OPT2022: 14th Annual Workshop on Optimization for Machine Learning

Accelerating Perturbed Stochastic Iterates in Asynchronous Lock-Free Optimization

Kaiwen Zhou KWZHOU@CSE.CUHK.EDU.HK
Department of Computer Science and Engineering, The Chinese University of Hong Kong

Anthony Man-Cho So MANCHOSO@SE.CUHK.EDU.HK
Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong

James Cheng JCHENG@CSE.CUHK.EDU.HK
Department of Computer Science and Engineering, The Chinese University of Hong Kong

Abstract

We show that stochastic acceleration can be achieved under the perturbed iterate framework [25] in asynchronous lock-free optimization, which leads to the optimal incremental gradient complexity for finite-sum objectives. We prove that our new accelerated method requires the same linear speed-up condition as existing non-accelerated methods. Our key algorithmic discovery is a new accelerated SVRG variant with sparse updates. Empirical results are presented to verify our theoretical findings.

1. Introduction

We consider the following unconstrained optimization problem, which appears frequently in machine learning and statistics, such as empirical risk minimization:

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

where each f_i is L-smooth and convex, f is μ-strongly convex.[1] We further assume that the computation of each f_i is *sparse*, i.e., the computation is supported on a set of coordinates $T_i \subseteq \{1,\ldots,d\}$. For example, generalized linear models (GLMs) satisfy this assumption. GLMs has the form: $\forall i, f_i(x) = \ell_i(\langle a_i, x\rangle)$, where ℓ_i is some loss function and $a_i \in \mathbb{R}^d$ is a data sample. In this case, the support of each f_i is the set of non-zero coordinates of a_i. Many large real-world datasets have very high sparsity (see Table 1 for several examples).

In the past decade, exciting progress has been made on solving problem (1) with the introduction of stochastic variance reduced methods, such as SAG [34, 36], SVRG [13, 44], SAGA [5], S2GD [18], SARAH [30], etc. These methods achieve fast linear convergence rate as gradient descent (GD), while having low iteration cost as stochastic gradient descent (SGD). There are also several recent works on further improving the practical performance of these methods based on past gradient information [7, 12, 37, 38, 41]. Inspired by Nesterov's seminal works [26, 27, 29], stochastic accelerated methods have been proposed in recent years [1, 19, 23, 39, 47–49]. These methods achieve the optimal convergence rates that match the lower complexity bounds established in [43].

1. In fact, we will only use a weaker quadratic growth assumption. The formal definitions are in Section 2.

For sparse problems, lagged update [36] is a common technique for the above methods to efficiently leverage the sparsity.

Inspired by the emerging parallel computing architectures such as multi-core computer and distributed system, many parallel variants of the aforementioned methods have been proposed, and there is a vast literature on those attempts. Among them, asynchronous lock-free algorithms are of special interest. Recht et al. [33] proposed an asynchronous lock-free variant of SGD called Hogwild! and first proved that it can achieve a *linear speed-up*, i.e., running Hogwild! with ρ parallel processes only requires $O(1/\rho)$-times the running time of the serial variant. The condition on the maximum overlaps among the parallel processes is called the *linear speed-up condition*. Following Recht et al. [33], Lian et al. [24], Sa et al. [35] analyzed asynchronous SGD for non-convex problems, and Duchi et al. [8] analyzed for stochastic optimization; Mania et al. [25] refined the analysis framework (called the perturbed iterate framework) and proposed KroMagnon (asynchronous SVRG); Leblond et al. [20] simplified the perturbed iterate analysis and proposed ASAGA (asynchronous SAGA), and in an extended version of this work, Leblond et al. [21] further improved the analysis of KroMagnon and Hogwild!; Pedregosa et al. [32] derived proximal variant of ASAGA; Nguyen et al. [31] refined the analysis of Hogwild!; Joulani et al. [14] proposed asynchronous methods in online and stochastic settings; Gu et al. [10] proposed several asynchronous variance reduced algorithms for non-smooth or non-convex problems; Stich et al. [40] studied a broad variety of SGD variants and improved the speed-up condition of asynchronous SGD.

All the above mentioned asynchronous methods are based on non-accelerated schemes such as SGD, SVRG and SAGA. Having witnessed the success of recently proposed stochastic accelerated methods, it is natural to ask: *Is it possible to achieve stochastic acceleration in asynchronous lock-free optimization? Will it lead to a worse linear speed-up condition?* The second question comes from a common perception that accelerated methods are less tolerant to gradient noise, e.g., [4, 6].

We answer these two questions in this work: We propose the first asynchronous lock-free stochastic accelerated method for solving problem (1), and we prove that it requires the identical linear speed-up condition as the non-accelerated counterparts.

The works that are most related to ours are [9, 11, 45, 47]. Fang et al. [9] proposed several asynchronous accelerated schemes. However, their analysis requires consistent read and does not consider sparsity. The dependence on the number of overlaps τ in their complexities is $O(\tau)$ compared with $O(\sqrt{\tau})$ in our result. Zhou et al. [47] naively extended their proposed accelerated method into asynchronous lock-free setting. However, even in the serial case, their result imposes strong assumptions on the sparsity. This issue is discussed in detail in Section 3.1. In the asynchronous case, no theoretical acceleration is achieved in [47]. Hannah et al. [11], Xiao et al. [45] proposed asynchronous accelerated block-coordinate descent methods (BCD). However, as pointed out in Appendix F in [32], BCD methods perform poorly in sparse problems since they focus only on a single coordinate of the gradient information.

2. Preliminaries

Notations We use $\langle \cdot, \cdot \rangle$ and $\|\cdot\|$ to denote the standard inner product and the Euclidean norm, respectively. We let $[n]$ denote the set $\{1, 2, \ldots, n\}$, $\mathbb{E}$ denote the total expectation and $\mathbb{E}_i$ denote the expectation with respect to a random sample i. We use $[x]_v$ to denote the coordinate v of the vector $x \in \mathbb{R}^d$, and for a subset $T \subseteq [d]$, $[x]_T$ denotes the collection of $[x]_v$ for $v \in T$. We let $\lceil \cdot \rceil$ denote the ceiling function, I_d denote the identity matrix and $(\alpha)_+ \triangleq \max\{\alpha, 0\}$.

Algorithm 1 SS-Acc-SVRG: Serial Sparse Accelerated SVRG

```
Input: initial guess $x_0 \in \mathbb{R}^d$, constant $\omega > 1$ which controls the restart frequency.
Initialize: set the scalars $m, \vartheta, \varphi, \eta, S, R$ according to Theorem 1, initialize the diagonal matrix $D$.
1: for $r = 0, \ldots, R-1$ do                                        ▷ performing restarts
2:     $\tilde{x}_0 = z_0^0 = x_r$.
3:     for $s = 0, \ldots, S-1$ do
4:         Compute and store $\nabla f(\tilde{x}_s)$.
5:         for $k = 0, \ldots, m-1$ do
6:             Sample $i_k$ uniformly in $[n]$ and let $T_{i_k}$ be the support of $f_{i_k}$.
7:             $[y_k]_{T_{i_k}} = \vartheta \cdot [z_k^s]_{T_{i_k}} + (1-\vartheta) \cdot [\tilde{x}_s]_{T_{i_k}} - \varphi \cdot [D\nabla f(\tilde{x}_s)]_{T_{i_k}}$.    ▷ sparse coupling
8:             $[z_{k+1}^s]_{T_{i_k}} = [z_k^s]_{T_{i_k}} - \eta \cdot \big(\nabla f_{i_k}([y_k]_{T_{i_k}}) - \nabla f_{i_k}([\tilde{x}_s]_{T_{i_k}}) + D_{i_k}\nabla f(\tilde{x}_s)\big)$.
9:         end for
10:        $\tilde{x}_{s+1} = y_t$ for uniformly random $t \in \{0, 1, \ldots, m-1\}$.
11:        $z_0^{s+1} = z_m^s$.
12:    end for
13:    $x_{r+1} = \frac{1}{S}\sum_{s=0}^{S-1} \tilde{x}_{s+1}$.
14: end for
Output: $x_R$.
```

We say that a function $f : \mathbb{R}^d \to \mathbb{R}$ is *L-smooth* if it has L-Lipschitz continuous gradients, i.e., $\forall x, y \in \mathbb{R}^d$, $\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|$. An important consequence of f being L-smooth and convex is that $\forall x, y \in \mathbb{R}^d$, $f(x) - f(y) - \langle \nabla f(y), x - y\rangle \geq \frac{1}{2L} \|\nabla f(x) - \nabla f(y)\|^2$. We call it the *interpolation condition* following [42]. A continuously differentiable f is called *μ-strongly convex* if $\forall x, y \in \mathbb{R}^d$, $f(x) - f(y) - \langle \nabla f(y), x - y\rangle \geq \frac{\mu}{2} \|x - y\|^2$. In particular, the strong convexity at any $x \in \mathbb{R}^d$ and $x^\star$ is called the *quadratic growth*, i.e., $f(x) - f(x^\star) \geq \frac{\mu}{2} \|x - x^\star\|^2$. All our results hold under this weaker condition. We define $\kappa \triangleq \frac{L}{\mu}$, which is always called the condition ratio. Other equivalent definitions of these assumptions can be found in [16, 29]. Oracle complexity refers to the number of incremental gradient computations needed to find an ϵ-accurate solution (in expectation), i.e., $\mathbb{E}\left[f(x)\right] - f(x^\star) \leq \epsilon$.

3. Serial Sparse Accelerated SVRG

We first introduce a new accelerated SVRG variant with sparse updates in the serial case (Algorithm 1), which serves as the base algorithm for our asynchronous scheme. Our technique is built upon the following sparse approximated SVRG gradient estimator proposed in [25]: For uniformly random $i \in [n]$ and $y, \tilde{x} \in \mathbb{R}^d$, define

$$\mathcal{G}_y \triangleq \nabla f_i(y) - \nabla f_i(\tilde{x}) + D_i \nabla f(\tilde{x}), \tag{2}$$

where $D_i \triangleq P_i D$ with $P_i \in \mathbb{R}^{d\times d}$ being the diagonal projection matrix of T_i (the support of f_i) and $D = (\frac{1}{n}\sum_{i=1}^n P_i)^{-1}$, which can be computed and stored in the first pass. Note that we assume the coordinates with zero cardinality have been removed in the objective function,[2] and thus $nI_d \succeq D \succeq I_d$. A diagonal element of D corresponds to the inverse probability of the coordinate

2. Clearly, the objective value is not supported on those coordinates.

belonging to a uniformly sampled support T_i. It is easy to verify that this construction ensures the unbiasedness $\mathbb{E}_i\big[D_i \nabla f(\tilde{x})\big] = \nabla f(\tilde{x})$. We first summarize our key technical novelty (i.e., Sparse Variance Correction) and then present the convergence result.

