Perturbed Iterate Analysis for Asynchronous Stochastic Optimization

Horia Mania
Xinghao Pan
Dimitris Papailiopoulos
Benjamin Recht
Kannan Ramchandran
Michael Jordan

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-12
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-12.html

May 1, 2019

Perturbed Iterate Analysis for Asynchronous Stochastic Optimization

Horia Mania

Research Project

Submitted to the Department of Electrical Engineering and Computer Science, University of California at Berkeley, in partial satisfaction of the requirements for the degree of Master of Science, Plan II.
Approval for the Report and Comprehensive Examination:

Professor Michael I. Jordan
Research Advisor

Professor Benjamin Recht
Research Advisor

Abstract

We introduce and analyze stochastic optimization methods where the input to each update is perturbed by bounded noise. We show that this framework forms the basis of a unified approach to analyze asynchronous implementations of stochastic optimization algorithms, by viewing them as serial methods operating on noisy inputs. Using our perturbed iterate framework, we provide new analyses of the HOGWILD! algorithm and asynchronous stochastic coordinate descent, that are simpler than earlier analyses, remove many assumptions of previous models, and in some cases yield improved upper bounds on the convergence rates. We proceed to apply our framework to develop and analyze KROMAGNON: a novel, parallel, sparse stochastic variance-reduced gradient (SVRG) algorithm. We demonstrate experimentally on a 16-core machine that the sparse and parallel version of SVRG is in some cases more than four orders of magnitude faster than the standard SVRG algorithm.

Keywords: stochastic optimization, asynchronous algorithms, parallel machine learning.

1 Introduction

Asynchronous parallel stochastic optimization algorithms have recently gained significant traction in algorithmic machine learning. A large body of recent work has demonstrated that near-linear speedups are achievable, in theory and practice, on many common machine learning tasks [1–8]. Moreover, when these lock-free algorithms are applied to non-convex optimization, significant speedups are still achieved with no loss of statistical accuracy. This behavior has been demonstrated in practice in state-of-the-art deep learning systems such as Google's Downpour SGD [9] and Microsoft's Project Adam [10].

Although asynchronous stochastic algorithms are simple to implement and enjoy excellent performance in practice, they are challenging to analyze theoretically. The current analyses require lengthy derivations and several assumptions that may not reflect realistic system behavior. Moreover, due to the difficult nature of the proofs, the algorithms analyzed are often simplified versions of those actually run in practice.

In this paper, we propose a general framework for deriving convergence rates for parallel, lock-free, asynchronous first-order stochastic algorithms. We interpret the algorithmic effects of asynchrony as perturbing the stochastic iterates with bounded noise. This interpretation allows us to show how a variety of asynchronous first-order algorithms can be analyzed as their serial counterparts operating on noisy inputs. The advantage of our framework is that it yields elementary convergence proofs, can remove or relax simplifying assumptions adopted in prior art, and can yield improved bounds when compared to earlier work.

We demonstrate the general applicability of our framework by providing new convergence analyses for Hogwild!, *i.e.*, the asynchronous stochastic gradient method (SGM), for asynchronous stochastic coordinate descent (ASCD), and KroMagnon: a novel asynchronous sparse version of the stochastic variance-reduced gradient (SVRG) method [11]. In particular, we provide a modified version of SVRG that allows for sparse updates, we show that this method can be parallelized in the asynchronous model, and we provide convergence guarantees using our framework. Experimentally, the asynchronous, parallel sparse SVRG achieves nearly-linear speedups on a machine with 16 cores and is sometimes four orders of magnitude faster than the standard (dense) SVRG method.

1.1 Related work

The algorithmic tapestry of parallel stochastic optimization is rich and diverse extending back at least to the late 60s [12]. Much of the contemporary work in this space is built upon the foundational work of Bertsekas, Tsitsiklis et al. [13, 14]; the shared memory access model that we are using in this work, is very similar to the partially asynchronous model introduced in the aforementioned manuscripts. Recent advances in parallel and distributed computing technologies have generated renewed interest in the theoretical understanding and practical implementation of parallel stochastic algorithms [15–20].

The power of lock-free, asynchronous stochastic optimization on shared-memory multicore systems was first demonstrated in the work of [1]. The authors introduce Hogwild!, a completely lock-free and asynchronous parallel stochastic gradient method (SGM) that exhibits nearly linear speedups for a variety of machine learning tasks. Inspired by Hogwild!, several authors developed lock-free and asynchronous algorithms that move beyond SGM, such as the work of Liu et al. on parallel stochastic coordinate descent [5, 21]. Additional work in first order optimization and beyond [6–8, 22, 23], extending to parallel iterative linear solvers [24, 25], has further shown that linear speedups are possible in the asynchronous shared memory model.

2 Perturbed Stochastic Gradients

Preliminaries and Notation We study parallel asynchronous iterative algorithms that minimize convex functions $f(\mathbf{x})$ with $\mathbf{x} \in \mathbb{R}^d$. The computational model is the same as that of Niu et al. [1]: a number of cores have access to the same shared memory, and each of them can read and update components of $\mathbf{x}$ in the shared memory. The algorithms that we consider are asynchronous and lock-free: cores do not coordinate their reads or writes, and while a core is reading/writing other cores can update the shared variables in $\mathbf{x}$.

We focus our analysis on functions f that are L-smooth and m-strongly convex. A function f is L-smooth if it is differentiable and has Lipschitz gradients

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\| \text{ for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d,$$

where $\|\cdot\|$ denotes the Euclidean norm. Strong convexity with parameter $m > 0$ imposes a curvature condition on f:

$$f(\mathbf{x}) \geq f(\mathbf{y}) + \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle + \frac{m}{2}\|\mathbf{x} - \mathbf{y}\|^2 \text{ for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d.$$

Strong convexity implies that f has a unique minimum $\mathbf{x}^*$ and satisfies

$$\langle \nabla f(\mathbf{x}) - \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle \geq m\|\mathbf{x} - \mathbf{y}\|^2.$$

In the following, we use i, j, and k to denote iteration counters, while reserving v and u to denote coordinate indices. We use $\mathcal{O}(1)$ to denote absolute constants.

Perturbed Iterates A popular way to minimize convex functions is via *first-order stochastic* algorithms. These algorithms can be described using the following general iterative expression:

$$\mathbf{x}_{j+1} = \mathbf{x}_j - \gamma \mathbf{g}(\mathbf{x}_j, \xi_j), \tag{2.1}$$

where ξ_j is a random variable independent of $\mathbf{x}_j$ and $\mathbf{g}$ is an unbiased estimator of the true gradient of f at $\mathbf{x}_j$: $\mathbb{E}_{\xi_j}\mathbf{g}(\mathbf{x}_j, \xi_j) = \nabla f(\mathbf{x}_j)$. The success of first-order stochastic techniques partly lies in their computational efficiency: the small computational cost of using noisy gradient estimates trumps the gains of using true gradients.

A major advantage of the iterative formula in (2.1) is that—in combination with strong convexity, and smoothness inequalities—one can easily track algorithmic progress and establish convergence rates to the optimal solution. Unfortunately, the progress of asynchronous parallel algorithms cannot be precisely described or analyzed using the above iterative framework. Processors do not read from memory actual iterates $\mathbf{x}_j$, as there is no global clock that synchronizes reads or writes while different cores write/read "stale" variables.

In the subsequent sections, we show that the following simple perturbed variant of Eq. (2.1) can capture the algorithmic progress of asynchronous stochastic algorithms. Consider the following iteration

$$\mathbf{x}_{j+1} = \mathbf{x}_j - \gamma \mathbf{g}(\mathbf{x}_j + \mathbf{n}_j, \xi_j), \tag{2.2}$$

where $\mathbf{n}_j$ is a stochastic error term. For simplicity let $\hat{\mathbf{x}}_j = \mathbf{x}_j + \mathbf{n}_j$. Then,

$$\begin{aligned}
&\|\mathbf{x}_{j+1} - \mathbf{x}^*\|^2 = \|\mathbf{x}_j - \gamma \mathbf{g}(\hat{\mathbf{x}}_j, \xi_j) - \mathbf{x}^*\|^2 \\
&= \|\mathbf{x}_j - \mathbf{x}^*\|^2 - 2\gamma\langle \mathbf{x}_j - \mathbf{x}^*, \mathbf{g}(\hat{\mathbf{x}}_j, \xi_j)\rangle + \gamma^2\|\mathbf{g}(\hat{\mathbf{x}}_j, \xi_j)\|^2 \\
&= \|\mathbf{x}_j - \mathbf{x}^*\|^2 - 2\gamma\langle \hat{\mathbf{x}}_j - \mathbf{x}^*, \mathbf{g}(\hat{\mathbf{x}}_j, \xi_j)\rangle + \gamma^2\|\mathbf{g}(\hat{\mathbf{x}}_j, \xi_j)\|^2 + 2\gamma\langle \hat{\mathbf{x}}_j - \mathbf{x}_j, \mathbf{g}(\hat{\mathbf{x}}_j, \xi_j)\rangle,
\end{aligned} \tag{2.3}$$

where in the last equation we added and subtracted the term $2\gamma\langle\hat{\mathbf{x}}_j, \mathbf{g}(\hat{\mathbf{x}}_j, \xi_j)\rangle$.

We assume that $\hat{\mathbf{x}}_j$ and ξ_j are independent. However, in contrast to recursion (2.1), we no longer require $\mathbf{x}_j$ to be independent of ξ_j. The importance of the above independence assumption will become clear in the next section.

