^2 (\frac{\tau}{2\sqrt{m}} + 1) \|g_a\|^2 \tag{21}
\]

For the term (iii), we have

\[
\sum_{a=k-\tau}^{k-1} \|x_{a+1} - x_a\| \|x_a - x^*\| \leq \sum_{a=k-\tau}^{k-1} T \|x_{a+1} - x_a\| \|x_a - x^*\|_{\eta_a} \leq \sum_{a=k-\tau}^{k-1} \left( \frac{T \eta}{2\sqrt{m}} \|g_a\|^2 + \frac{T \eta \sqrt{m}}{2} \|x_{a-1} - x^*\|_{\eta_a}^2 \right). \tag{22}
\]

Taking expectations on both size of Ineq. (22), we can get:

\[
E \left( \sum_{a=k-\tau}^{k-1} \|x_{a+1} - x_a\| \|x_a - x^*\| \right) \leq E \sum_{a=k-\tau}^{k-1} \left( \frac{T \eta}{2\sqrt{m}} \|g_a\|^2 + \frac{T \eta \sqrt{m}}{2} \|x_{a-1} - x^*\|_{\eta_a}^2 \right)
\]

\[
= E \sum_{a=k-\tau}^{k-1} \left( \frac{T \eta}{2\sqrt{m}} \|g_a\|^2 + \frac{T \eta \sqrt{m}}{2} \|x_a - x^*\|^2 \right) \tag{24}
\]

Summing up Ineq. (19), (21) and (24), we can get

\[
- (\nabla f_{B_k}(\hat{x}_k) - \nabla f_{B_k}(x_k))^T (\hat{x}_{k+1} - x^*) \leq \frac{T \eta^2 \sqrt{m}}{2} E \|g_a\|^2 + \frac{T \eta}{2\sqrt{m}} \sum_{a=k-\tau}^{k-1} E \|x_a - x^*\|^2 + \frac{T \eta}{2\sqrt{m}} \left( \left( \frac{T \eta}{2\sqrt{m}} + T \eta^2 \left( \frac{\tau}{2\sqrt{m}} + 1 + \sqrt{m} \right) \right) E \|g_a\|^2 \right). \tag{25}
\]

We have finished bounding the term \(- (\nabla f_{B_k}(\hat{x}_k) - \nabla f_{B_k}(x_k))^T (\hat{x}_{k+1} - x^*)\). Taking expectation on both sides of Ineq. (18) and putting Ineq. (25) in Ineq. (18), we can get

\[
- E(g_k)^T (x_k - x^*) + (\eta - T \frac{T \eta^2}{2} - T \frac{T \eta^2 \tau}{2\sqrt{m}}) E \|g_k\|^2 \leq - (\text{EP}(x_{k+1}) - P(x^*)) + \frac{(m-1)}{m} E (P(x_k) - P(x^*)) - \frac{1}{m} (v_k - \nabla F(x_k))^T (\hat{x}_{k+1} - x^*) + \frac{T \eta}{2m} \sum_{a=k-\tau}^{k-1} E \|x_a - x^*\|^2 + \frac{T \eta}{2\sqrt{m}} \left( \left( \frac{T \eta}{2\sqrt{m}} + T \eta^2 \left( \frac{\tau}{2\sqrt{m}} + 1 + \sqrt{m} \right) \right) E \|g_a\|^2 \right). \tag{26}
\]
We conduct experiments for comparing Async-ProxSVRG and Async-ProxSVRCD with other asynchronous proximal algorithms: Stochastic Proximal Optimization Algorithms with Variance Reduction”. We use a decreasing step size for ProxSGD with \( \frac{\eta}{\mu m^2} \) and set the step size for ProxSCD with \( \frac{\eta L}{mB} \). We set the step size for ProxCD with \( \frac{\eta}{\mu m^2} \) which satisfies \( \tau = O(\frac{L}{\mu m^2}) \) and choosing \( B \) which satisfies \( B > \frac{L}{T} \), we can obtain

\[
P(\bar{x}) - P(x^*) \leq \frac{4\eta_0 T(K + 1)}{\mu m^2} \frac{L}{d} + \frac{4\sigma_0 T(K + 1)}{(1 - \tau T) \mu m^2} \frac{L}{d} \frac{\eta_0}{d}.
\]

Additional Experiments

We conduct experiments for comparing Async-ProxSVRG and Async-ProxSVRCD with other asynchronous proximal algorithms: Async-ProxSGD and Async-ProxSVRCD. For all the experiments, we set the number of local workers \( P = 10 \). The parameter settings for Async-ProxSVRG and Async-ProxSVRCD are the same as the settings in section 5 in paper ”Asynchronous Stochastic Proximal Optimization Algorithms with Variance Reduction”. We use a decreasing step size for ProxSGD with \( \eta = \eta_0 \frac{2\sigma_0}{\gamma_0} \) (Reddi et al. 2015), where constant \( \eta_0 \) and \( \sigma_0 \) specify the scale and speed of decay. Since \( L \) has an upper bound of 0.25. We set the step size for ProxSCD with \( \eta = \frac{1}{\mu} \approx \gamma \sqrt{d} \) and we choose \( \gamma = 0.4 \).
The results are showed in Figure 2. Figure 2(a), 2(c) and 2(e) show the comparison between Async-ProxSVRG and Async-ProxSGD on different data sets. The results show that Async-ProxSVRG outperforms Async-ProxSGD on all the three data sets. Figure 2(b), 2(d) and 2(f) show the comparison between Async-ProxSVRCD and Async-ProxSGD and Async-ProxSCD. The results show that Async-ProxSVRG outperforms other algorithms on all the three data sets. It means that our proposed algorithms are efficient.