3.1. Sparse Variance Correction

Intuitively, the sparse SVRG estimator (2) will lead to a larger variance compared with the dense one (when $D = I_d$). In the previous attempt, Zhou et al. [47] naively extends an accelerated SVRG variant into the sparse setting, which results in a fairly strong restriction on the sparsity as admitted by the authors.[3] In contrast, sparse variants of SVRG and SAGA in [20, 25] require no assumption on the sparsity, and they achieve the same oracle complexities as their original dense versions.

Analytically speaking, the only difference of adopting the sparse estimator (2) is on the variance bound. The analysis of non-accelerated dense SVRG typically uses the following variance bound ($D = I_d$) [13, 44]: $\mathbb{E}\left[\|\mathcal{G}_y - \nabla f(y)\|^2\right] \le 4L\big(f(y) - f(x^\star) + f(\tilde{x}) - f(x^\star)\big)$. It is shown in [25] that the sparse estimator (2) admits the same variance bound for any D. Thus, the analysis in the dense case can be directly applied to the sparse variant, which leads to a convergence guarantee that is independent of the sparsity.

However, things are not as smooth in the accelerated case. To (directly) accelerate SVRG, we typically uses a much tighter bound [1]: $\mathbb{E}\left[\|\mathcal{G}_y - \nabla f(y)\|^2\right] \le 2L\big(f(\tilde{x}) - f(y) - \langle \nabla f(y), \tilde{x} - y\rangle\big)$ (in the dense case $D = I_d$). Unfortunately, in the sparse case ($D \neq I_d$), we do not have an identical variance bound as before. The variance of the sparse estimator (2) can be bounded as follows. The proof is given in Appendix A.

Lemma 1 (Variance bound for accelerated SVRG) *The variance of $\mathcal{G}_y$ (2) can be bounded as*

$$\begin{aligned}\mathbb{E}_i\left[\|\mathcal{G}_y - \nabla f(y)\|^2\right] \le\ & 2L\big(f(\tilde{x}) - f(y) - \langle \nabla f(y), \tilde{x} - y\rangle\big) - \|\nabla f(y)\|^2 \\ & + \underbrace{2\,\langle \nabla f(y), D\nabla f(\tilde{x})\rangle}_{\mathcal{R}_1} - \langle \nabla f(\tilde{x}), D\nabla f(\tilde{x})\rangle\,. \end{aligned} \tag{3}$$

In general, except for the dense case, where we can drop the last three terms above by completing the square, this upper bound will always be correlated with the sparsity (i.e., D). This correlation causes the strong sparsity assumption in [47]. We may consider a more specific case where $D = nI_d$. In this case, the last three terms above can be written as $(n-1)\,\|\nabla f(y)\|^2 - n\,\|\nabla f(y) - \nabla f(\tilde{x})\|^2$, which is not always non-positive.

Inspecting (3), we see that it is basically the term $\mathcal{R}_1$, which could be positive, that causes the issue. We thus propose a novel *sparse variance correction* for accelerated SVRG, which is designed to perfectly cancel $\mathcal{R}_1$. The correction is added to the coupling step (Step 7):

$$y_k = \vartheta \cdot z_k + \underbrace{(1-\vartheta)\cdot \tilde{x}_s}_{\text{Negative Momentum}} - \underbrace{\varphi \cdot D\nabla f(\tilde{x}_s)}_{\text{Sparse Variance Correction}}\,.$$

This correction neutralizes all the negative effect of the sparsity, in a similar way as how the negative momentum cancels the term $\langle \nabla f(y), \tilde{x} - y\rangle$ in the analysis [1]. We can understand this correction as a variance reducer that controls the additional sparse variance. Then we have the following sparsity-independent convergence result for Algorithm 1, and its proof is given in Appendix B.

3. It can be shown that when κ is large, almost no sparsity is allowed in their result.

Algorithm 2 AS-Acc-SVRG: Asynchronous Sparse Accelerated SVRG

```
Input: initial guess $x_0 \in \mathbb{R}^d$, constant $\omega > 1$ which controls the restart frequency.
Initialize: set $m, \vartheta, \varphi, \eta, S, R$ according to Theorem 2, initialize shared $z$ and diagonal matrix $D$.
 1: for $r = 0, \ldots, R-1$ do                                      ▷ performing restarts
 2:     $\tilde{x}_0 = z = x_r$.
 3:     for $s = 0, \ldots, S-1$ do
 4:         Compute in parallel and store $\nabla f(\tilde{x}_s)$.
 5:         while number of samples $\leq m$ do in parallel
 6:             Sample $i$ uniformly in $[n]$ and let $T_i$ be the support of $f_i$.
 7:             $[\hat{z}]_{T_i}$ = inconsistent read of $z$ on $T_i$.
 8:             $[\hat{y}]_{T_i} = \vartheta \cdot [\hat{z}]_{T_i} + (1-\vartheta) \cdot [\tilde{x}_s]_{T_i} - \varphi \cdot [D\nabla f(\tilde{x}_s)]_{T_i}$.
 9:             $[u]_{T_i} = -\eta \cdot \big(\nabla f_i([\hat{y}]_{T_i}) - \nabla f_i([\tilde{x}_s]_{T_i}) + D_i \nabla f(\tilde{x}_s)\big)$.
10:             for $v \in T_i$ do
11:                 $[z]_v = [z]_v + [u]_v$.                          ▷ coordinate-wise atomic write
12:             end for
13:         end while
14:         $\tilde{x}_{s+1} = \hat{y}_t$, where $\hat{y}_t$ is chosen uniformly at random among $\hat{y}$ in the previous epoch.
15:     end for
16:     $x_{r+1} = \frac{1}{S}\sum_{s=0}^{S-1} \tilde{x}_{s+1}$.
17: end for
Output: $x_R$.
```

Theorem 1 *For any constant $\omega > 1$, let $m = \Theta(n)$, $\vartheta = \frac{\sqrt{m}}{\sqrt{\kappa}+\sqrt{m}}$, $\varphi = \frac{1-\vartheta}{L}$, $\eta = \frac{1-\vartheta}{L\vartheta}$, $S = \big\lceil 2\omega\sqrt{\frac{\kappa}{m}}\big\rceil$. For any accuracy $\epsilon > 0$, we restart $R = O\big(\log \frac{f(x_0)-f(x^\star)}{\epsilon}\big)$ rounds. Then, Algorithm 1 outputs x_R satisfying $\mathbb{E}[f(x_R)] - f(x^\star) \leq \epsilon$ in $O\left(\max\{n, \sqrt{\kappa n}\} \log \frac{f(x_0)-f(x^\star)}{\epsilon}\right)$ oracle complexity.*

This complexity matches the lower bound established in [43] (up to a log factor), and is substantially faster than the $O\big((n+\kappa)\log\frac{1}{\epsilon}\big)$ complexity of sparse approximated SVRG and SAGA derived in [20, 21] in the ill-conditioned regime ($\kappa \gg n$). We provide some additional remarks on the construction of Algorithm 1 in Appendix C.

4. Asynchronous Sparse Accelerated SVRG

We then extend Algorithm 1 into the asynchronous setting (Algorithm 2), and analyze it under the perturbed iterate framework proposed in [25]. Note that Algorithm 2 degenerates into Algorithm 1 if there is only one thread (or worker). We provide a detailed review on the perturbed iterate analysis framework and formally define the ordering of sequences $\{\hat{z}_k\}$, $\{\hat{y}_k\}$ in Appendix D. The analysis in this line of work crucially relies on the following two assumptions, and we discuss them in details in Appendix E.

Assumption 1 (unbiasedness) *$\hat{z}_k$ is independent of the sample i_k, i.e., $\mathbb{E}[\mathcal{G}_{\hat{y}_k} | \hat{z}_k] = \nabla f(\hat{y}_k)$, where $\mathcal{G}_{\hat{y}_k} \triangleq \nabla f_{i_k}(\hat{y}_k) - \nabla f_{i_k}(\tilde{x}_s) + D_{i_k}\nabla f(\tilde{x}_s)$.*

Assumption 2 (bounded overlaps) *There exists a uniform bound τ on the maximum number of iterations that can overlap together.*

A subtlety here is that due to the periodic synchronization structure of Algorithm 2, we always have $\tau \leq m$. Defining the same sparsity measure as in [33]: $\Delta = \frac{1}{n} \cdot \max_{v\in[d]} |\{i : v \in T_i\}|$, we establish the convergence result of Algorithm 2 as follows. We first present the following guarantee for any τ given that some upper estimation of τ is available. The proof is provided in Appendix E.

Theorem 2 *For any $m \geq \widetilde{\tau} \geq \tau$ and any constant $\omega > 1$, let $m = \Theta(n)$, $\vartheta = \frac{\sqrt{m}}{\sqrt{\kappa(1+2\sqrt{\Delta}\widetilde{\tau})}+\sqrt{m}}$, $\varphi = \frac{1-\vartheta}{L}$, $\eta = \frac{(1-\vartheta)}{L\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}$ and $S = \left\lceil 2\omega\sqrt{\frac{\kappa}{m}(1+2\sqrt{\Delta}\widetilde{\tau})}\right\rceil$. For any accuracy $\epsilon > 0$, we restart $R = O\big(\log\frac{f(x_0)-f(x^\star)}{\epsilon}\big)$ rounds. Then, Algorithm 2 outputs x_R satisfying $\mathbb{E}\left[f(x_R)\right] - f(x^\star) \leq \epsilon$ in $O\left(\max\left\{n, \sqrt{\kappa n(1+2\sqrt{\Delta}\widetilde{\tau})}\right\}\log\frac{f(x_0)-f(x^\star)}{\epsilon}\right)$ oracle complexity.*

An interesting observation is that Theorem 2 establishes an $O(\sqrt{\kappa n\tau})$ dependence, which seems to conflict with the $\Omega(\tau\sqrt{\kappa})$ lower bound in the deterministic $n = 1$ case (by adapting Theorem 3.15 in [2] with delayed gradient). Certainly there is no contradiction. The subtlety is again the periodic synchronization structure of Algorithm 2. When $n = 1$, we have $\tau \leq m = \Theta(1)$ and Algorithm 2 is "almost synchronous". Based on this theorem, it is direct to identify the region of τ where a theoretical linear speed-up is achievable. The proof is straightforward and is thus omitted.

Corollary 2.1 (Speed-up condition) *In Theorem 2, suppose $\tau \leq O\left(\frac{1}{\sqrt{\Delta}}\max\left\{\frac{n}{\kappa}, 1\right\}\right)$. Then, setting $\widetilde{\tau} = O\left(\frac{1}{\sqrt{\Delta}}\max\left\{\frac{n}{\kappa}, 1\right\}\right)$, we have $S = O\left(\max\left\{1, \sqrt{\frac{\kappa}{n}}\right\}\right)$, which leads to the total oracle complexity #grad $= R \cdot S \cdot (n+2m) = O\big(\max\{n, \sqrt{\kappa n}\}\log\frac{f(x_0)-f(x^\star)}{\epsilon}\big)$.*

Since $\tau \leq m$, the precise linear speed-up condition of AS-Acc-SVRG is that $\tau = O(n)$ and $\tau = O(\frac{1}{\sqrt{\Delta}}\max\left\{\frac{n}{\kappa}, 1\right\})$, which is identical to that of ASAGA (cf., Corollary 9 in [21]) and slightly better than that of KroMagnon (cf., Corollary 18 in [21]). We provide some insights about the asynchronous acceleration in Appendix F.