We now take the expectation of both sides in (2.3). Since $\hat{\mathbf{x}}_j$ and $\mathbf{x}^*$ are independent of ξ_j, we use iterated expectations to obtain $\mathbb{E}\langle\hat{\mathbf{x}}_j - \mathbf{x}^*, \mathbf{g}(\hat{\mathbf{x}}_j, \xi_j)\rangle = \mathbb{E}\langle\hat{\mathbf{x}}_j - \mathbf{x}^*, \nabla f(\hat{\mathbf{x}}_j)\rangle$. Moreover, since f is m-strongly convex, we know that

$$\langle\hat{\mathbf{x}}_j - \mathbf{x}^*, \nabla f(\hat{\mathbf{x}}_j)\rangle \geq m\|\hat{\mathbf{x}}_j - \mathbf{x}^*\|^2 \geq \frac{m}{2}\|\mathbf{x}_j - \mathbf{x}^*\|^2 - m\|\hat{\mathbf{x}}_j - \mathbf{x}_j\|^2, \tag{2.4}$$

where the second inequality is a simple consequence of the triangle inequality. Now, let $a_j = \mathbb{E}\|\mathbf{x}_j - \mathbf{x}^*\|^2$ and substitute (2.4) back into Eq. (2.3) to get

$$a_{j+1} \leq (1-\gamma m)a_j + \gamma^2 \underbrace{\mathbb{E}\|g(\hat{\mathbf{x}}_j, \xi_j)\|^2}_{R_0^j} + 2\gamma m \underbrace{\mathbb{E}\|\hat{\mathbf{x}}_j - \mathbf{x}_j\|^2}_{R_1^j} + 2\gamma \underbrace{\mathbb{E}\langle\hat{\mathbf{x}}_j - \mathbf{x}_j, \mathbf{g}(\hat{\mathbf{x}}_j, \xi_j)\rangle}_{R_2^j}. \tag{2.5}$$

The recursive equation (2.5) is key to our analysis. We show that for given R_0^j, R_1^j, and R_2^j, we can obtain convergence rates through elementary algebraic manipulations. Observe that there are three "error" terms in (2.5): R_0^j captures the stochastic gradient decay with each iteration, R_1^j captures the mismatch between the true iterate and its noisy estimate, and R_2^j measures the size of the projection of that mismatch on the gradient at each step. The key contribution of our work is to show that 1) this iteration can capture the algorithmic progress of asynchronous algorithms, and 2) the error terms can be bounded to obtain a $\mathcal{O}(\log(1/\epsilon)/\epsilon)$ rate for Hogwild!, and linear rates of convergence for asynchronous SCD and asynchronous sparse SVRG.

3 Analyzing Hogwild!

In this section, we provide a simple analysis of Hogwild!, the asynchronous implementation of SGM. We focus on functions f that are decomposable into n terms:

$$f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_{e_i}(\mathbf{x}), \tag{3.1}$$

where $\mathbf{x} \in \mathbb{R}^d$, and each $f_{e_i}(\mathbf{x})$ depends only on the coordinates indexed by the subset e_i of $\{1, 2, \ldots, d\}$. For simplicity we assume that the terms of f are differentiable; our results can be readily extended to non-differentiable f_{e_i}s.

We refer to the sets e_i as *hyperedges* and denote the set of hyperedges by $\mathcal{E}$. We sometimes refer to f_{e_i}s as the *terms* of f. As shown in Fig. 1, the hyperedges induce a bipartite graph between the n terms and the d variables in $\mathbf{x}$, and a conflict graph between the n terms. Let $\overline{\Delta}_{\mathrm{C}}$ be the average degree in the conflict graph; that is, the average number of terms that are in conflict with a single term. We assume that $\overline{\Delta}_{\mathrm{C}} \geq 1$, otherwise we could decompose the problem into smaller independent sub-problems. As we will see, under our perturbed iterate analysis framework the convergence rate of asynchronous algorithms depends on $\overline{\Delta}_{\mathrm{C}}$.

Hogwild! (Alg. 1) is a method to parallelize SGM in the asynchronous setting [1]. It is deployed on multiple cores that have access to shared memory, where the optimization variable $\mathbf{x}$ and the data points that define the f terms are stored. During its execution each core samples uniformly at random a hyperedge s from $\mathcal{E}$. It reads the coordinates $v \in s$ of the shared vector $\mathbf{x}$, evaluates ∇f_s at the point read, and finally adds $-\gamma\nabla f_s$ to the shared variable.

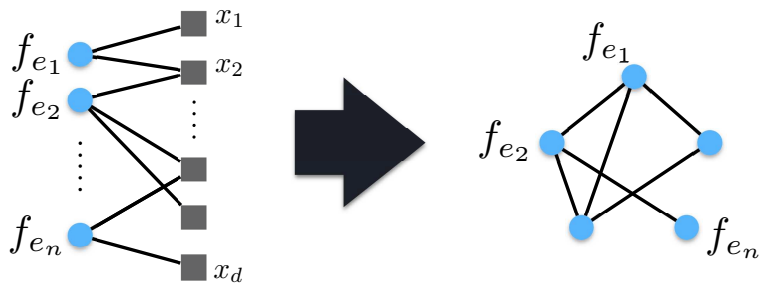


Figure 1: The bipartite graph on the left has as its leftmost vertices the n function terms and as its rightmost vertices the coordinates of $\mathbf{x}$. A term f_{e_i} is connected to a coordinate x_j if hyperedge e_i contains j (*i.e.*, if the i-th term is a function of that coordinate). The graph on the right depicts a conflict graph between the function terms. The vertices denote the function terms, and two terms are joined by an edge if they conflict on at least one coordinate in the bipartite graph.

Algorithm 1 HOGWILD!

```
while number of sampled hyperedges ≤ T do in parallel
    sample a random hyperedge s
    [x̂]_s = an inconsistent read of the shared variable [x]_s
    [u]_s = −γ · g([x̂]_s, s)
    for v ∈ s do
        [x]_v = [x]_v + [u]_v                    // atomic write
    end for
end while
```

During the execution of HOGWILD! cores do not synchronize or follow an order between reads or writes. Moreover, they access (*i.e.*, read or write) a set of coordinates in $\mathbf{x}$ without the use of any locking mechanisms that would ensure a conflict-free execution. This implies that the reads/writes of distinct cores can intertwine in arbitrary ways, *e.g.*, while a core updates a subset of variables, before completing its task, other cores can read/write the same subset of variables.

In [1], the authors analyzed a variant of HOGWILD! in which several simplifying assumptions were made. Specifically, in [1] *1)* only a single coordinate per sampled hyperedge is updated (*i.e.*, the for loop in HOGWILD! is replaced with a single coordinate update); *2)* the authors assumed *consistent reads*, *i.e.*, it was assumed that while a core is reading the shared variable, no writes from other cores occur; *3)* the authors make an implicit assumption on the uniformity of the processing times of cores (explained in the following), that does not generically hold in practice. These simplifications alleviate some of the challenges in analyzing HOGWILD! and allowed the authors to provide a convergence result. As we show in the current paper, however, these simplifications are not necessary to obtain a convergence analysis. Our perturbed iterates framework can be used in an elementary way to analyze the original version of HOGWILD!, yielding improved bounds compared to earlier analyses.

3.1 Ordering the samples

A subtle but important point in the analysis of HOGWILD! is the need to define an order for the sampled hyperedges. A key point of difference of our work is that *we order the samples based on the order in which they were sampled*, not the order in which cores complete the processing of the samples.

Definition 1. *We denote by s_i the i-th sampled hyperedge in a run of Alg. 1.*

That is, s_i denotes the sample obtained when line 2 in Alg. 1 is executed for the i-th time. This is different from the original work of [1], in which the samples were ordered according to the completion time of each thread. The issue with such an ordering is that the distribution of the samples, conditioned on the ordering, is not always uniform; for example, hyperedges of small cardinality are more likely to be "early" samples. A uniform distribution is needed for the theoretical analysis of stochastic gradient methods, a point that is disregarded in [1]. Our ordering according to sampling time resolves this issue by guaranteeing uniformity among samples in a trivial way.

3.2 Defining read iterates and clarifying independence assumptions

Since the shared memory variable can change inconsistently during reads and writes, we also have to be careful about the notion of *iterates* in HOGWILD!.

Definition 2. *We denote by $\overline{\mathbf{x}}_i$ the contents of the shared memory before the i-th execution of line 2. Moreover, we denote by $\hat{\mathbf{x}}_i \in \mathbb{R}^d$ the vector, that in coordinates $v \in s_i$ contains exactly what the core that sampled s_i read. We then define $[\hat{\mathbf{x}}_i]_v = [\overline{\mathbf{x}}_i]_v$ for all $v \notin s_i$. Note that we do not assume consistent reads,* i.e., *the contents of the shared memory can potentially change while a core is reading.*

At this point we would like to briefly discuss an *independence* assumption held by all prior work. In the following paragraph, we explain why this assumption is not always true in practice. In Appendix A, we show how to lift such independence assumption, but for ease of exposition we do adopt it in our main text.

Assumption 1. *The vector $\hat{\mathbf{x}}_i$ is independent of the sampled hyperedge s_i.*

The above independence assumption is important when establishing the convergence rate of the algorithm, and has been held explicitly or implicitly in prior work [1, 5, 6, 21]. Specifically, when proving convergence rates for these algorithms we need to show via iterated expectations that $\mathbb{E}\langle \hat{\mathbf{x}}_i - \mathbf{x}^*, g(\hat{\mathbf{x}}_i, s_i)\rangle = \langle \hat{\mathbf{x}}_i - \mathbf{x}^*, \nabla(\hat{\mathbf{x}}_i)\rangle$, which follows from the independence of $\hat{\mathbf{x}}_i$ and s_i. However, observe that although $\overline{\mathbf{x}}_i$ is independent of s_i by construction, this is not the case for the vector $\hat{\mathbf{x}}_i$ read by the core that sampled s_i. For example, consider the scenario of two consecutively sampled hyperedges in Alg. 1 that overlap on a subset of coordinates. Then, say one core is reading the coordinates of the shared variables indexed by its hyperedge, while the second core is updating a subset of these coordinates. In this case, the values read by the first core depend on the support of the sampled hyperedge.

One way to rigorously enforce the independence of $\hat{\mathbf{x}}_i$ and s_i is to require the processors to read the *entire* shared variable $\mathbf{x}$ before sampling a new hyperedge. However, this might not be reasonable in practice, as the dimension of $\mathbf{x}$ tends to be considerably larger than the sparsity of the hyperedges. As we mentioned earlier, in Appendix A, we show how to overcome the issue of dependence and thereby remove Assumption 1; however, this results in a slightly more cumbersome analysis. To ease readability, in our main text we do adopt Assumption 1.

3.3 The perturbed iterates view of asynchrony

In this work, we assume that all writes are atomic, in the sense that they will be successfully recorded in the shared memory at some point. Atomicity is a reasonable assumption in practice, as it can be strictly enforced through compare-and-swap operations [1].

Assumption 2. *Every write in line 6 of Alg. 1 will complete successfully.*

This assumption implies that all writes will appear in the shared memory by the end of the execution, in the form of coordinate-wise updates. Due to commutativity the order in which these updates are recorded in the shared memory is irrelevant. Hence, after processing a total of T hyperedges the shared memory contains:

$$\underbrace{\overbrace{\mathbf{x}_0 - \gamma \mathbf{g}(\hat{\mathbf{x}}_0, s_0)}^{\mathbf{x}_1} - \ldots - \gamma \mathbf{g}(\hat{\mathbf{x}}_{T-1}, s_{T-1})}_{\mathbf{x}_T}, \tag{3.2}$$

where $\mathbf{x}_0$ is the initial guess and $\mathbf{x}_i$ is defined as the vector that contains *all* gradient updates up to sample s_{i-1}.