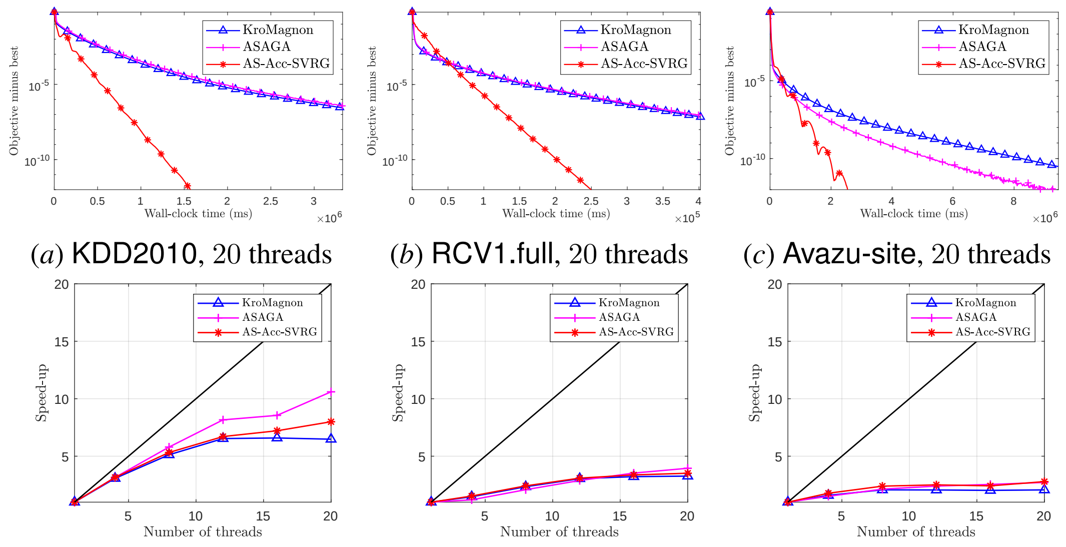
5. Experiments

We present numerical results on optimizing the ℓ_2-logistic regression problem:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n}\log\left(1+\exp\left(-b_i\langle a_i, x\rangle\right)\right) + \frac{\mu}{2}\|x\|^2, \tag{4}$$

where $a_i \in \mathbb{R}^d$, $b_i \in \{-1, +1\}$, $i \in [n]$ are the data samples and μ is the regularization parameter. We use the datasets from LIBSVM website [3], including KDD2010 [46], RCV1 [22], News20 [17], Avazu [15]. All the datasets are normalized to ensure a precise control on κ. We focus on the ill-conditioned case where $\kappa \gg n$ (the case where acceleration is effective). Detailed experimental setup and dataset descriptions can be found in Appendix L.

We compare the practical convergence and speed-up of AS-Acc-SVRG (Algorithm 2) with KroMagnon and ASAGA in Figure 1. We do not compare with the empirical method MiG in [47], which requires us to tune two highly correlated parameters with only limited insights. This is expensive or even prohibited for large scale tasks. For the compared methods, we only tune the τ-related constants in their theoretical parameter settings. That is, we tune the constant $1 + 2\sqrt{\Delta}\widetilde{\tau}$ in Theorem 2



(*a*) KDD2010, 20 threads (*b*) RCV1.full, 20 threads (*c*) Avazu-site, 20 threads

(*d*) KDD2010, speed-up (*e*) RCV1.full, speed-up (*f*) Avazu-site, speed-up

Figure 1: Convergence and speed-up for asynchronous sparse methods. Speed-up is the improvement on the wall-clock time to achieve 10^{-5} sub-optimality relative to using a single thread.

for AS-Acc-SVRG and the constant c in the step size $\frac{1}{cL}$ for KroMagnon and ASAGA. In fact, in all the experiments, we simply fixed the constant to 1 for AS-Acc-SVRG (the same parameters as the serial case), which worked smoothly. The main tuning effort was devoted to KroMagnon and ASAGA, and we tried to choose their step sizes as large as possible. The detailed choices can be found in Appendix L. Due to the scale of the problems, we only conducted a single run. From Figure 1, we see significant improvement of AS-Acc-SVRG for ill-conditioned tasks and similar practical speed-ups among the three methods, which verifies Theorem 2 and Corollary 2.1. We also observe a strong correlation between the practical speed-up and Δ, which is predicted by the theoretical $O(1/\sqrt{\Delta})$ dependence. It seems that we cannot reproduce the speed-up results in [21] on the RCV1.full dataset. This could be due to the difference in the programming languages, as we used C++ and they used Scalar. Pedregosa et al. [32] also used C++ implementation and we observe a similar $10\times$ speed-up of ASAGA on the KDD2010 dataset.

Please see the appendices for additional experiments, including an ablation study of sparse variance correction (Appendix G), comparison between sparse estimator and lagged update (Appendix H), verifying the $O(\sqrt{\kappa})$ dependence (Appendix J) and a sanity check (Appendix K).

6. Conclusion

In this work, we proposed a new asynchronous accelerated SVRG method which achieves the optimal oracle complexity under the perturbed iterate framework. We show that it requires the same linear speed-up condition as the non-accelerated methods. Empirical results justified our findings. The limitations of this work are that it requires a known μ and it does not support proximal operators. Directly incorporating the sparse proximal techniques in [32] results in an accelerated method that requires the knowledge of $\nabla f(x^\star)$.

References

[1] Z. Allen-Zhu. Katyusha: The First Direct Acceleration of Stochastic Gradient Methods. *Journal of Machine Learning Research*, 18(1):8194–8244, 2017.

[2] S. Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[3] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[4] S. Cyrus, B. Hu, B. Van Scoy, and L. Lessard. A robust accelerated optimization algorithm for strongly convex functions. In *Annual American Control Conference (ACC)*, pages 1376–1381. IEEE, 2018.

[5] A. Defazio, F. R. Bach, and S. Lacoste-Julien. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

[6] O. Devolder, F. Glineur, and Y. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1):37–75, 2014.

[7] B. Dubois-Taine, S. Vaswani, R. Babanezhad, M. Schmidt, and S. Lacoste-Julien. SVRG Meets AdaGrad: Painless Variance Reduction. *arXiv preprint arXiv:2102.09645*, 2021.

[8] J. C. Duchi, M. I. Jordan, and H. B. McMahan. Estimation, Optimization, and Parallelism when Data is Sparse. In *Advances in Neural Information Processing Systems*, pages 2832–2840, 2013.

[9] C. Fang, Y. Huang, and Z. Lin. Accelerating asynchronous algorithms for convex optimization by momentum compensation. *arXiv preprint arXiv:1802.09747*, 2018.

[10] B. Gu, W. Xian, Z. Huo, C. Deng, and H. Huang. A Unified q-Memorization Framework for Asynchronous Stochastic Optimization. *Journal of Machine Learning Research*, 21:190–1, 2020.

[11] R. Hannah, F. Feng, and W. Yin. A2BCD: Asynchronous Acceleration with Optimal Complexity. In *International Conference on Learning Representations*, 2019.

[12] T. Hofmann, A. Lucchi, S. Lacoste-Julien, and B. McWilliams. Variance Reduced Stochastic Gradient Descent with Neighbors. In *Advances in Neural Information Processing Systems*, pages 2305–2313, 2015.

[13] R. Johnson and T. Zhang. Accelerating Stochastic Gradient Descent using Predictive Variance Reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

[14] P. Joulani, A. György, and C. Szepesvári. Think out of the "Box": Generically-Constrained Asynchronous Composite Optimization and Hedging. In *Advances in Neural Information Processing Systems*, pages 12225–12235, 2019.

[15] Y. Juan, Y. Zhuang, W. Chin, and C. Lin. Field-aware Factorization Machines for CTR Prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 43–50. ACM, 2016.

[16] H. Karimi, J. Nutini, and M. Schmidt. Linear Convergence of Gradient and Proximal-Gradient Methods Under the Polyak-Łojasiewicz Condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.

[17] S. S. Keerthi, D. DeCoste, and T. Joachims. A Modified Finite Newton Method for Fast Solution of Large Scale Linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

[18] J. Konečnỳ and P. Richtárik. Semi-stochastic gradient descent methods. *arXiv preprint arXiv:1312.1666*, 2013.

[19] G. Lan, Z. Li, and Y. Zhou. A unified variance-reduced accelerated gradient method for convex optimization. In *Advances in Neural Information Processing Systems*, volume 32, pages 10462–10472, 2019.

[20] R. Leblond, F. Pedregosa, and S. Lacoste-Julien. ASAGA: Asynchronous Parallel SAGA. In *Proceedings of the Twentieth International Conference on Artificial Intelligence and Statistics*, volume 54, pages 46–54, 2017.

[21] R. Leblond, F. Pedregosa, and S. Lacoste-Julien. Improved Asynchronous Parallel Optimization Analysis for Stochastic Incremental Methods. *Journal of Machine Learning Research*, 19:81:1–81:68, 2018.

[22] D. D. Lewis, Y. Yang, T. Russell-Rose, and F. Li. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[23] Z. Li. ANITA: An Optimal Loopless Accelerated Variance-Reduced Gradient Method. *arXiv preprint arXiv:2103.11333*, 2021.

[24] X. Lian, Y. Huang, Y. Li, and J. Liu. Asynchronous Parallel Stochastic Gradient for Nonconvex Optimization. In *Advances in Neural Information Processing Systems*, pages 2737–2745, 2015.

[25] H. Mania, X. Pan, D. S. Papailiopoulos, B. Recht, K. Ramchandran, and M. I. Jordan. Perturbed Iterate Analysis for Asynchronous Stochastic Optimization. *SIAM Journal on Optimization*, 27(4):2202–2229, 2017.

[26] Y. Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.

[27] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

[28] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[29] Y. Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.

[30] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč. SARAH: A Novel Method for Machine Learning Problems Using Stochastic Recursive Gradient. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2613–2621, 2017.

[31] L. M. Nguyen, P. H. Nguyen, M. van Dijk, P. Richtárik, K. Scheinberg, and M. Takác. SGD and Hogwild! Convergence Without the Bounded Gradients Assumption. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3747–3755, 2018.

[32] F. Pedregosa, R. Leblond, and S. Lacoste-Julien. Breaking the Nonsmooth Barrier: A Scalable Parallel Method for Composite Optimization. In *Advances in Neural Information Processing Systems*, pages 56–65, 2017.

[33] B. Recht, C. Ré, S. J. Wright, and F. Niu. Hogwild!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

[34] N. L. Roux, M. Schmidt, and F. R. Bach. A Stochastic Gradient Method with an Exponential Convergence Rate for Finite Training Sets. In *Advances in Neural Information Processing Systems*, pages 2663–2671, 2012.

[35] C. D. Sa, C. Zhang, K. Olukotun, and C. Ré. Taming the Wild: A Unified Analysis of Hogwild-Style Algorithms. In *Advances in Neural Information Processing Systems*, pages 2674–2682, 2015.

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

[37] F. Shang, Y. Liu, K. Zhou, J. Cheng, K. K. W. Ng, and Y. Yoshida. Guaranteed Sufficient Decrease for Stochastic Variance Reduced Gradient Optimization. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84, pages 1027–1036, 2018.

[38] Z. Shi, N. Loizou, P. Richtárik, and M. Takáč. AI-SARAH: Adaptive and Implicit Stochastic Recursive Gradient Methods. *arXiv preprint arXiv:2102.09700*, 2021.

[39] C. Song, Y. Jiang, and Y. Ma. Variance Reduction via Accelerated Dual Averaging for Finite-Sum Optimization. In *Advances in Neural Information Processing Systems*, volume 33, pages 833–844, 2020.

[40] S. U. Stich, A. Mohtashami, and M. Jaggi. Critical Parameters for Scalable Distributed Learning with Large Batches and Asynchronous Updates. In *Proceedings of the Twenty-Fourth International Conference on Artificial Intelligence and Statistics*, volume 130, pages 4042–4050, 2021.

[41] C. Tan, S. Ma, Y. Dai, and Y. Qian. Barzilai-Borwein Step Size for Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems*, pages 685–693, 2016.

[42] A. B. Taylor, J. M. Hendrickx, and F. Glineur. Smooth strongly convex interpolation and exact worst-case performance of first-order methods. *Mathematical Programming*, 161(1-2): 307–345, 2017.

[43] B. E. Woodworth and N. Srebro. Tight Complexity Bounds for Optimizing Composite Objectives. In *Advances in Neural Information Processing Systems*, pages 3639–3647, 2016.

[44] L. Xiao and T. Zhang. A Proximal Stochastic Gradient Method with Progressive Variance Reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[45] L. Xiao, A. W. Yu, Q. Lin, and W. Chen. DSCOVR: Randomized Primal-Dual Block Coordinate Algorithms for Asynchronous Distributed Optimization. *Journal of Machine Learning Research*, 20(1):1634–1691, 2019.

[46] H.-F. Yu, H.-Y. Lo, H.-P. Hsieh, J.-K. Lou, T. G. McKenzie, J.-W. Chou, P.-H. Chung, C.-H. Ho, C.-F. Chang, Y.-H. Wei, et al. Feature engineering and classifier ensemble for KDD cup 2010. In *KDD cup*, 2010.

[47] K. Zhou, F. Shang, and J. Cheng. A Simple Stochastic Variance Reduced Algorithm with Fast Convergence Rates. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5980–5989, 2018.

[48] K. Zhou, Q. Ding, F. Shang, J. Cheng, D. Li, and Z.-Q. Luo. Direct Acceleration of SAGA using Sampled Negative Momentum. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, pages 1602–1610, 2019.

[49] K. Zhou, Y. Jin, Q. Ding, and J. Cheng. Amortized Nesterov's Momentum: A Robust Momentum and Its Application to Deep Learning. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence*, pages 211–220, 2020.

[50] K. Zhou, A. M.-C. So, and J. Cheng. Boosting First-Order Methods by Shifting Objective: New Schemes with Faster Worst-Case Rates. In *Advances in Neural Information Processing Systems*, pages 15405–15416, 2020.

[51] K. Zhou, L. Tian, A. M.-C. So, and J. Cheng. Practical Schemes for Finding Near-Stationary Points of Convex Finite-Sums. *arXiv preprint arXiv:2105.12062*, 2021.

Appendix A. Proof of Lemma 1

$$\begin{aligned}
&\mathbb{E}_i\left[\|\mathcal{G}_y - \nabla f(y)\|^2\right]\\
&\overset{(a)}{=} \mathbb{E}_i\left[\|\nabla f_i(y) - \nabla f_i(\tilde{x}) + D_i \nabla f(\tilde{x})\|^2\right] - \|\nabla f(y)\|^2\\
&= \mathbb{E}_i\left[\|\nabla f_i(y) - \nabla f_i(\tilde{x})\|^2\right] + 2\mathbb{E}_i\left[\langle \nabla f_i(y) - \nabla f_i(\tilde{x}), D_i\nabla f(\tilde{x})\rangle\right]\\
&\quad + \mathbb{E}_i\left[\|D_i\nabla f(\tilde{x})\|^2\right] - \|\nabla f(y)\|^2\\
&\overset{(b)}{=} \mathbb{E}_i\left[\|\nabla f_i(y) - \nabla f_i(\tilde{x})\|^2\right] + 2\mathbb{E}_i\left[\langle \nabla f_i(y) - \nabla f_i(\tilde{x}), D\nabla f(\tilde{x})\rangle\right]\\
&\quad + \mathbb{E}_i\left[\langle D\nabla f(\tilde{x}), D_i\nabla f(\tilde{x})\rangle\right] - \|\nabla f(y)\|^2\\
&= \mathbb{E}_i\left[\|\nabla f_i(y) - \nabla f_i(\tilde{x})\|^2\right] + 2\left\langle \nabla f(y), D\nabla f(\tilde{x})\right\rangle - \left\langle \nabla f(\tilde{x}), D\nabla f(\tilde{x})\right\rangle - \|\nabla f(y)\|^2\\
&\overset{(c)}{\leq} 2L\big(f(\tilde{x}) - f(y) - \langle \nabla f(y), \tilde{x} - y\rangle\big) + 2\left\langle \nabla f(y), D\nabla f(\tilde{x})\right\rangle - \left\langle \nabla f(\tilde{x}), D\nabla f(\tilde{x})\right\rangle\\
&\quad - \|\nabla f(y)\|^2,
\end{aligned}$$

where (a) uses the unbiasedness $\mathbb{E}_i\left[\mathcal{G}_y\right] = \nabla f(y)$, (b) follows from that $\nabla f_i(y) - \nabla f_i(\tilde{x})$ is supported on T_i and $P_i^2 = P_i$, and (c) uses the interpolation condition of f_i (see Section 2).

Appendix B. Proof of Theorem 1

We omit the superscript s in the one-epoch analysis for clarity. Using convexity, we have

$$\begin{aligned}
f(y_k) - f(x^\star) &\leq \left\langle \nabla f(y_k), y_k - x^\star\right\rangle\\
&= \left\langle \nabla f(y_k), y_k - z_k\right\rangle + \left\langle \nabla f(y_k), z_k - x^\star\right\rangle\\
&\overset{(\star)}{\leq} \frac{1-\vartheta}{\vartheta}\left\langle \nabla f(y_k), \tilde{x}_s - y_k\right\rangle - \frac{\varphi}{\vartheta}\left\langle \nabla f(y_k), D\nabla f(\tilde{x}_s)\right\rangle + \left\langle \nabla f(y_k), z_k - x^\star\right\rangle, \qquad (5)
\end{aligned}$$

where $(\star)$ follows from the construction $y_k = \vartheta z_k + (1-\vartheta)\,\tilde{x}_s - \varphi D\nabla f(\tilde{x}_s)$.

Denote $\mathcal{G}_{y_k} = \nabla f_{i_k}(y_k) - \nabla f_{i_k}(\tilde{x}_s) + D_{i_k}\nabla f(\tilde{x}_s)$. Based on the updating rule: $z_{k+1} = z_k - \eta\cdot\mathcal{G}_{y_k}$, it holds that

$$\begin{aligned}
&\|z_{k+1} - x^\star\|^2 = \|z_k - x^\star - \eta\cdot\mathcal{G}_{y_k}\|^2\\
\Rightarrow\ &\left\langle \mathcal{G}_{y_k}, z_k - x^\star\right\rangle = \frac{\eta}{2}\|\mathcal{G}_{y_k}\|^2 + \frac{1}{2\eta}\left(\|z_k - x^\star\|^2 - \|z_{k+1} - x^\star\|^2\right)\\
\overset{(\star)}{\Rightarrow}\ &\left\langle \nabla f(y_k), z_k - x^\star\right\rangle = \frac{\eta}{2}\mathbb{E}_{i_k}\left[\|\mathcal{G}_{y_k}\|^2\right] + \frac{1}{2\eta}\left(\|z_k - x^\star\|^2 - \mathbb{E}_{i_k}\left[\|z_{k+1} - x^\star\|^2\right]\right)\\
\Rightarrow\ &\left\langle \nabla f(y_k), z_k - x^\star\right\rangle = \frac{\eta}{2}\mathbb{E}_{i_k}\left[\|\mathcal{G}_{y_k} - \nabla f(y_k)\|^2\right] + \frac{\eta}{2}\|\nabla f(y_k)\|^2\\
&\qquad + \frac{1}{2\eta}\left(\|z_k - x^\star\|^2 - \mathbb{E}_{i_k}\left[\|z_{k+1} - x^\star\|^2\right]\right), \qquad (6)
\end{aligned}$$

where $(\star)$ follows from taking the expectation with respect to sample i_k.