Remark 1. *Throughout this section we denote $\mathbf{g}(\mathbf{x}, s_j) = \nabla f_{s_j}(\mathbf{x})$, which we assume to be bounded: $\|\mathbf{g}(\mathbf{x}, s)\| \leq M$. Such a uniform bound on the norm of the stochastic gradient is true when operating on a bounded ℓ_∞ ball; this can in turn be enforced by a simple, coordinate-wise thresholding operator. We can refine our analysis by avoiding the uniform bound on $\|\mathbf{g}(\mathbf{x}, s)\|$, through a simple application of the co-coercivity lemma as it was used in [26]; in this case, our derivations would only require a uniform bound on $\|\mathbf{g}(\mathbf{x}^*, s)\|$. Our subsequent derivations can be adapted to the above, however to keep our derivations elementary we will use the uniform bound on $\|\mathbf{g}(\mathbf{x}, s)\|$.*

Remark 2. *Observe that although a core is only reading the subset of variables that are indexed by its sampled hyperedge, in* (3.2) *we use the entire vector $\hat{\mathbf{x}}$ as the input to the sampled gradient. We can do this since $\mathbf{g}(\hat{\mathbf{x}}_k, s_k)$ is independent of the coordinates of $\hat{\mathbf{x}}_k$ outside the support of hyperedge s_k.*

Using the above definitions, we define the perturbed iterates of HOGWILD! as

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma \mathbf{g}(\hat{\mathbf{x}}_i, s_i), \tag{3.3}$$

for $i = 0, 1, ..., T-1$, where s_i is the i-th uniformly sampled hyperedge. Observe that all but the first and last of these iterates are "fake": there might not be an actual time when they exist in the shared memory during the execution. However, $\mathbf{x}_0$ is what is stored in memory before the execution starts, and $\mathbf{x}_T$ is exactly what is stored in shared memory at the end of the execution.

We observe that the iterates in (3.3) place HOGWILD! in the perturbed gradient framework introduced in §2:

$$a_{j+1} \leq (1 - \gamma m) a_j + \gamma^2 \underbrace{\mathbb{E}\|g(\hat{\mathbf{x}}_j, s_j)\|^2}_{R_0^j} + 2\gamma m \underbrace{\mathbb{E}\|\hat{\mathbf{x}}_j - \mathbf{x}_j\|^2}_{R_1^j} + 2\gamma \underbrace{\mathbb{E}\langle \hat{\mathbf{x}}_j - \mathbf{x}_j, \mathbf{g}(\hat{\mathbf{x}}_j, s_j)\rangle}_{R_2^j}.$$

We are only left to bound the three error terms R_0^j, R_1^j, and R_2^j. Before we proceed, we note that for the technical soundness of our theorems, we have to also define a random variable that captures the *system randomness*. In particular, let ξ denote a random variable that encodes the randomness of the system (*i.e.*, random delays between reads and writes, gradient computation time, etc). Although we do not explicitly use ξ, its distribution is required implicitly to compute the expectations for the convergence analysis. This is because the random samples $s_0, s_1, \ldots, s_{T-1}$ do not fully determine the output of Alg. 1. However, $s_0, \ldots, s_{T-1}$ along with ξ completely determine the time of all reads and writes. We continue with our final assumption needed by our analysis, that is also needed by prior art.

Assumption 3 (Bounded overlaps)**.** *Two hyperedges s_i and s_j overlap in time if they are processed concurrently at some point during the execution of* HOGWILD!*. The time during which a hyperedge s_i is being processed begins when the sampling function is called and ends after the last coordinate of $\mathbf{g}(\hat{\mathbf{x}}_i, s_i)$ is written to the shared memory. We assume that there exists a number $\tau \geq 0$, such that the maximum number of sampled hyperedges that can overlap in time with a particular sampled hyperedge cannot be more than τ.*

The usefulness of the above assumption is that it essentially abstracts away all system details relative to delays, processing overlaps, and number of cores into a single parameter. Intuitively, τ can be perceived as a proxy for the number of cores, *i.e.*, we would expect that no more than roughly $O(\#\text{cores})$ sampled hyperedges overlap in time with a single hyperedge, assuming that the processing times across the samples are approximately similar. Observe that if τ is small, then we expect the distance between $\mathbf{x}_j$ and the noisy iterate $\hat{\mathbf{x}}_j$ to be small. In our perturbed iterate framework, if we set $\tau = 0$, then we obtain the classical iterative formula of serial SGM.

To quantify the distance between $\hat{\mathbf{x}}_j$ (*i.e.*, the iterate read by the core that sampled s_j) and $\mathbf{x}_j$ (*i.e.*, the "fake" iterate used to establish convergence rates), we observe that any difference between them is caused solely by hyperedges that overlap with s_j in time. To see this, let s_i be an "earlier" sample, *i.e.*, $i < j$, that does not overlap with s_j in time. This implies that the processing of s_i finishes before s_j starts being processed. Hence, the *full* contribution of $\gamma\mathbf{g}(\hat{\mathbf{x}}_i, s_i)$ will be recorded in both $\hat{\mathbf{x}}_j$ and $\mathbf{x}_j$ (for the latter this holds by definition). Similarly, if $i > j$ and s_i does not overlap with s_j in time, then neither $\hat{\mathbf{x}}_j$ nor $\mathbf{x}_j$ (for the latter, again by definition) contain *any* of the coordinate updates involved in the gradient update $\gamma\mathbf{g}(\hat{\mathbf{x}}_i, s_i)$. Assumption 3 ensures that if $i < j - \tau$ or $i > j + \tau$, the sample s_i *does not* overlap in time with s_j.

By the above discussion, and due to Assumption 3, there exist diagonal matrices $\mathbf{S}_i^j$ with diagonal entries in $\{-1, 0, 1\}$ such that

$$\hat{\mathbf{x}}_j - \mathbf{x}_j = \sum_{i=j-\tau,\, i\neq j}^{j+\tau} \gamma \mathbf{S}_i^j \mathbf{g}(\hat{\mathbf{x}}_i, s_i). \tag{3.4}$$

These diagonal matrices account for any possible pattern of (potentially) partial updates that can occur while hyperedge s_j is being processed. We would like to note that the above notation bears resemblance to the coordinate-update mismatch formulation of asynchronous coordinate-based algorithms, as in [21, 27, 28].

We now turn to the convergence proof, emphasizing its elementary nature within the perturbed iterate analysis framework. We begin by bounding the error terms R_1^j and R_2^j (R_0^j is already assumed to be at most M^2).

Lemma 3. HOGWILD! *satisfies the recursion in* (2.5) *with*

$$R_1^j = \mathbb{E}\|\hat{\mathbf{x}}_j - \mathbf{x}_j\|^2 \leq \gamma^2 M^2 \left(2\tau + 8\tau^2 \frac{\overline{\Delta}_C}{n}\right) \text{ and } R_2^j = \mathbb{E}\langle \hat{\mathbf{x}}_j - \mathbf{x}_j, \mathbf{g}(\hat{\mathbf{x}}_j, s_j)\rangle \leq 4\gamma M^2 \tau \frac{\overline{\Delta}_C}{n},$$

where $\overline{\Delta}_C$ is the average degree of the conflict graph between the hyperedges.

Proof. The norm of the mismatch can be bounded in the following way:

$$\begin{aligned} R_1^j = \gamma^2 \mathbb{E}\Bigg\| \sum_{\substack{i=j-\tau \\ i\neq j}}^{j+\tau} \mathbf{S}_i^j \mathbf{g}(\hat{\mathbf{x}}_i, s_i) \Bigg\|^2 &\leq \gamma^2 \sum_i \mathbb{E}\|\mathbf{S}_i^j \mathbf{g}(\hat{\mathbf{x}}_i, s_i)\|^2 + \gamma^2 \sum_{\substack{i,k \\ i\neq k}} \mathbb{E}\left|\langle \mathbf{S}_i^j \mathbf{g}(\hat{\mathbf{x}}_i, s_i), \mathbf{S}_k^j \mathbf{g}(\hat{\mathbf{x}}_k, s_k)\rangle\right| \\ &\leq \gamma^2 \sum_i \mathbb{E}\|\mathbf{g}(\hat{\mathbf{x}}_i, s_i)\|^2 + \gamma^2 \sum_{\substack{i,k \\ i\neq k}} \mathbb{E}\left\{\|\mathbf{g}(\hat{\mathbf{x}}_i, s_i)\|\|\mathbf{g}(\hat{\mathbf{x}}_k, s_k)\|\mathbb{1}(s_i \cap s_k \neq \emptyset)\right\}, \end{aligned}$$

since $\mathbf{S}_i^j$ are diagonal sign matrices and since the steps $\mathbf{g}(\hat{\mathbf{x}}_i, s_i)$ are supported on the samples s_i. We use the upper bound $\|\mathbf{g}(\hat{\mathbf{x}}_i, s_i)\| \leq M$ to obtain

$$R_1^j \leq 2\tau \cdot \gamma^2 M^2 + \gamma^2 M^2 (2\tau)^2 \Pr(s_i \cap s_j \neq \emptyset) \leq \gamma^2 M^2 \cdot \left(2\tau + 8\tau^2 \frac{\overline{\Delta}_{\mathrm{C}}}{n}\right).$$

The last step follows because two sampled hyperedges (sampled with replacement) intersect with probability at most $2\frac{\overline{\Delta}_{\mathrm{C}}}{n}$. We can bound R_2^j in a similar way:

$$R_2^j = \gamma \sum_{\substack{i=j-\tau \\ i \neq j}}^{j+\tau} \mathbb{E}\langle \mathbf{S}_i^j \mathbf{g}(\hat{\mathbf{x}}_i, s_i), \mathbf{g}(\hat{\mathbf{x}}_j, s_j)\rangle \leq \gamma M^2 \sum_{\substack{i=j-\tau \\ i \neq j}}^{j+\tau} \mathbb{E}\left\{\mathbb{1}(s_i \cap s_j \neq \emptyset)\right\} \leq 4\gamma M^2 \tau \frac{\overline{\Delta}_{\mathrm{C}}}{n}.$$

□

Plugging the bounds of Lemma 3 in our recursive formula, we see that HOGWILD! satisfies the recursion

$$a_{j+1} = (1 - \gamma m)\, a_j + \gamma^2 M^2 \Bigg(1 + \underbrace{8\tau \frac{\overline{\Delta}_{\mathrm{C}}}{n} + 4\gamma m \tau + 16 \gamma m \tau^2 \frac{\overline{\Delta}_{\mathrm{C}}}{n}}_{\delta}\Bigg). \tag{3.5}$$

On the other hand, serial SGM satisfies the recursion $a_{j+1} \leq (1 - \gamma m)\, a_j + \gamma^2 M^2$. If the step size is set to $\gamma = \frac{\epsilon m}{2M^2}$, it attains target accuracy ϵ in $T \geq 2M^2/(\epsilon m^2) \log\left(\frac{2a_0}{\epsilon}\right)$ iterations. Hence, when the term δ of (3.5) is order-wise constant, HOGWILD! satisfies the same recursion (up to constants) as serial SGM. This directly implies the main result of this section.