Combine (5), (6) and use the variance bound in Lemma 1.

$$\begin{aligned}
f(y_k) - f(x^\star) \leq \eta L\big(f(\tilde{x}_s) - f(y_k)\big) &+ \left(\frac{1-\vartheta}{\vartheta} - \eta L\right) \langle \nabla f(y_k), \tilde{x}_s - y_k \rangle \\
&+ \left(\eta - \frac{\varphi}{\vartheta}\right) \langle \nabla f(y_k), D\nabla f(\tilde{x}_s) \rangle - \frac{\eta}{2} \langle \nabla f(\tilde{x}_s), D\nabla f(\tilde{x}_s) \rangle \\
&+ \frac{1}{2\eta}\left(\|z_k - x^\star\|^2 - \mathbb{E}_{i_k}\left[\|z_{k+1} - x^\star\|^2 \right] \right).
\end{aligned}$$

Substituting the choices $\eta = \frac{1-\vartheta}{L\vartheta}$ and $\varphi = \eta\vartheta = \frac{1-\vartheta}{L}$ and noting that $D \succ 0$, we obtain

$$\begin{aligned}
f(y_k) - f(x^\star) \leq &(1-\vartheta)\big(f(\tilde{x}_s) - f(x^\star)\big) \\
&+ \frac{L\vartheta^2}{2(1-\vartheta)}\left(\|z_k - x^\star\|^2 - \mathbb{E}_{i_k}\left[\|z_{k+1} - x^\star\|^2 \right] \right).
\end{aligned}$$

Summing the above inequality from $k = 0$ to $m-1$ and noting that $z_0^{s+1} = z_m^s$, we obtain

$$\begin{aligned}
\mathbb{E}\left[f(\tilde{x}_{s+1}) - f(x^\star)\right] &= \frac{1}{m}\sum_{k=0}^{m-1} \mathbb{E}\left[f(y_k) - f(x^\star)\right] \\
&\leq (1-\vartheta)\mathbb{E}\left[f(\tilde{x}_s) - f(x^\star)\right] \\
&\quad + \frac{L\vartheta^2}{2m(1-\vartheta)}\left(\mathbb{E}\left[\|z_0^s - x^\star\|^2\right] - \mathbb{E}\left[\|z_0^{s+1} - x^\star\|^2\right] \right).
\end{aligned}$$

Denoting $Q_s \triangleq \mathbb{E}\left[f(\tilde{x}_s) - f(x^\star)\right]$ and $P_s \triangleq \mathbb{E}\left[\|z_0^s - x^\star\|^2\right]$, we can write the above as

$$Q_{s+1} \leq \frac{1-\vartheta}{\vartheta}(Q_s - Q_{s+1}) + \frac{L\vartheta}{2m(1-\vartheta)}\left(P_s - P_{s+1}\right).$$

Summing this inequality from $s = 0$ to $S-1$ and using Jensen's inequality, we have

$$\begin{aligned}
\mathbb{E}\left[f(x_{r+1}) - f(x^\star)\right] &\leq \frac{1}{S}\sum_{s=0}^{S-1} Q_{s+1} \\
&\leq \frac{1-\vartheta}{\vartheta S}(Q_0 - Q_S) + \frac{L\vartheta}{2m(1-\vartheta)S}\left(P_0 - P_S\right) \\
&\overset{(\star)}{\leq} \frac{1-\vartheta}{\vartheta S}\mathbb{E}\left[f(x_r) - f(x^\star)\right] + \frac{L\vartheta}{2m(1-\vartheta)S}\mathbb{E}\left[\|x_r - x^\star\|^2\right],
\end{aligned}$$

where $(\star)$ follows from $\tilde{x}_0 = z_0^0 = x_r$.

Using μ-strong convexity at $(x_r, x^\star)$ (or quadratic growth), $f(x_r) - f(x^\star) \geq \frac{\mu}{2}\|x_r - x^\star\|^2$, we arrive at

$$\mathbb{E}\left[f(x_{r+1}) - f(x^\star)\right] \leq \left(\frac{1-\vartheta}{\vartheta S} + \frac{\kappa\vartheta}{m(1-\vartheta)S}\right)\mathbb{E}\left[f(x_r) - f(x^\star)\right].$$

Choosing $S = \left\lceil \omega \cdot \left(\frac{1-\vartheta}{\vartheta} + \frac{\kappa\vartheta}{m(1-\vartheta)}\right)\right\rceil$, we have $\mathbb{E}\left[f(x_{r+1}) - f(x^\star)\right] \leq \frac{1}{\omega}\cdot\mathbb{E}\left[f(x_r) - f(x^\star)\right]$. Thus, for any accuracy $\epsilon > 0$ and any constant $\omega > 1$, to guarantee that the output satisfies

$\mathbb{E}\left[f(x_R)\right] - f(x^\star) \le \epsilon$, we need to perform totally $R = O\left(\log \frac{f(x_0)-f(x^\star)}{\epsilon}\right)$ restarts. Note that the total oracle complexity of Algorithm 1 is #grad $= R \cdot S \cdot (n + 2m)$. Setting $m = \Theta(n)$ and minimizing S with respect to ϑ, we obtain the optimal choice $\vartheta = \frac{\sqrt{m}}{\sqrt{\kappa}+\sqrt{m}}$. In this case, we have $S = \left\lceil 2\omega\sqrt{\frac{\kappa}{m}}\right\rceil = O\left(\max\left\{1, \sqrt{\frac{\kappa}{n}}\right\}\right)$. Finally, the total oracle complexity is

$$\#\text{grad} = O\left(\max\left\{n, \sqrt{\kappa n}\right\} \log \frac{f(x_0) - f(x^\star)}{\epsilon}\right).$$

Appendix C. Additional Remarks about Algorithm 1

- We adopt a restart framework to handle the strong convexity, which is also used in [47] and is suggested in [1] (footnote 9). This framework allows us to relax the strong convexity assumption to quadratic growth. Other techniques for handling the strong convexity such as (i) assuming a strongly convex regularizer and using proximal update [1, 28], and (ii) the direct constructions in [19, 23, 50] fail to keep the inner iterates sparse, and we are currently not sure how to modify them.

- The restarting point x_{r+1} is chosen as the averaged point instead of a uniformly random one because large deviations are observed in the loss curve if a random restarting point is used.

- We can also use sparse variance correction to fix the sparsity issue in [47], which leads to a slightly different method and somewhat longer proof.

- Algorithm 1 degenerates to a strongly convex variant of Acc-SVRG-G proposed in [51] in the dense case ($D = I_d$), which summarizes our original inspiration. Note that Acc-SVRG-G is designed for minimizing the gradient norm and function value at the same time, which has nothing to do with handling sparsity.

- A general convex ($\mu = 0$) variant of Algorithm 1 can be derived by removing the restarts and adopting a variable parameter choice similar to Acc-SVRG-G in [51], which leads to a similar rate. We also find that our correction can be used in Varag [19] and ANITA [23] in the general convex case. Since the previous works on asynchronous lock-free optimization mainly focus on the strongly convex case, we omit the discussion here.

Appendix D. Perturbed Iterate Analysis Framework

Perturbed iterate analysis Let us denote the kth update as $\mathcal{G}_{\hat{y}_k} \triangleq \nabla f_{i_k}(\hat{y}_k) - \nabla f_{i_k}(\tilde{x}_s) + D_{i_k}\nabla f(\tilde{x}_s)$ in Algorithm 2. The precise ordering of the parallel updates will be defined in the next paragraph. Mania et al. [25] proposed to analyze the following virtual iterates:

$$z_{k+1} = z_k - \eta \cdot \mathcal{G}_{\hat{y}_k}, \text{ for } k = 0, \ldots, m-1. \tag{7}$$

They are called the virtual iterates because except for z_0 and z_m, other iterates may not exist in the shared memory due to the lock-free design. Then, the inconsistent read $\hat{z}_k$ is interpreted as a perturbed version of z_k, which will be formalized shortly. Note that z_m is precisely z in the shared memory after all the one-epoch updates are completed due to the atomic write requirement, which is critical in our analysis.

Ordering the iterates An important issue of the analysis in asynchrony is how to correctly order the updates that happen in parallel. Recht et al. [33] increases the counter k after each successful write of the update to the shared memory, and this ordering has been used in many follow-up works. Note that under this ordering, the iterates z_k at (7) exist in the shared memory. However, this ordering is incompatible with the unbiasedness assumption (Assumption 1 below), that is, enforcing the unbiasedness would require some additional overly strong assumption. Mania et al. [25] addressed this issue by increasing the counter k just before each inconsistent read of z. In this case, the unbiasedness can be simply enforced by reading all the coordinates of z (not just those on the support). Although this is expensive and is not used in their implementation, the unbiasedness is enforceable under this ordering, which is thus more reasonable. Leblond et al. [21] further refined and simplified their analysis by proposing to increase k after each inconsistent read of z is completed. This modification removes the dependence of $\hat{z}_k$ on "future" updates, i.e., on i_r for $r > k$, which significantly simplifies the analysis and leads to better speed-up conditions. See [21] for more detailed discussion on this issue. We follow the ordering of [21] to analyze Algorithm 2. Given this ordering, the value of $\hat{z}_k$ can be explicitly described as

$$[\hat{z}_k]_v = [z_0]_v - \eta \sum_{\substack{j \in \{0,\ldots,k-1\} \\ \text{s.t. coordinate } v \text{ was} \\ \text{written for } j \text{ before } k}} [\mathcal{G}_{\hat{y}_j}]_v.$$

Since $\hat{y}$ is basically composing $\hat{z}$ with constant vectors, it can also be ordered as

$$\hat{y}_k = \vartheta \hat{z}_k + (1 - \vartheta)\, \tilde{x}_s - \varphi D \nabla f(\tilde{x}_s). \tag{8}$$

Appendix E. Proof of Theorem 2

The analysis in this line of work crucially relies on the following two assumptions.

Assumption 1 (unbiasedness) *$\hat{z}_k$ is independent of the sample i_k, i.e., $\mathbb{E}\left[\mathcal{G}_{\hat{y}_k} | \hat{z}_k\right] = \nabla f(\hat{y}_k)$, where $\mathcal{G}_{\hat{y}_k} \triangleq \nabla f_{i_k}(\hat{y}_k) - \nabla f_{i_k}(\tilde{x}_s) + D_{i_k} \nabla f(\tilde{x}_s)$.*

As we discussed in Appendix D, the unbiasedness can be enforced by reading all the coordinates of z while in practice one would only read those necessary coordinates. This inconsistency exists in all the follow-up works that use the revised ordering [10, 14, 21, 25, 32, 47], and it is currently unknown how to avoid such an issue. Another inconsistency in Algorithm 2 is that in the implementation, when a $\hat{y}_k$ is selected as the next snapshot $\tilde{x}_{s+1}$, all the coordinates of $\hat{z}_k$ are loaded. This makes the choice of $\tilde{x}_{s+1}$ not necessarily uniformly random. KroMagnon (the improved version in [21]) also has this issue. In practice, no noticeable negative impact is observed for the two inconsistencies.

Assumption 2 (bounded overlaps) *There exists a uniform bound τ on the maximum number of iterations that can overlap together.*

This is a common assumption in the analysis with stale gradients. Under this assumption, the explicit effect of asynchrony can be modeled as:

$$\hat{z}_k = z_k + \eta \sum_{j=(k-\tau)_+}^{k-1} J_j^k \mathcal{G}_{\hat{y}_j}, \tag{9}$$

where J_j^k is a diagonal matrix with its elements in $\{0, 1\}$. The 1 elements indicate that $\hat{z}_k$ lacks some "past" updates on those coordinates.

We use the following lemma from [21] to bound the term in asynchrony. Although the gradient is evaluated at a different point, the same proof works in our case, and thus is omitted here.

Lemma 2 (Inequality (63) in [21]) *We have the following bound for the iterates z_k (defined at (7)) and $\hat{z}_k$ (defined at (9)) in one epoch of Algorithm 2:*

$$\mathbb{E}\left[\langle \mathcal{G}_{\hat{y}_k}, \hat{z}_k - z_k \rangle\right] \le \frac{\sqrt{\Delta}\eta}{2}\left(\sum_{j=(k-\tau)_+}^{k-1} \mathbb{E}\left[\|\mathcal{G}_{\hat{y}_j}\|^2\right] + \tau\mathbb{E}\left[\|\mathcal{G}_{\hat{y}_k}\|^2\right]\right).$$

We start with analyzing one epoch of virtual iterates defined at (7): $z_{k+1} = z_k - \eta \cdot \mathcal{G}_{\hat{y}_k}$. The expectation is first taken conditioned on the previous epochs.

$$\begin{aligned}
&\|z_{k+1} - x^\star\|^2 = \|z_k - x^\star - \eta \cdot \mathcal{G}_{\hat{y}_k}\|^2 \\
\Rightarrow\ & \langle \mathcal{G}_{\hat{y}_k}, \hat{z}_k - x^\star \rangle + \langle \mathcal{G}_{\hat{y}_k}, z_k - \hat{z}_k \rangle = \frac{\eta}{2}\|\mathcal{G}_{\hat{y}_k}\|^2 + \frac{1}{2\eta}\left(\|z_k - x^\star\|^2 - \|z_{k+1} - x^\star\|^2\right) \\
\overset{(\star)}{\Rightarrow}\ & \mathbb{E}\left[\langle \nabla f(\hat{y}_k), \hat{z}_k - x^\star \rangle\right] = \frac{\eta}{2}\mathbb{E}\left[\|\mathcal{G}_{\hat{y}_k}\|^2\right] + \mathbb{E}\left[\langle \mathcal{G}_{\hat{y}_k}, \hat{z}_k - z_k \rangle\right] \\
&\qquad + \frac{1}{2\eta}\left(\mathbb{E}\left[\|z_k - x^\star\|^2\right] - \mathbb{E}\left[\|z_{k+1} - x^\star\|^2\right]\right),
\end{aligned} \tag{10}$$

where $(\star)$ follows from taking the expectation and the unbiasedness assumption $\mathbb{E}\left[\mathcal{G}_{\hat{y}_k} | \hat{z}_k\right] = \nabla f(\hat{y}_k)$.

Using the convexity at $x^\star$ and the inconsistent $\hat{y}_k$ (ordered at (8)), we have

$$\begin{aligned}
f(\hat{y}_k) - f(x^\star) &\le \langle \nabla f(\hat{y}_k), \hat{y}_k - x^\star \rangle \\
&= \langle \nabla f(\hat{y}_k), \hat{y}_k - \hat{z}_k \rangle + \langle \nabla f(\hat{y}_k), \hat{z}_k - x^\star \rangle \\
&\overset{(\star)}{\le} \frac{1-\vartheta}{\vartheta}\langle \nabla f(\hat{y}_k), \tilde{x}_s - \hat{y}_k \rangle - \frac{\varphi}{\vartheta}\langle \nabla f(\hat{y}_k), D\nabla f(\tilde{x}_s) \rangle + \langle \nabla f(\hat{y}_k), \hat{z}_k - x^\star \rangle.
\end{aligned}$$

where $(\star)$ follows from the construction $\hat{y}_k = \vartheta \hat{z}_k + (1 - \vartheta)\tilde{x}_s - \varphi D\nabla f(\tilde{x}_s)$.

After taking the expectation and combining with (10), we obtain

$$\begin{aligned}
\mathbb{E}\left[f(\hat{y}_k)\right] - f(x^\star) \le\ & \frac{1-\vartheta}{\vartheta}\mathbb{E}\left[\langle \nabla f(\hat{y}_k), \tilde{x}_s - \hat{y}_k \rangle\right] - \frac{\varphi}{\vartheta}\mathbb{E}\left[\langle \nabla f(\hat{y}_k), D\nabla f(\tilde{x}_s) \rangle\right] + \frac{\eta}{2}\mathbb{E}\left[\|\mathcal{G}_{\hat{y}_k}\|^2\right] \\
& + \mathbb{E}\left[\langle \mathcal{G}_{\hat{y}_k}, \hat{z}_k - z_k \rangle\right] + \frac{1}{2\eta}\left(\mathbb{E}\left[\|z_k - x^\star\|^2\right] - \mathbb{E}\left[\|z_{k+1} - x^\star\|^2\right]\right).
\end{aligned}$$

Summing this inequality from $k = 0$ to $m - 1$, we have

$$\begin{aligned}
&\sum_{k=0}^{m-1} \mathbb{E}\left[f(\hat{y}_k) - f(x^\star)\right] \\
&\le \sum_{k=0}^{m-1}\left(\frac{1-\vartheta}{\vartheta}\mathbb{E}\left[\langle \nabla f(\hat{y}_k), \tilde{x}_s - \hat{y}_k \rangle\right] - \frac{\varphi}{\vartheta}\mathbb{E}\left[\langle \nabla f(\hat{y}_k), D\nabla f(\tilde{x}_s) \rangle\right] + \frac{\eta}{2}\mathbb{E}\left[\|\mathcal{G}_{\hat{y}_k}\|^2\right]\right) \\
&\quad + \sum_{k=0}^{m-1}\mathbb{E}\left[\langle \mathcal{G}_{\hat{y}_k}, \hat{z}_k - z_k \rangle\right] + \frac{1}{2\eta}\left(\|z_0 - x^\star\|^2 - \mathbb{E}\left[\|z_m - x^\star\|^2\right]\right).
\end{aligned}$$

Invoke Lemma 2 to bound the asynchronous perturbation.

$$
\begin{aligned}
&\sum_{k=0}^{m-1} \mathbb{E}\left[f(\hat{y}_k) - f(x^\star)\right] \\
\leq\ & \sum_{k=0}^{m-1} \left(\frac{1-\vartheta}{\vartheta}\mathbb{E}\left[\langle \nabla f(\hat{y}_k), \tilde{x}_s - \hat{y}_k\rangle\right] - \frac{\varphi}{\vartheta}\mathbb{E}\left[\langle \nabla f(\hat{y}_k), D\nabla f(\tilde{x}_s)\rangle\right] + \frac{\eta}{2}\mathbb{E}\left[\|\mathcal{G}_{\hat{y}_k}\|^2\right]\right) \\
& + \frac{\sqrt{\Delta}\eta}{2}\sum_{k=0}^{m-1}\left(\sum_{j=(k-\tau)_+}^{k-1}\mathbb{E}\left[\left\|\mathcal{G}_{\hat{y}_j}\right\|^2\right] + \tau\mathbb{E}\left[\|\mathcal{G}_{\hat{y}_k}\|^2\right]\right) + \frac{1}{2\eta}\left(\|z_0 - x^\star\|^2 - \mathbb{E}\left[\|z_m - x^\star\|^2\right]\right) \\
=\ & \sum_{k=0}^{m-1} \left(\frac{1-\vartheta}{\vartheta}\mathbb{E}\left[\langle \nabla f(\hat{y}_k), \tilde{x}_s - \hat{y}_k\rangle\right] - \frac{\varphi}{\vartheta}\mathbb{E}\left[\langle \nabla f(\hat{y}_k), D\nabla f(\tilde{x}_s)\rangle\right] + \frac{\eta(1+\sqrt{\Delta}\tau)}{2}\mathbb{E}\left[\|\mathcal{G}_{\hat{y}_k}\|^2\right]\right) \\
& + \frac{\sqrt{\Delta}\eta}{2}\sum_{k=0}^{m-1}\sum_{j=(k-\tau)_+}^{k-1}\mathbb{E}\left[\left\|\mathcal{G}_{\hat{y}_j}\right\|^2\right] + \frac{1}{2\eta}\left(\|z_0 - x^\star\|^2 - \mathbb{E}\left[\|z_m - x^\star\|^2\right]\right) \\
\leq\ & \sum_{k=0}^{m-1} \left(\frac{1-\vartheta}{\vartheta}\mathbb{E}\left[\langle \nabla f(\hat{y}_k), \tilde{x}_s - \hat{y}_k\rangle\right] - \frac{\varphi}{\vartheta}\mathbb{E}\left[\langle \nabla f(\hat{y}_k), D\nabla f(\tilde{x}_s)\rangle\right] + \frac{\eta(1+\sqrt{\Delta}\tau)}{2}\mathbb{E}\left[\|\mathcal{G}_{\hat{y}_k}\|^2\right]\right) \\
& + \frac{\sqrt{\Delta}\eta\tau}{2}\sum_{k=0}^{m-1}\mathbb{E}\left[\|\mathcal{G}_{\hat{y}_k}\|^2\right] + \frac{1}{2\eta}\left(\|z_0 - x^\star\|^2 - \mathbb{E}\left[\|z_m - x^\star\|^2\right]\right) \\
=\ & \sum_{k=0}^{m-1} \left(\frac{1-\vartheta}{\vartheta}\mathbb{E}\left[\langle \nabla f(\hat{y}_k), \tilde{x}_s - \hat{y}_k\rangle\right] - \frac{\varphi}{\vartheta}\mathbb{E}\left[\langle \nabla f(\hat{y}_k), D\nabla f(\tilde{x}_s)\rangle\right] + \frac{\eta(1+2\sqrt{\Delta}\tau)}{2}\mathbb{E}\left[\|\mathcal{G}_{\hat{y}_k}\|^2\right]\right) \\
& + \frac{1}{2\eta}\left(\|z_0 - x^\star\|^2 - \mathbb{E}\left[\|z_m - x^\star\|^2\right]\right).
\end{aligned}
$$

For any $m \geq \widetilde{\tau} \geq \tau$, we choose $\eta = \frac{(1-\vartheta)}{L\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}$, and then

$$
\begin{aligned}
&\sum_{k=0}^{m-1} \mathbb{E}\left[f(\hat{y}_k) - f(x^\star)\right] \\
\leq\ & \sum_{k=0}^{m-1} \left(\frac{1-\vartheta}{\vartheta}\mathbb{E}\left[\langle \nabla f(\hat{y}_k), \tilde{x}_s - \hat{y}_k\rangle\right] - \frac{\varphi}{\vartheta}\mathbb{E}\left[\langle \nabla f(\hat{y}_k), D\nabla f(\tilde{x}_s)\rangle\right] + \frac{1-\vartheta}{2L\vartheta}\mathbb{E}\left[\|\mathcal{G}_{\hat{y}_k}\|^2\right]\right) \\
& + \frac{L\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}{2(1-\vartheta)}\left(\|z_0 - x^\star\|^2 - \mathbb{E}\left[\|z_m - x^\star\|^2\right]\right).
\end{aligned}
$$

Note that $\mathbb{E}\left[\|\mathcal{G}_{\hat{y}_k}\|^2\right] = \mathbb{E}\left[\|\mathcal{G}_{\hat{y}_k} - \nabla f(\hat{y}_k)\|^2\right] + \mathbb{E}\left[\|\nabla f(\hat{y}_k)\|^2\right]$ due to the unbiasedness assumption. Using Lemma 1, we can conclude that

$$\begin{aligned}
&\sum_{k=0}^{m-1} \mathbb{E}\left[f(\hat{y}_k) - f(x^\star)\right] \\
&\leq \sum_{k=0}^{m-1} \left(\left(\frac{1-\vartheta}{L\vartheta} - \frac{\varphi}{\vartheta} \right) \mathbb{E}\left[\langle \nabla f(\hat{y}_k), D\nabla f(\tilde{x}_s)\rangle\right] + \frac{1-\vartheta}{\vartheta}\big(f(\tilde{x}_s) - \mathbb{E}\left[f(\hat{y}_k)\right]\big) \right) \\
&\quad + \frac{L\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}{2(1-\vartheta)} \left(\|z_0 - x^\star\|^2 - \mathbb{E}\left[\|z_m - x^\star\|^2\right] \right).
\end{aligned}$$