Theorem 4. *If the number of samples that overlap in time with a single sample during the execution of* HOGWILD! *is bounded as*

$$\tau = \mathcal{O}\left(\min\left\{\frac{n}{\overline{\Delta}_C}, \frac{M^2}{\epsilon m^2}\right\}\right),$$

HOGWILD!, *with step size* $\gamma = \frac{\epsilon m}{2M^2}$, *reaches an accuracy of* $\mathbb{E}\|\mathbf{x}_k - \mathbf{x}^*\|^2 \leq \epsilon$ *after*

$$T \geq \mathcal{O}(1) \frac{M^2 \log\left(\frac{a_0}{\epsilon}\right)}{\epsilon m^2}$$

iterations.

Since the iteration bound in the theorem is (up to a constant) the same as that of serial SGM, our result implies a linear speedup. We would like to note that an improved rate of $O(1/\epsilon)$ can be obtained by appropriately diminishing stepsizes per epoch (see, *e.g.*, [1,29]). Furthermore, observe that although the $\frac{M^2}{\epsilon m^2}$ bound on τ might seem restrictive, it is—up to a logarithmic factor—proportional to the total number of iterations required by HOGWILD! (or even serial SGM) to reach ϵ accuracy. Assuming that the average degree of the conflict graph is constant, and that we perform a constant number of passes over the data, *i.e.*, $T = c \cdot n$, then τ can be as large as $\tilde{\mathcal{O}}(n)$, *i.e.*, nearly linear in the number of function terms.[1]

[1] $\tilde{\mathcal{O}}$ hides logarithmic terms.

3.4 Comparison with the original Hogwild! analysis of [1]

Let us summarize the key points of improvement compared to the original HOGWILD! analysis:

- Our analysis is elementary and compact, and follows simply by bounding the R_0^j, R_1^j, and R_2^j terms, after introducing the perturbed gradient framework of § 2.
- We do not assume consistent reads: while a core is reading from the shared memory other cores are allowed to read, or write.
- In [1] the authors analyze a simplified version of HOGWILD! where for each sampled hyperedge only a randomly selected coordinate is updated. Here we analyze the "full-update" version of HOGWILD!.
- We order the samples by the order in which they were sampled, not by completion time. This allows to rigorously prove our convergence bounds, without assuming anything on the distribution of the processing time of each hyperedge. This is unlike [1], where there is an implicit assumption of uniformity with respect to processing times.
- The previous work of [1] establishes a nearly-linear speedup for HOGWILD! if τ is bounded as $\tau = \mathcal{O}\left(\sqrt[4]{n/\Delta_{\mathrm{R}}\Delta_{\mathrm{L}}^2}\right)$, where Δ_{R} is the maximum right degree of the term-variables bipartite graph, shown in Fig 1, and Δ_{L} is the maximum left degree of the same graph. Observe that $\Delta_{\mathrm{R}} \cdot \Delta_{\mathrm{L}}^2 \geq \Delta_{\mathrm{L}} \cdot \Delta_{\mathrm{C}}$, where Δ_{C} is the maximum degree of the conflict graph. Here, we obtain a linear speedup for up to $\tau = \mathcal{O}\left(\min\left\{n/\overline{\Delta}_{\mathrm{C}}, M^2/\epsilon m^2\right\}\right)$, where $\overline{\Delta}_{\mathrm{C}}$ is only the average degree of the conflict graph in Fig 1. Our bound on the delays can be orders of magnitude better than that of [1].

4 Asynchronous Stochastic Coordinate Descent

In this section, we use the perturbed gradient framework to analyze the convergence of asynchronous parallel stochastic coordinate descent (ASCD). This algorithm has been previously analyzed in [5, 21]. We show that the algorithm admits an elementary treatment in our perturbed iterate framework, under the same assumptions made for HOGWILD!.

Algorithm 2 ASCD

```
while iterations ≤ T do in parallel
    x̂ = an inconsistent read of the shared variable x
    Sample a coordinate s
    u_s = −γ · d[∇f(x)]_s
    [x]_s = [x]_s + u_s                                   // atomic write
end while
```

ASCD, shown in Alg. 2, is a linearly convergent algorithm for minimizing strongly convex functions f. At each iteration a core samples one of the coordinates, computes a full gradient update for that coordinate, and proceeds with updating a single element of the shared memory variable $\mathbf{x}$. The challenge in analyzing ASCD, compared to HOGWILD!, is that, in order to show linear convergence, we need to show that the error due to the asynchrony between cores decays fast when the iterates arrive close to the optimal solution. The perturbed iterate framework can handle this type of noise analysis in a straightforward manner, using simple recursive bounds.

We define $\hat{\mathbf{x}}_i$ as in the previous section, but now the samples s_i are coordinates sampled uniformly at random from $\{1, 2, \ldots, d\}$. After T samples have been processed completely, the following vector is contained in shared memory:

$$\underbrace{\overbrace{\mathbf{x}_0 - \gamma d[\nabla f(\hat{\mathbf{x}}_0)]_{s_0}\mathbf{e}_{s_0}}^{\mathbf{x}_1} - \ldots - \gamma d[\nabla f(\hat{\mathbf{x}}_{T-1})]_{s_{T-1}}\mathbf{e}_{s_{T-1}}}_{\mathbf{x}_T},$$

where $\mathbf{x}_0$ is the initial guess, $\mathbf{e}_{s_j}$ is the standard basis vector with a one at position s_j, $[\nabla f(\mathbf{x})]_{s_j}$ denotes the s_j-th coordinate of the gradient of f computed at $\mathbf{x}$. Similar to HOGWILD! in the previous section, ASCD satisfies the following iterative formula

$$\mathbf{x}_{j+1} = \mathbf{x}_j - \gamma \cdot d \cdot [\nabla f(\hat{\mathbf{x}}_j)]_{s_j}\mathbf{e}_{s_j} = \mathbf{x}_j - \gamma \cdot \mathbf{g}(\hat{\mathbf{x}}_j, s_j).$$

Notice that $\mathbb{E}_{s_j}\mathbf{g}(\hat{\mathbf{x}}_j, s_j) = \nabla f(\hat{\mathbf{x}}_j)$, and thus, similarly to HOGWILD!, ASCD's iterates $a_j = \mathbb{E}\|\mathbf{x}_j - \mathbf{x}^*\|^2$ satisfy the recursion of Eq. (2.5):

$$a_{j+1} \leq (1 - \gamma m)\, a_j + \gamma^2 \underbrace{\mathbb{E}\|g(\hat{\mathbf{x}}_j, s_j)\|^2}_{R_0^j} + 2\gamma m \underbrace{\mathbb{E}\|\hat{\mathbf{x}}_j - \mathbf{x}_j\|^2}_{R_1^j} + 2\gamma \underbrace{\mathbb{E}\langle \hat{\mathbf{x}}_j - \mathbf{x}_j, \mathbf{g}(\hat{\mathbf{x}}_j, s_j)\rangle}_{R_2^j}.$$

Before stating the main result of this section, let us introduce some further notation. Let us define the largest distance between the optimal vector, and the vector read by the cores during the execution of the algorithm:$\hat{a}_0 := \max_{0\leq k\leq T} \mathbb{E}\|\hat{\mathbf{x}}_k - \mathbf{x}^*\|^2$, a value which should be thought of as proportional to $a_0 = \mathbb{E}\|\mathbf{x}_0 - \mathbf{x}^*\|^2$. Furthermore, by a simple application of the L-Lipschitz assumption on f, we have a uniform bound on the norm of each computed gradient $M^2 := \max_{0\leq k\leq T} \mathbb{E}\|\nabla f(\hat{\mathbf{x}}_k)\|^2 \leq L^2\hat{a}_0$. Here we assume that the optimization takes place in an ℓ_∞ ball, so that $M < \infty$. This simply means that the iterates will never have infinitely large coordinate values. This assumption is made in previous work explicitly or implicitly, and in practice it can be implemented easily since the projection on an ℓ_∞ ball can be done component-wise. Finally, let us define the condition number of f as $\kappa := L/m$, where L is the Lipschitz constant, and m the strong convexity parameter.

Theorem 5. *Let the maximum number of coordinates that can be concurrently processed while a core is processing a single coordinate be at most*

$$\tau = \mathcal{O}\left(\min\left\{\frac{\kappa\sqrt{d}}{\log\left(\frac{\hat{a}_0}{\epsilon}\right)}, \sqrt[6]{d}\right\}\right).$$

Then, ASCD with step-size $\gamma = \frac{\mathcal{O}(1)}{dL\kappa}$ achieves an error $\mathbb{E}\|\mathbf{x}_k - \mathbf{x}^*\|^2 \leq \epsilon$ after

$$k \geq \mathcal{O}(1) \cdot d\kappa^2 \log\left(\frac{a_0}{\epsilon}\right)$$

iterations.

Using the recursive inequality (2.5), serial SCD with the same step-size as in the above theorem, can be shown to achieve the same accuracy as ASCD in the same number of steps. Hence, as long as the proxy for the number of cores is bounded as $\tau = \mathcal{O}(\min\{\kappa\sqrt{d}\log(\hat{a}_0/\epsilon)^{-1}, \sqrt[6]{d}\})$, our theorem implies a linear speedup with respect to this simple convergence bound. We would like to note, however, that the coordinate descent literature sometimes uses more refined properties of

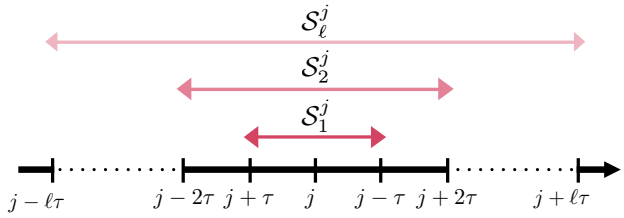


Figure 2: The set $\mathcal{S}_r^j = \{\max\{j-r\tau, 0\}, \ldots, j-1, j, j+1, \ldots, \min\{j+r\tau, T\}\}$ comprises the indices around j (including j) within $r\tau$ steps. The cardinality of such a set is $2r\tau+1$. Here, $\mathcal{S}_0^j = \{j\}$.

the function to be optimized that can lead to potentially better convergence bounds, especially in terms of function value accuracy, *i.e.*, $f(\mathbf{x}_k) - f(\mathbf{x}^*)$ (see *e.g.*, [5, 21, 30]).