Choosing $\varphi = \frac{1-\vartheta}{L}$ and dividing both sides by m, we can arrange this inequality as

$$\begin{aligned}
\frac{1}{m}\sum_{k=0}^{m-1} \mathbb{E}\left[f(\hat{y}_k) - f(x^\star)\right] &\leq (1-\vartheta)\big(f(\tilde{x}_s) - f(x^\star)\big) \\
&\qquad + \frac{L\vartheta^2(1+2\sqrt{\Delta}\widetilde{\tau})}{2m(1-\vartheta)} \left(\|z_0 - x^\star\|^2 - \mathbb{E}\left[\|z_m - x^\star\|^2\right] \right).
\end{aligned}$$

Since $\tilde{x}_{s+1}$ is chosen uniformly at random from $\{\hat{y}_0, \ldots, \hat{y}_{m-1}\}$ and that $z_0^{s+1} = z_m^s$ (the first and the last virtual iterates exist in the shared memory, and z is unchanged after each epoch in Algorithm 2), the following holds

$$\begin{aligned}
\mathbb{E}\left[f(\tilde{x}_{s+1}) - f(x^\star)\right] &\leq (1-\vartheta)\big(f(\tilde{x}_s) - f(x^\star)\big) \\
&\qquad + \frac{L\vartheta^2(1+2\sqrt{\Delta}\widetilde{\tau})}{2m(1-\vartheta)} \left(\|z_0^s - x^\star\|^2 - \mathbb{E}\left[\left\|z_0^{s+1} - x^\star\right\|^2\right] \right).
\end{aligned}$$

For the sake of clarity, we denote $Q_s \triangleq \mathbb{E}\left[f(\tilde{x}_s) - f(x^\star)\right]$ and $P_s \triangleq \mathbb{E}\left[\|z_0^s - x^\star\|^2\right]$. Then, it holds that

$$Q_{s+1} \leq \frac{1-\vartheta}{\vartheta}(Q_s - Q_{s+1}) + \frac{L\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}{2m(1-\vartheta)}\left(P_s - P_{s+1}\right).$$

Summing this inequality from $s = 0$ to $S-1$ and using Jensen's inequality, we have

$$\begin{aligned}
\mathbb{E}\left[f(x_{r+1})\right] - f(x^\star) &\leq \frac{1}{S}\sum_{s=0}^{S-1} Q_{s+1} \\
&\leq \frac{1-\vartheta}{\vartheta S}(Q_0 - Q_S) + \frac{L\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}{2m(1-\vartheta)S}\left(P_0 - P_S\right) \\
&\leq \frac{1-\vartheta}{\vartheta S}Q_0 + \frac{L\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}{2m(1-\vartheta)S}P_0.
\end{aligned}$$

Using μ-strong convexity at $(x_r, x^\star)$ (or quadratic growth), we have $\|x_r - x^\star\|^2 \leq \frac{2}{\mu}\big(f(x_r) - f(x^\star)\big) \Leftrightarrow P_0 \leq \frac{2}{\mu}Q_0$ and thus

$$\mathbb{E}\left[f(x_{r+1})\right] - f(x^\star) \leq \left(\frac{1-\vartheta}{\vartheta S} + \frac{\kappa\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}{m(1-\vartheta)S} \right) \mathbb{E}\left[f(x_r) - f(x^\star)\right],$$

Letting $S = \left\lceil \omega \cdot \left(\frac{1-\vartheta}{\vartheta} + \frac{\kappa\vartheta(1+2\sqrt{\Delta}\widetilde{\tau})}{m(1-\vartheta)} \right) \right\rceil$, we have $\mathbb{E}\left[f(x_{r+1})\right] - f(x^\star) \leq \frac{1}{\omega} \cdot \mathbb{E}\left[f(x_r) - f(x^\star)\right]$. Then, since $\omega > 1$ is a constant, to achieve an ϵ-additive error, we need to restart totally $R = O\left(\log \frac{f(x_0) - f(x^\star)}{\epsilon}\right)$ times. Note that the total oracle complexity of Algorithm 2 is #grad $= R \cdot S \cdot (n + 2m)$. Setting $m = \Theta(n)$ and choosing ϑ that minimizes S, we obtain the optimal choice $\vartheta = \frac{\sqrt{m}}{\sqrt{\kappa(1+2\sqrt{\Delta}\widetilde{\tau})} + \sqrt{m}}$, which leads to

$$S = \left\lceil 2\omega \sqrt{\frac{\kappa}{m}(1 + 2\sqrt{\Delta}\widetilde{\tau})} \right\rceil = O\left(\max\left\{ 1, \sqrt{\frac{\kappa}{n}(1 + 2\sqrt{\Delta}\widetilde{\tau})} \right\} \right).$$

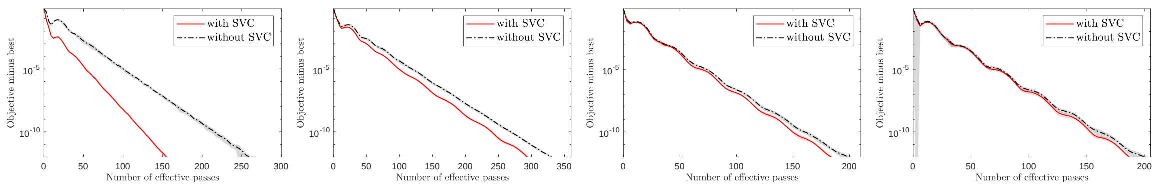
Finally, the total oracle complexity is

$$\#\text{grad} = O\left(\max\left\{ n, \sqrt{\kappa n(1 + 2\sqrt{\Delta}\widetilde{\tau})} \right\} \log \frac{f(x_0) - f(x^\star)}{\epsilon} \right).$$

Appendix F. Some Insights about the Asynchronous Acceleration

Let us first consider the serial case (Algorithm 1). Observe that in one epoch, the iterate y_k is basically composing z_k with constant vectors, we can equivalently write the update (Step 8) as $y_{k+1} = y_k - \eta\vartheta \cdot \mathcal{G}_{y_k}$. Thus, the inner loop of Algorithm 1 is identical to that of (sparse) SVRG. The difference is that at the end of each epoch, when the snapshot $\tilde{x}$ is changed, an offset (or momentum) is added to the iterate. This has been similarly observed for accelerated SVRG in [48]. Note that since the sequence z appears in the potential function, the current formulation of Algorithm 1 allows a cleaner analysis. Then, in the asynchronous case, the inner loop of Algorithm 2 can also be equivalently written as the updates of asynchronous SVRG, and the momentum is added at the end of each epoch. This gives us some insights about the identical speed-up condition in Corollary 2.1: Since the asynchronous perturbation only affects the inner loop, the momentum is almost uncorrupted, unlike the cases of noisy gradient oracle. That is, the asynchrony only corrupts the "non-accelerated part" of Algorithm 2, which thus leads to the same speed-up condition as the non-accelerated methods.

Appendix G. The Effectiveness of Sparse Variance Correction

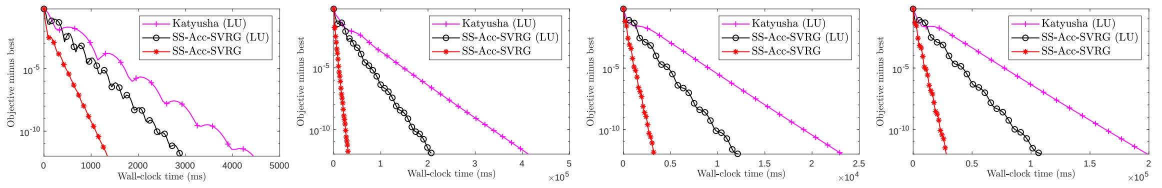


(*a*) Synthetic (*b*) KDD2010.S (*c*) RCV1.train (*d*) News20

Figure 2: Ablation study for the practical effect of sparse variance correction (abbreviated as SVC in the legends). Run 10 seeds. Shaded bands indicate ± 1 standard deviation.

We study the practical effect of the correction in the serial case. In Figure 2, "with SVC" refers to Algorithm 1 and "without SVC" is a naive extension of the dense version of Algorithm 1 into the sparse case (i.e., simply using a sparse estimator (2)), which suffers from the same sparsity issue as in [47]. For the two variants, we chose the same parameters in Theorem 1 to conduct an ablation study. In theory, we would expect the sparsity issue to be more severe if D is closer to nI_d (i.e., more sparse). Note that by definition, $\Delta = \max_{i\in[d]} D_{ii}^{-1}$, and thus $D \succeq \Delta^{-1} I_d$. Hence, smaller Δ indicates that D is closer to nI_d. From Figure 2d to 2a, Δ is decreasing and we observe more improvement from the correction. The improvement is consistent in our experiments, which justifies the effectiveness of sparse variance correction.

Appendix H. Sparse Estimator v.s Lagged Update



(*a*) Synthetic (*b*) KDD2010.S (*c*) RCV1.train (*d*) News20

Figure 3: Running time comparison between using sparse gradient estimator and lagged update (abbreviated as LU). The wall-clock time and objective value are averaged over 10 runs.

We examine the running time improvement from adopting the sparse estimator compared with using the lagged update technique. Lagged update technique handles sparsity by maintaining a last seen iteration for each coordinate. When a coordinate is involved in the current iteration, it computes an accumulated update from the last seen iteration in closed form. Such computation is dropped when adopting a sparse estimator, which is the source of running time improvement. Moreover, it is extremely difficult to extend the lagged update technique into the asynchronous setting as discussed in Appendix E in [21]. In Figure 3, we compare SS-Acc-SVRG (Algorithm 1) with lagged update implementations of (dense) SS-Acc-SVRG and Katyusha. Their default parameters were used.

Note that the lagged update technique is much trickier to implement, especially for Katyusha. We need to derive the closed-form solution of some complicated constant recursive sequence for the accumulated update. Plots with respect to effective passes are provided in Appendix J, in which SS-Acc-SVRG and Katyusha show similar performance.

Appendix I. The Effect of the Constant ω (the Restart Frequency)

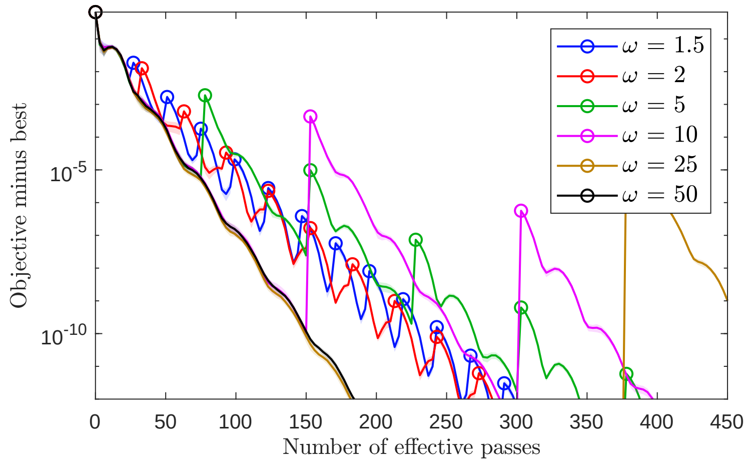


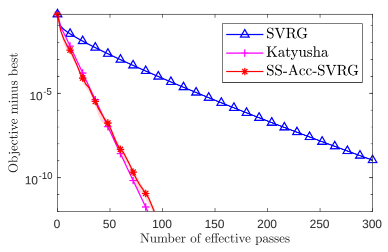
Figure 4: The practical effect of ω. RCV1.train, run 10 seeds. The circle marks the restarting points, i.e., $\{x_r\}$. Shaded bands indicate ± 1 standard deviation.

We numerically evaluate the effect of the constant ω in Figure 4. We choose ω from 1.5 (frequent restart) to 50 (not restart in this task). The results on the News20 and KDD2010.S datasets are basically identical, and thus are omitted here. As we can see, unfortunately, the restarts only deteriorate the performance. An intuitive explanation is that the restart strategy is more conservative as it periodically retracts the point. In theory, the explicit dependence on ω (in the serial case) is

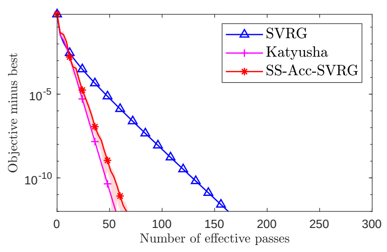
$$\frac{1}{\log \omega}\left\lceil 2\omega\sqrt{\frac{\kappa}{m}}\right\rceil,$$

which suggests that if ω is large, the convergence on the restarting points $\{x_r\}$ will be slower. This is observed in Figure 4. However, what our current theory cannot explain is the superior performance of not performing restart, and we are not aware of a situation where the "aggressiveness" of no restart would hurt the convergence. Optimally tuning ω in the complexity will only lead to a small ω that is close to 1. Further investigation is needed for the convergence of Algorithms 1 and 2 without restart (we have strong numerical evidence in Appendix J that without restart, they are still optimal).

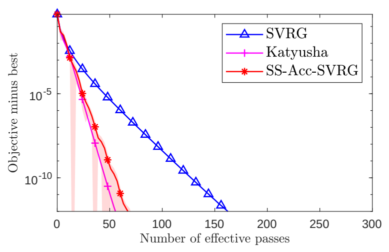
Appendix J. Justifying the $\sqrt{\kappa}$ Dependence



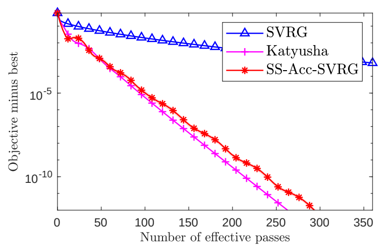
(*a*) KDD2010.S, $\mu = 10^{-6}$.



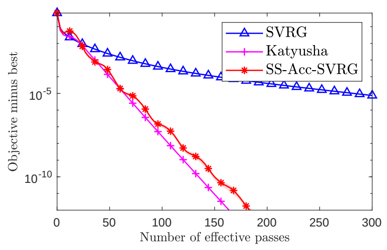
(*b*) RCV1.train, $\mu = 10^{-5}$.



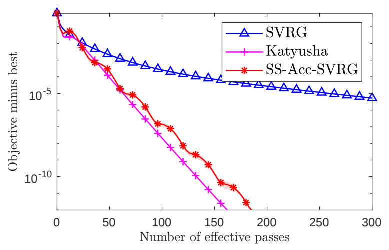
(*c*) News20, $\mu = 10^{-5}$.



(*d*) KDD2010.S, $\mu = 10^{-7}$.



(*e*) RCV1.train, $\mu = 10^{-6}$.



(*f*) News20, $\mu = 10^{-6}$.

Figure 5: Justifying the $\sqrt{\kappa}$ dependence. Run 10 seeds. Shaded bands indicate ± 1 standard deviation.

We choose μ to be 10-times larger (Figures 5a to 5c) than the ones specified in Table 1 (Figures 5d to 5f) to verify the $\sqrt{\kappa}$ dependence. In this case, κ is 10-times smaller and we expect the accelerated methods to be around 3-times faster in terms of the number of data passes. This has been observed across all the datasets for Katyusha and SS-Acc-SVRG, which verifies the accelerated rate.

Appendix K. Sanity Check for Our Implementation

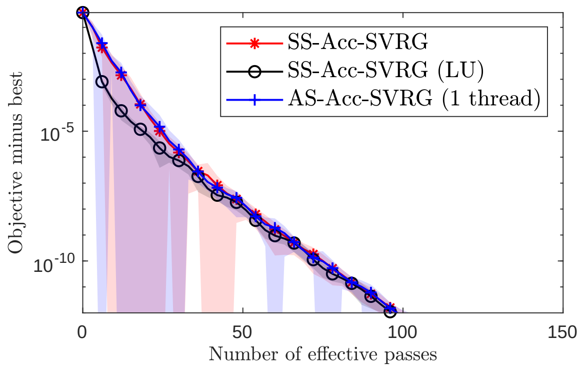


Figure 6: Sanity check. a9a.dense, $\mu = 10^{-6}$. Run 20 seeds. Shaded bands indicate ± 1 standard deviation.

If the dataset is fully dense, then the following three schemes should be equivalent: SS-Acc-SVRG, SS-Acc-SVRG with lagged update implementation and AS-Acc-SVRG with a single thread. We construct a fully dense dataset by adding a small positive number to each elements in the a9a dataset [3] ($n = 32{,}561, d = 123$), and we name this dataset as a9a.dense. The sanity check is then provided in Figure 6. SS-Acc-SVRG (LU) shows slightly different convergence because we used an averaged snapshot instead of a random one in its implementation. Implementing lagged update technique with a random snapshot is quite tricky, since the last seen iteration might be in the previous epochs in this case. We tested SS-Acc-SVRG with an averaged snapshot and it shows almost identical convergence as SS-Acc-SVRG (LU).

Appendix L. Experimental Setup

Table 1: Summary of the datasets. Density is the ratio of non-zero elements.

Scale	Dataset	n	d	μ	Density	Δ	Description
Small	Synthetic	100,000	100,000	10^{-7}	10^{-5}	10^{-5}	Identity data matrix with random labels
	KDD2010.S	70,000	29,890,095	10^{-7}	10^{-6}	0.15	The first 70,000 data samples of KDD2010
	RCV1.train	20,242	47,236	10^{-6}	$1.6 \cdot 10^{-3}$	0.42	
	News20	19,996	1,355,191	10^{-6}	$3.4 \cdot 10^{-4}$	0.93	
Large	KDD2010	19,264,097	29,890,095	10^{-10}	10^{-6}	0.16	
	RCV1.full	697,641	47,236	10^{-9}	$1.5 \cdot 10^{-3}$	0.43	Combined test and train sets of RCV1
	Avazu-site	23,567,843	999,962	10^{-10}	$1.5 \cdot 10^{-5}$	0.96	Avazu-site.train

We provide the dataset descriptions and the choices of μ in Table 1. We conduct serial experiments on the small datasets and asynchronous experiments on the large ones. The asynchronous experiments were conducted on a multi-core HPC. All the methods are implemented in C++.

Here we discuss two subtleties in the implementation:

- If each $f_i(x) = \log\big(1 + \exp(-b_i \langle a_i, x\rangle)\big) + \frac{\mu}{2}\|x\|^2$, then it is supported on every coordinate due to the ℓ_2-regularization. Following [21], we sparsify the gradient of the regularization term as $\mu D_i x$; or equivalently, $f_i(x) = \log\big(1 + \exp(-b_i \langle a_i, x\rangle)\big) + \frac{\mu}{2}\langle x, D_i x\rangle$. Clearly, this f_i also sums up to the objective (4). The difference is that the Lipschitz constant of f_i will be larger, i.e., from $0.25 + \mu$ to at most $0.25 + \mu n$. Since we focus on the ill-conditioned case ($L \gg \mu n$), this modification will not have significant effect.

- In Theorems 1 and 2, all the parameters have been chosen optimally in our analysis except for ω, which controls the restart frequency. Despite all the theoretical benefits of the restart framework mentioned in Section C, in practice, we do not find performing restarts lead to a faster convergence. Thus, we do not perform restarts in our experiments (or equivalently, we choose a relatively large $\omega = 50$). Clearly, this choice will not make our method a "heuristic" since the theorems hold for any $\omega > 1$. Detailed discussion is given in Appendix I.

Serial setup We ran serial experiments on an HP Z440 machine with a single Intel Xeon E5-1630v4 with 3.70GHz cores, 16GB RAM, Ubuntu 18.04 LTS with GCC 9.4.0, MATLAB R2021a. We fixed the epoch length $m = 2n$ and chose the default parameters: SS-Acc-SVRG follows Theorem 1; Katyusha uses $\tau_2 = \frac{1}{2}, \tau_1 = \sqrt{\frac{m}{3\kappa}}, \alpha = \frac{1}{3\tau_1 L}$ [1]; SVRG uses $\eta = \frac{1}{4L}$.

Asynchronous setup We ran asynchronous experiments on a Dell PowerEdge R840 HPC with four Intel Xeon Platinum 8160 CPU @ 2.10GHz each with 24 cores, 768GB RAM, CentOS 7.9.2009 with GCC 9.3.1, MATLAB R2021a. We implemented the original version of KroMagnon in [25]. We fixed the epoch length $m = 2n$ for KroMagnon and AS-Acc-SVRG. AS-Acc-SVRG used the same parameters as in the serial case for all datasets. The step sizes of KroMagnon and ASAGA were chosen as follows: On RCV1.full, KroMagnon uses $\frac{1}{L}$, ASAGA uses $\frac{1}{1.3L}$; on Avazu-site.train, KroMagnon uses $\frac{1}{2L}$, ASAGA uses $\frac{1}{2L}$; on KDD2010, KroMagnon uses $\frac{1}{2L}$, ASAGA uses $\frac{1}{3L}$. Due to the scale of the tasks, we cannot do fine-grained grid search for the step sizes of KroMagnon and ASAGA. They were chosen to ensure a stable and consistent performance in the 20-thread experiments.