We would further like to remark that between the two bounds on τ, the second one, *i.e.*, $\mathcal{O}(\sqrt[6]{d})$, is the more restrictive, as the first one is proportional—up to log factors—to the square root of the number of iterations, which is usually $\Omega(d)$. We explain in our subsequent derivation how this loose bound can be improved, but leave the tightness of the bound as an open question for future work.

4.1 Proof of Theorem 5

The analysis here is slightly more involved compared to HOGWILD!.The main technical bottleneck is to relate the decay of R_0^j with that of R_1^j, and then to exploit the sparsity of the updates for bounding R_2^j.

We start with a simple upper bound on the norm of the gradient updates. From the L-Lipschitz assumption on $\nabla f(\mathbf{x})$, we have

$$\mathbb{E}_{s_k} \|\mathbf{g}(\hat{\mathbf{x}}_k, s_k)\|^2 = d \cdot \|\nabla f(\hat{\mathbf{x}}_k)\|^2 \leq dL^2 \|\hat{\mathbf{x}}_k - \mathbf{x}^*\|^2 \leq 2dL^2 \|\mathbf{x}_j - \mathbf{x}^*\|^2 + 2dL^2 \|\mathbf{x}_j - \hat{\mathbf{x}}_k\|^2,$$

where the last inequality is due to Jensen's inequality. This yields the following result.

Lemma 6. *For any k and j we have* $\mathbb{E}\|\mathbf{g}(\hat{\mathbf{x}}_k, s_k)\|^2 \leq 2dL^2 \left(a_j + \mathbb{E}\|\mathbf{x}_j - \hat{\mathbf{x}}_k\|^2\right)$.

Let T be the total number of ASCD iterations, and let us define the set

$$\mathcal{S}_r^j = \{\max\{j - r\tau, 0\}, \ldots, j-1, j, j+1, \ldots, \min\{j + r\tau, T\}\},$$

which has cardinality at most $2r\tau + 1$ and contains all indices around j within $r\tau$ steps, as sketched in Fig. 2. Due to Assumption 3, and similar to [21], there exist variables $\sigma_{i,k}^j \in \{-1, 0, 1\}$ such that, for any index k in the set $\mathcal{S}_r^j$, we have

$$\hat{\mathbf{x}}_k - \mathbf{x}_j = \sum_{i \in \mathcal{S}_{r+1}^j} \sigma_{i,k}^j \gamma \mathbf{g}(\hat{\mathbf{x}}_i, s_i). \tag{4.1}$$

The above equation implies that the difference between a "fake" iterate at time j and the value that was read at time k can be expressed as a linear combination of any coordinate updates that occurred during the time interval defined by $\mathcal{S}_{r+1}^j$.

From Eq. (4.1) we see that $\|\hat{\mathbf{x}}_k - \mathbf{x}_j\|$, for any $k \in \mathcal{S}_r^j$, can be upper bounded in terms of the magnitude of the coordinate updates that occur in $\mathcal{S}_{r+1}^j$. Since these updates are coordinates of the true gradient, we can use their norm to bound the size of $\hat{\mathbf{x}}_k - \mathbf{x}_j$. This will be useful towards

bounding R_1^j. Moreover, Lemma 6 shows that the magnitude of the gradient steps can be upper bounded in terms of the size of the mismatches. This will in turn be useful in bounding R_0^j. The above observations are fundamental to our approach. The following lemma makes the above ideas explicit.

Lemma 7. *For any $j \in \{0, \dots, T\}$, we have*

$$\max_{k \in \mathcal{S}_r^j} \mathbb{E}\|\mathbf{g}(\hat{\mathbf{x}}_k, s_k)\|^2 \leq 2dL^2 \left(a_j + \max_{k \in \mathcal{S}_r^j} \mathbb{E}\|\hat{\mathbf{x}}_k - \mathbf{x}_j\|^2 \right) \tag{4.2}$$

$$\max_{k \in \mathcal{S}_r^j} \mathbb{E}\|\hat{\mathbf{x}}_k - \mathbf{x}_j\|^2 \leq (3\gamma\tau(r+1))^2 \max_{k \in \mathcal{S}_{r+1}^j} \mathbb{E}\|\mathbf{g}(\hat{\mathbf{x}}_k, s_k)\|. \tag{4.3}$$

Proof. The first inequality is a consequence of Lemma 6. For the second, as mentioned previously, we have $\hat{\mathbf{x}}_k - \mathbf{x}_j = \sum_{i \in \mathcal{S}_{r+1}^j} \sigma_{i,k} \gamma \mathbf{g}(\hat{\mathbf{x}}_i, s_i)$ when $k \in \mathcal{S}_r^j$. Hence,

$$\begin{aligned}
\mathbb{E}\|\hat{\mathbf{x}}_k - \mathbf{x}_j\|^2 &= \gamma^2 \cdot \mathbb{E}\left\{ \left\| \sum_{i \in \mathcal{S}_{r+1}^j} \sigma_{i,k}^j \cdot \mathbf{g}(\hat{\mathbf{x}}_i, s_i) \right\|^2 \right\} \leq \gamma^2 \cdot \mathbb{E}\left\{ |\mathcal{S}_{r+1}^j| \sum_{i \in \mathcal{S}_{r+1}^j} \|\mathbf{g}(\hat{\mathbf{x}}_i, s_i)\|^2 \right\} \\
&\leq \gamma^2 \cdot |\mathcal{S}_{r+1}^j|^2 \max_{i \in \mathcal{S}_{r+1}^j} \mathbb{E}\|\mathbf{g}(\hat{\mathbf{x}}_i, s_i)\|^2 \leq (3\gamma\tau(r+1))^2 \max_{i \in \mathcal{S}_{r+1}^j} \mathbb{E}\|\mathbf{g}(\hat{\mathbf{x}}_i, s_i)\|^2,
\end{aligned}$$

where the first inequality follows due to Jensen's inequality, and the last inequality uses the bound $|S_{r+1}^j| \leq 2(r+1)\tau + 1 \leq 3\tau(r+1)$. □

Remark 3. *The τ^2 factor in the upper bound on $\max \mathbb{E}\|\hat{\mathbf{x}}_k - \mathbf{x}_j\|^2$ in Lemma 7 might be loose. We believe that it should instead be τ, when τ is smaller than some measure of the sparsity. If the sparsity of the steps $\mathbf{g}(\hat{\mathbf{x}}_i, s_i)$ can be exploited, we suspect that the condition $\tau = \mathcal{O}(\sqrt[6]{d})$ in Theorem 5 could be improved to $\tau = \mathcal{O}(\sqrt[4]{d})$.*

Let us now define for simplicity $G_r = \max_{k \in \mathcal{S}_r^j} \mathbb{E}\|\mathbf{g}(\hat{\mathbf{x}}_k, s_k)\|^2$ and $\Delta_r = \max_{k \in \mathcal{S}_r^j} \mathbb{E}\|\hat{\mathbf{x}}_k - \mathbf{x}_j\|^2$. Observe that that all gradient norms can be bounded as

$$G_r = \max_{k \in \mathcal{S}_r^j} \mathbb{E}\|\mathbf{g}(\hat{\mathbf{x}}_k, s_k)\|^2 = d \max_{k \in \mathcal{S}_r^j} \mathbb{E}\|\nabla f(\hat{\mathbf{x}}_k)\|^2 \leq d \max_{0 \leq k \leq T} \mathbb{E}\|\nabla f(\hat{\mathbf{x}}_k)\|^2 = dM^2,$$

a property that we will use in our bounds. Observe that $R_0^j = \mathbb{E}\|\mathbf{g}(\hat{\mathbf{x}}_j, s_j)\|^2 = G_0$ and $R_1^j = \mathbb{E}\|\hat{\mathbf{x}}_j - \mathbf{x}_j\|^2 = \Delta_0$. To obtain bounds for our first two error terms, R_0^j and R_1^j, we will expand the recursive relations that are implied by Lemma 7. As shown in § B.1 of the Appendix, we obtain the following bounds.

Lemma 8. *Let $\tau \leq \frac{\kappa\sqrt{d}}{\ell}$ and set $\gamma = \frac{\theta}{6dL\kappa}$, for any $\theta \leq 1$ and $\ell \geq 1$. Then,*

$$R_0^j \leq \mathcal{O}(1) \left(dL^2 a_j + \theta^{2\ell} dM^2 \right) \text{ and } R_1^j \leq \mathcal{O}(1) \left(\theta^2 a_j + \theta^{2\ell} \frac{M^2}{L^2} \right).$$

The Cauchy-Schwartz inequality implies the bound $R_2^j \leq \sqrt{R_0^j R_1^j}$. Unfortunately this approach yields a result that can only guarantee convergence up to a factor of $\sqrt{d}$ slower than serial SCD. This happens because upper bounding the inner product $\langle \hat{\mathbf{x}}_j - \mathbf{x}_j, \mathbf{g}(\hat{\mathbf{x}}_j, s_j) \rangle$ by $\|\hat{\mathbf{x}}_j - \mathbf{x}_j\| \|\mathbf{g}(\hat{\mathbf{x}}_j, s_j)\|$ disregards the extreme sparsity of $\mathbf{g}(\hat{\mathbf{x}}_j, s_j)$. The next lemma uses a slightly more involved argument to bound R_2^j exploiting the sparsity of the gradient update. The proof can be found in Appendix B.1.

Lemma 9. *Let $\tau \leq \frac{\kappa\sqrt{d}}{\ell}$ and $\tau = \mathcal{O}(\sqrt[6]{d})$. Then, $R_2^j \leq \mathcal{O}(1) \left(\theta m a_j + \theta^{2\ell} \frac{M^2}{L\kappa} \right)$.*

4.1.1 Putting it all together

We can now plug in the upper bounds on R_0^j, R_1^j, and R_2^j in our perturbed iterate recursive formula

$$a_{j+1} \leq (1-\gamma m)\, a_j + \gamma^2 \underbrace{\mathbb{E}\|g(\hat{\mathbf{x}}_j, s_j)\|^2}_{R_0^j} + 2\gamma m \underbrace{\mathbb{E}\|\hat{\mathbf{x}}_j - \mathbf{x}_j\|^2}_{R_1^j} + 2\gamma \underbrace{\mathbb{E}\langle \hat{\mathbf{x}}_j - \mathbf{x}_j, \mathbf{g}(\hat{\mathbf{x}}_j, s_j)\rangle}_{R_2^j},$$

to find that ASCD satisfies

$$a_{j+1} \leq \Bigg(1 - \gamma m + \underbrace{\mathcal{O}(1)\Big(\gamma^2 dL^2 + \gamma m\theta^2 + \gamma\theta m\Big)}_{=r(\gamma)}\Bigg) a_j + \mathcal{O}(1)\Bigg(\underbrace{\gamma^2\theta^{2\ell} dM^2 + \gamma\theta^{2\ell}\frac{M^2}{L\kappa}}_{=\delta(\gamma)}\Bigg).$$

Observe that in the serial case of SCD the errors R_1^j and R_2^j are zero, and $R_0^j = \mathbb{E}\|g(\mathbf{x}_j, s_j)\|^2$. By applying the Lipschitz assumption on f, we get $\mathbb{E}\|g(\mathbf{x}_j, s_j)\|^2 \leq dL^2 a_j$, and obtain the simple recursive formula

$$a_{j+1} \leq (1 - \gamma m + \gamma^2 dL^2) a_j. \tag{4.4}$$

To guarantee that ASCD follows the same recursion, *i.e.*, it has the same convergence rate as the one implied by Eq. (4.4), we require that $\gamma m - r(\gamma) \geq C(\gamma m - \gamma^2 dL^2)$, where $C < 1$ is a constant. Solving for γ we get

$$\begin{aligned}
&\gamma m - C'\left(\gamma^2 dL^2 + \gamma m\theta^2 + \gamma\theta m\right) \geq C(\gamma m - \gamma^2 dL^2)\\
\Leftrightarrow &(1-C)\gamma m - (C'-C)\gamma^2 dL^2 + C'(\gamma m\theta^2 + \gamma\theta m) \geq 0\\
\Leftrightarrow &(C'-C)\gamma dL^2 \leq [(1-C) + C'(\theta^2+\theta)]m \Leftrightarrow \gamma \leq \mathcal{O}(1)\frac{\theta m}{dL^2} = \mathcal{O}(1)\frac{\theta}{d\kappa L},
\end{aligned}$$

where $C' > 1$ is some absolute constant. For $\gamma = \mathcal{O}(1)\frac{\theta}{d\kappa L}$, the $\delta(\gamma)$ term in the recursive bound becomes

$$\mathcal{O}(1)\left(\frac{\theta^2}{d^2\kappa^2 L^2}\theta^{2\ell} dM^2 + \frac{\theta}{d\kappa L}\theta^{2\ell}\frac{M^2}{L\kappa}\right) = \mathcal{O}(1)\left(\theta^{2\ell+2}\frac{M^2}{d\kappa^2 L^2} + \theta^{2\ell+1}\frac{M^2}{d\kappa^2 L^2}\right) \leq \mathcal{O}(1)\theta^{2\ell}\frac{\hat{a}_0}{d\kappa^2},$$

where we used the inequality $M^2 \leq L^2\hat{a}_0$. Hence, ASCD satisfies

$$a_{j+1} \leq (1 - \mathcal{O}(1)^{\theta}/_{d\kappa^2})\, a_j + \mathcal{O}(1)\theta^{2\ell}\frac{\hat{a}_0}{d\kappa^2} \leq (1 - \mathcal{O}(1)^{\theta}/_{d\kappa^2})^{j+1}\, a_0 + \mathcal{O}(1)\theta^{2\ell}\hat{a}_0.$$

Let us set θ to be a sufficiently small constant so that $\mathcal{O}(1)\frac{\theta}{d\kappa^2} = \frac{1}{d\kappa^2}$ and solve for ℓ such that $\mathcal{O}(1)\theta^{2\ell}\hat{a}_0 = \epsilon/2$. This gives $\ell = \mathcal{O}(1)\log\left({}^{\hat{a}_0}/_{\epsilon}\right)$. Our main theorem for ASCD now follows from solving $(1 - {}^{\mathcal{O}(1)}/_{d\kappa^2})^{j+1}\, a_0 = \epsilon/2$ for j.

5 Sparse and Asynchronous SVRG

The SVRG algorithm, presented in [11], is a variance-reduction approach to stochastic gradient descent with strong theoretical guarantees and empirical performance. In this section, we present a parallel, asynchronous and sparse variant of SVRG. We also present a convergence analysis, showing that the analysis proceeds in a nearly identical way to that of ASCD.

5.1 Serial Sparse SVRG

The original SVRG algorithm of [11] runs for a number of epochs; the per epoch iteration is given as follows:

$$\mathbf{x}_{j+1} = \mathbf{x}_j - \gamma \left(\mathbf{g}(\mathbf{x}_j, s_j) - \mathbf{g}(\mathbf{y}, s_j) + \nabla f(\mathbf{y}) \right), \tag{5.1}$$

where $\mathbf{y}$ is the last iterate of the previous epoch, and as such is updated at the end of every epoch. Here f is of the same form as in (3.1):

$$f(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} f_{e_i}(\mathbf{x}),$$

and $\mathbf{g}(\mathbf{x}, s_j) = \nabla f_{s_j}(\mathbf{x})$, with hyperedges $s_j \in \mathcal{E}$ sampled uniformly at random. As is common in the SVRG literature, we further assume that the individual f_{e_i} terms are L-smooth. The theoretical innovation in SVRG is having an SGM flavored algorithm, with small amortized cost per iteration, where the variance of the gradient estimate is smaller than that of standard SGM. For a certain selection of learning rate, epoch size, and number of iterations, [11] establishes that SVRG attains a linear rate.

Observe that when optimizing a decomposable f with sparse terms, in contrast to SGM, the SVRG iterates will be dense due to the term $\nabla f(\mathbf{y})$. From a practical perspective, when the SGM iterates are sparse—the case in several applications [1]—the cost of writing a sparse update in shared memory is significantly smaller than applying the dense gradient update term $\nabla f(\mathbf{y})$. Furthermore, these dense updates will cause significantly more memory conflicts in an asynchronous execution, amplifying the error terms in (2.5), and introducing time delays due to memory contention.

A sparse version of SVRG can be obtained by letting the support of the update be determined by that of $\mathbf{g}(\mathbf{y}, s_j)$:

$$\mathbf{x}_{j+1} = \mathbf{x}_j - \gamma \left(\mathbf{g}(\mathbf{x}_j, s_j) - \mathbf{g}(\mathbf{y}, s_j) + \mathbf{D}_{s_j} \nabla f(\mathbf{y}) \right) = \mathbf{x}_j - \gamma \mathbf{v}_j, \tag{5.2}$$

where $\mathbf{D}_{s_j} = \mathbf{P}_{s_j} \mathbf{D}$, and $\mathbf{P}_{s_j}$ is the projection on the support of s_j and $\mathbf{D} = \operatorname{diag}\left(p_1^{-1}, \ldots, p_d^{-1}\right)$ is a $d \times d$ diagonal matrix. The weight p_v is equal to the probability that index v belongs to a hyperedge sampled uniformly at random from $\mathcal{E}$. These probabilities can be computed from the right degrees of the bipartite graph shown in Fig. 1. The normalization ensures that $\mathbb{E}_{s_j} \mathbf{D}_{s_j} \nabla f(\mathbf{y}) = \nabla f(\mathbf{y})$ and thus that $\mathbb{E}\mathbf{v}_j = \nabla f(\mathbf{x}_j)$. We will establish the same upper bound on $\mathbb{E}\|\mathbf{v}_j\|^2$ for sparse SVRG as the one used in [11] to establish a linear rate of convergence for dense SVRG. As before we assume that there exists a uniform bound $M > 0$ such that $\|\mathbf{v}_j\| \leq M$.

Lemma 10. *The variance of the serial sparse SVRG procedure in* (5.2) *satisfies*

$$\mathbb{E}\|\mathbf{v}_j\|^2 \leq 2\mathbb{E}\|\mathbf{g}(\mathbf{x}_j, s_j) - \mathbf{g}(\mathbf{x}^*, s_j)\|^2 + 2\mathbb{E}\|\mathbf{g}(\mathbf{y}, s_j) - \mathbf{g}(\mathbf{x}^*, s_j)\|^2 - 2\nabla f(\mathbf{y})^\top \mathbf{D} \nabla f(\mathbf{y}).$$

Proof. By definition $\mathbf{v}_j = \mathbf{g}(\mathbf{x}_j, s_j) - \mathbf{g}(\mathbf{y}, s_j) + \mathbf{D}_{s_j} \nabla f(\mathbf{y})$. Therefore

$$\begin{aligned} \mathbb{E}\|\mathbf{v}_j\|^2 &= \mathbb{E}\|\mathbf{g}(\mathbf{x}_j, s_j) - \mathbf{g}(\mathbf{y}, s_j) + \mathbf{D}_{s_j} \nabla f(\mathbf{y})\|^2 \\ &\leq 2\mathbb{E}\|\mathbf{g}(\mathbf{x}_j, s_j) - \mathbf{g}(\mathbf{x}^*, s_j)\|^2 + 2\mathbb{E}\|\mathbf{g}(\mathbf{y}, s_j) - \mathbf{g}(\mathbf{x}^*, s_j) - \mathbf{D}_{s_j} \nabla f(\mathbf{y})\|^2. \end{aligned}$$

We expand the second term to find that

$$\begin{aligned} &\mathbb{E}\|\mathbf{g}(\mathbf{y}, s_j) - \mathbf{g}(\mathbf{x}^*, s_j) - \mathbf{D}_{s_j} \nabla f(\mathbf{y})\|^2 \\ &= \mathbb{E}\|\mathbf{g}(\mathbf{y}, s_j) - \mathbf{g}(\mathbf{x}^*, s_j)\|^2 - 2\mathbb{E}\langle \mathbf{g}(\mathbf{y}, s_j) - \mathbf{g}(\mathbf{x}^*, s_j), \mathbf{D}_{s_j} \nabla f(\mathbf{y}) \rangle + \mathbb{E}\|\mathbf{D}_{s_j} \nabla f(\mathbf{y})\|^2. \end{aligned}$$

Since $\mathbf{g}(\mathbf{x}, s_j)$ is supported on s_j for all $\mathbf{x}$, we have

$$\mathbb{E}\langle \mathbf{g}(\mathbf{y}, s_j) - \mathbf{g}(\mathbf{x}^*, s_j), \mathbf{D}_{s_j} \nabla f(\mathbf{y})\rangle = \mathbb{E}\langle \mathbf{g}(\mathbf{y}, s_j) - \mathbf{g}(\mathbf{x}^*, s_j), \mathbf{D}\nabla f(\mathbf{y})\rangle = \nabla f(\mathbf{y})^\top \mathbf{D} \nabla f(\mathbf{y}),$$

where the second equality follows by the property of iterated expectations. The conclusion follows because $\mathbb{E}\|\mathbf{D}_{s_j}\nabla f(\mathbf{y})\|^2 = \nabla f(\mathbf{y})^\top \mathbf{D}\nabla f(\mathbf{y})$. □

Observe that the last term in the variance bound is a non-negative quadratic form, hence we can drop it and obtain the same variance bound as the one obtained in [11] for dense SVRG. This directly leads to the following corollary.

Corollary 11. *Sparse SVRG admits the same convergence rate upper bound as that of the SVRG of [11].*

We note that usually the convergence rates for SVRG are obtained for function value differences. However, since our perturbed iterate framework of § 2 is based on iterate differences, we re-derive a convergence bound for iterates.

Lemma 12. *Let the step size be $\gamma = \frac{1}{4L\kappa}$ and the length of an epoch be $8\kappa^2$. Then, $\mathbb{E}\|\mathbf{y}_k - \mathbf{x}^*\|^2 \leq 0.75^k \cdot \mathbb{E}\|\mathbf{y}_0 - \mathbf{x}^*\|^2$, where $\mathbf{y}_k$ is the iterate at the end of the k-th epoch.*

Proof. We bound the distance to the optimum after one epoch of length $8\kappa^2$:

$$\begin{aligned}
\mathbb{E}\|\mathbf{x}_{j+1} - \mathbf{x}^*\|^2 &= \mathbb{E}\|\mathbf{x}_j - \mathbf{x}^*\|^2 - 2\gamma\mathbb{E}\langle \mathbf{x}_j - \mathbf{x}^*, \mathbf{v}_j\rangle + \gamma^2\mathbb{E}\|\mathbf{v}_j\|^2 \\
&\leq \mathbb{E}\|\mathbf{x}_j - \mathbf{x}^*\|^2 - 2\gamma\mathbb{E}\langle \mathbf{x}_j - \mathbf{x}^*, \nabla f(\mathbf{x}_j)\rangle + 2\gamma^2\mathbb{E}\|\mathbf{g}(\mathbf{x}_j, s_j) - \mathbf{g}(\mathbf{x}^*, s_j)\|^2 \\
&\quad + 2\gamma^2\mathbb{E}\|\mathbf{g}(\mathbf{y}, s_j) - \mathbf{g}(\mathbf{x}^*, s_j)\|^2 \\
&\leq \mathbb{E}\|\mathbf{x}_j - \mathbf{x}^*\|^2 - 2\gamma\mathbb{E}\langle \mathbf{x}_j - \mathbf{x}^*, \nabla f(\mathbf{x}_j)\rangle + 2\gamma^2L^2\mathbb{E}\|\mathbf{x}_j - \mathbf{x}^*\|^2 + 2\gamma^2L^2\mathbb{E}\|\mathbf{y} - \mathbf{x}^*\|^2 \\
&\leq (1 - 2\gamma m + 2\gamma^2L^2)\mathbb{E}\|\mathbf{x}_j - \mathbf{x}^*\|^2 + 2\gamma^2L^2\mathbb{E}\|\mathbf{y} - \mathbf{x}^*\|^2.
\end{aligned}$$

The first inequality follows from Lemma 10 and an application of iterated expectations to obtain $\mathbb{E}\langle \mathbf{x}_j - \mathbf{x}^*, \mathbf{v}_j\rangle = \mathbb{E}\langle \mathbf{x}_j - \mathbf{x}^*, \nabla f(\mathbf{x}_j)\rangle$. The second inequality follows from the smoothness of $\mathbf{g}(\mathbf{x}, s_j)$, and the third inequality follows since f is m-strongly convex.

We can rewrite the inequality as $a_{j+1} \leq (1 - 2\gamma m + 2\gamma^2L^2)a_j + 2\gamma^2L^2a_0$, because by construction $\mathbf{y} = \mathbf{x}_0$. Let $\gamma = \frac{1}{4L\kappa}$. Then, $1 - 2\gamma m + 2\gamma^2L^2 \leq 1 - \frac{1}{4\kappa^2}$ and

$$\sum_{i=0}^{j}(1 - 2\gamma m + 2\gamma^2L^2)^i \leq \sum_{i=0}^{j}(1 - 1/4\kappa^2)^i \leq \sum_{i=0}^{\infty}(1 - 1/4\kappa^2)^i = 4\kappa^2,$$

since $\frac{1}{4\kappa^2} \leq \frac{1}{4}$. Therefore

$$\begin{aligned}
a_{j+1} &\leq (1 - 1/4\kappa^2)\, a_j + 2\gamma^2L^2a_0 \leq (1 - 1/4\kappa^2)^{j+1}\, a_0 + \sum_{i=0}^{j-1}(1 - 1/4\kappa^2)^i \cdot 2\gamma^2L^2a_0 \\
&\leq (1 - 1/4\kappa^2)^{j+1}\, a_0 + 4\kappa^2\gamma^2L^2a_0 = \left[(1 - 1/4\kappa^2)^{j+1} + 1/4\right] a_0.
\end{aligned}$$

Setting the length of an epoch to be $j = 2 \cdot (4\kappa^2)$ gives us $a_{j+1} \leq (1/2 + 1/4) \cdot a_0 = 0.75 \cdot a_0$, and the conclusion follows. □

We thus obtain the following convergence rate result:

Theorem 13. *Sparse SVRG, with step size $\gamma = \mathcal{O}(1)\frac{1}{L\kappa}$ and epoch size $S = \mathcal{O}(1)\kappa^2$, reaches accuracy $\mathbb{E}\|\mathbf{y}_E - \mathbf{x}^*\|^2 \leq \epsilon$ after $E = \mathcal{O}(1)\log(a_0/\epsilon)$ epochs, where $\mathbf{y}_E$ is the last iterate of the final epoch, and $a_0 = \|\mathbf{x}_0 - \mathbf{x}^*\|^2$ is the initial distance squared to the optimum.*

5.2 KroMagnon: Asynchronous Parallel Sparse SVRG

We now present an asynchronous implementation of sparse SVRG. This implementation, which we refer to as KroMagnon, is given in Algorithm 3.

Algorithm 3 KroMagnon

```
x = y = x_0
for Epoch = 1 : E do
    Compute in parallel z = ∇f(y)
    while number of sampled hyperedges ≤ S do in parallel
        sample a random hyperedge s
        [x̂]_s = an inconsistent read of the shared variable [x]_s
        [u]_s = −γ · (∇f_s([x]_s) − ∇f_s([y]_s) − D_s z)
        for v ∈ s do
            [x]_v = [x]_v + [u]_v                                   // atomic write
        end for
    end while
    y = x
end for
```

Let $\mathbf{v}(\hat{\mathbf{x}}_j, s_j) = \mathbf{g}(\hat{\mathbf{x}}_j, s_j) - \mathbf{g}(\mathbf{y}, s_j) + \mathbf{D}_{s_j}\nabla f(\mathbf{y})$ be the noisy gradient update vector. Then, after processing a total of T hyperedges, the shared memory contains:

$$\underbrace{\overbrace{\mathbf{x}_0 - \gamma\mathbf{v}(\hat{\mathbf{x}}_0, s_0)}^{\mathbf{x}_1} - \ldots - \gamma\mathbf{v}(\hat{\mathbf{x}}_{T-1}, s_{T-1})}_{\mathbf{x}_T}. \tag{5.3}$$

We now define the perturbed iterates as $\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma\mathbf{v}(\hat{\mathbf{x}}_i, s_i)$ for $i = 0, 1, ..., T-1$, where s_i is the i-th uniformly sampled hyperedge. Since $\mathbb{E}\mathbf{v}(\hat{\mathbf{x}}_j, s_j) = \nabla f(\mathbf{x}_j)$, KroMagnon also satisfies recursion (2.5):

$$a_{j+1} \leq (1-\gamma m)a_j + \gamma^2 \underbrace{\mathbb{E}\|\mathbf{v}(\hat{\mathbf{x}}_j, s_j)\|^2}_{R_0^j} + 2\gamma m \underbrace{\mathbb{E}\|\hat{\mathbf{x}}_j - \mathbf{x}_j\|^2}_{R_1^j} + 2\gamma \underbrace{\mathbb{E}\langle \hat{\mathbf{x}}_j - \mathbf{x}_j, \mathbf{v}(\hat{\mathbf{x}}_j, s_j)\rangle}_{R_2^j}.$$

To prove the convergence of KroMagnon we follow the line of reasoning presented in the previous section. Most of the arguments used here come from a straightforward generalization of the analysis of ASCD. The main result of this section is given below.

Theorem 14. *Let the maximum number of samples that can overlap in time with a single sample be bounded as*

$$\tau = \mathcal{O}\left(\min\left\{\frac{\kappa}{\log\left(\frac{M^2}{L^2\epsilon}\right)}, \sqrt[6]{\frac{n}{\overline{\Delta}_C}}\right\}\right).$$

Then, KroMagnon*, with step size $\gamma = \mathcal{O}(1)\frac{1}{L\kappa}$ and epoch size $S = \mathcal{O}(1)\kappa^2$, attains $\mathbb{E}\|\mathbf{y}_E - \mathbf{x}^*\|^2 \leq \epsilon$ after $E = \mathcal{O}(1)\log(a_0/\epsilon)$ epochs, where $\mathbf{y}_E$ is the last iterate of the final epoch, and $a_0 = \|\mathbf{x}_0 - \mathbf{x}^*\|^2$ is the initial distance squared to the optimum.*

We would like to note that the total number of iterations in the above bound is—up to a universal constant—the same as that of serial sparse SVRG as presented in Theorem 13. Again, as with HOGWILD! and ASCD, this implies a linear speedup.

Similar to our ASCD analysis, we remark that between the two bounds on τ, the second one is the more restrictive. The first one is, up to logarithmic factors, equal to the square root of the total number of iterations per epoch; we expect that the size of the epoch is proportional to n, the number of function terms (or data points). This suggests that the first bound is proportional to $\tilde{\mathcal{O}}(\sqrt{n})$ for most reasonable applications. Moreover, the second bound is certainly loose; we argue that it can be tightened using a more refined analysis.

5.3 Proof of Theorem 14

It is easy to see that due to Lemma 10 we get the following bound on the norm of the gradient estimate.

Lemma 15. *For any k and j we have*

$$\mathbb{E}\left\|\mathbf{v}(\hat{\mathbf{x}}_k, s_k)\right\|^2 \leq 4L^2\left(a_j + a_0 + \mathbb{E}\|\mathbf{x}_j - \hat{\mathbf{x}}_k\|^2\right). \tag{5.4}$$

Proof. Due to Lemma 10 we have $\mathbb{E}\|\mathbf{v}(\hat{\mathbf{x}}_j, s_j)\|^2 \leq 2L^2\mathbb{E}\|\hat{\mathbf{x}}_j - \mathbf{x}^*\|^2 + 2L^2\mathbb{E}\|\mathbf{y} - \mathbf{x}^*\|^2$. Then, using the fact that $\mathbf{y} = \mathbf{x}_0$ and applying the triangle inequality, we obtain the result. □

The set $\mathcal{S}_r^j$ is defined as in the previous section: $\mathcal{S}_r^j = \{\max\{j - r\tau, 0\}, \ldots, j-1, j, j+1, \ldots, \min\{j + r\tau, T\}\}$, and has cardinality at most $2r\tau + 1$. By Assumption 3, there exist diagonal sign matrices $\mathbf{S}_i^j$ with diagonal entries in $\{-1, 0, 1\}$ such that

$$\hat{\mathbf{x}}_k - \mathbf{x}_j = \gamma \sum_{i \in \mathcal{S}_{\ell+1}^j} \mathbf{S}_i^j \mathbf{v}(\hat{\mathbf{x}}_i, s_i). \tag{5.5}$$

This leads to the following lemma.

Lemma 16. *If $G_r = \max_{k \in \mathcal{S}_r^j} \mathbb{E}\left\|\mathbf{v}(\hat{\mathbf{x}}_k, s_k)\right\|^2$ and $\Delta_r = \max_{k \in \mathcal{S}_r^j} \mathbb{E}\|\hat{\mathbf{x}}_k - \mathbf{x}_j\|^2$,*

$$G_r \leq 4L^2\left(a_j + a_0 + \Delta_r\right) \text{ and } \Delta_r \leq (3\gamma\tau(r+1))^2 G_{r+1}. \tag{5.6}$$

Proof. The proof for the bound on Δ_r is identical to the proof of Lemma 7. We then use Lemma 15 to bound $\mathbb{E}\left\|\mathbf{v}(\hat{\mathbf{x}}_k, s_k)\right\|^2$. □

As explained in the remark after Lemma 7, it should be possible to improve τ^2 to τ in the upper bound on Δ_r. Doing so would improve the condition $\tau = \mathcal{O}(\sqrt[6]{n/\Delta_C})$ of Theorem 14 to $\tau = \mathcal{O}(\sqrt[4]{n/\Delta_C})$. One possible approach to this problem can be found in § B.2.2 of the Appendix.

We can now obtain bounds on the errors due to asynchrony. The proofs for the following two lemmas can be found in Appendix B.2.

Lemma 17. *Suppose $\tau \leq \frac{\kappa}{\ell}$ and $\gamma = \frac{\theta}{12L\kappa}$. Then the error terms R_0^j and R_1^j of* KROMAGNON *satisfy the following inequalities:*

$$R_0^j \leq \mathcal{O}(1)\left(L^2(a_j + a_0) + \theta^{2\ell}M^2\right) \quad \text{and} \quad R_1^j \leq \mathcal{O}(1)\left(\theta^2(a_j + a_0) + \theta^{2\ell}M^2/L^2\right).$$

Similarly to the ASCD derivations, we obtain the following bound for R_2^j.

Lemma 18. *Suppose $\tau \leq \frac{\kappa}{\ell}$ and $\tau = \mathcal{O}\left(\sqrt[6]{\frac{n}{\Delta_C}}\right)$, and let $\gamma = \frac{\theta}{12L\kappa}$. Then,*

$$R_2^j \leq \mathcal{O}(1)\left(\theta \cdot m \cdot (a_j + a_0) + \theta^{2\ell}\frac{M^2}{L\kappa}\right).$$

Problem	Dataset	# data	# features	sparsity
Linear regression	Synthetic	3M	10K	20
Logistic regression	Synthetic	3M	10K	20
	rcv1 [31] from [32]	$\approx$ 700K	$\approx$ 42K	$\approx$ 73
	url [33] from [32]	$\approx$ 3.2M	$\approx$ 2.4M	$\approx$ 116
Vertex cover[2]	eswiki-2013 [34–36]	$\approx$ 970K	$\approx$ 23M	$\approx$ 24
	wordassociation-2011 [34–36]	$\approx$ 10.6K	$\approx$ 72K	$\approx$ 7

Table 1: The problems and data sets used in our experimental evaluation. We test KroMagnon, dense SVRG, and Hogwild! on three different tasks: linear regression, logistic regression, and vertex cover. We test the algorithms on sparse data sets, of various sizes and feature dimensions.

5.3.1 Putting it all together

After plugging in the upper bounds on R_0^j, R_1^j, and R_2^j in the main recursion satisfied by KroMagnon, we find that:

$$a_{j+1} \leq \left(1 - \gamma m + \mathcal{O}(1)\left(\gamma^2 L^2 + \gamma m \theta^2 + \gamma \theta m\right)\right) a_j + \\ + \mathcal{O}(1)\left(\gamma^2 L^2 + \gamma m \theta^2 + \gamma \theta m\right) a_0 + \gamma^2 \mathcal{O}(1) \theta^{2\ell} M^2 + \gamma \mathcal{O}(1) \theta^{2\ell} \frac{M^2}{L\kappa}.$$

If we set $\gamma = \mathcal{O}(1)\frac{\theta}{L\kappa}$, *i.e.*, the same step size as serial sparse SVRG (Theorem 13), then the above becomes

$$a_{j+1} \leq \left(1 - \mathcal{O}(1)\frac{\theta}{\kappa^2}\right) a_j + \mathcal{O}(1)\frac{\theta^2}{\kappa^2} a_0 + \theta^{2\ell+1}\mathcal{O}(1)\frac{M^2}{L^2\kappa^2} \\ \leq \left[\left(1 - \mathcal{O}(1)\frac{\theta}{\kappa^2}\right)^{j+1} + \mathcal{O}(1)\theta\right] a_0 + \theta^{2\ell}\mathcal{O}(1)\frac{M^2}{L^2}.$$

We choose $\theta = \mathcal{O}(1) \leq 1/2$ to be a sufficiently small constant, so that the term $\mathcal{O}(1)\theta$ in the brackets above is at most 0.5. Then we can choose $j = \mathcal{O}(1)\kappa^2$ so that the entire coefficient in the brackets is at most 0.75. Finally, we set $\ell = \mathcal{O}(1)\log\left(M^2/L^2\epsilon\right)$, so that the last term is smaller than $\epsilon/8$. Let $\mathbf{y}_k$ be the iterate after the k-th epoch and $A_k = \mathbb{E}\|\mathbf{y}_k - \mathbf{x}^*\|^2$. Therefore, KroMagnon satisfies the recursion

$$A_{k+1} \leq 0.75 \cdot A_k + \frac{\epsilon}{8} \leq (0.75)^{k+1} A_0 + \frac{\epsilon}{2}.$$

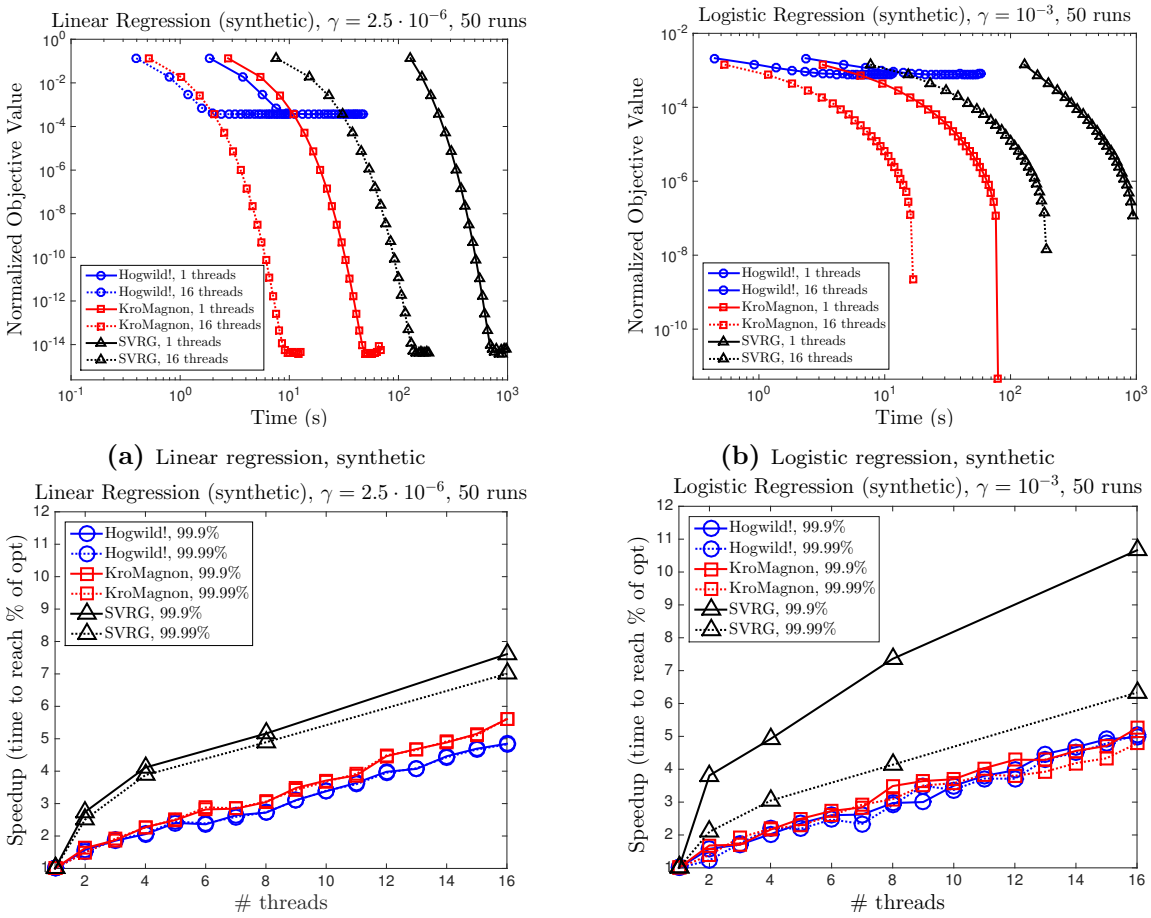
This implies that $\mathcal{O}(1)\log(a_0/\epsilon)$ epochs are sufficient to reach ϵ accuracy, where a_0 is $\|\mathbf{x}_0 - \mathbf{x}^*\|^2$ the initial distance squared to the optimum.

6 Empirical Evaluation of KroMagnon

In this section we evaluate KroMagnon empirically. Our two goals are to demonstrate that (1) KroMagnon is faster than dense SVRG, and (2) KroMagnon has speedups comparable to those of Hogwild!. We implemented Hogwild!, asynchronous dense SVRG, and KroMagnon in Scala, and tested them on the problems and datasets listed in Table 1. Each algorithm was run for 50 epochs, using up to 16 threads. For the SVRG algorithms, we recompute $\mathbf{y}$ and the full gradient $\nabla f(\mathbf{y})$ every two epochs. We normalize the objective values such that the objective at

[2]Following [5], we optimize a quadratic penalty relaxation for vertex cover $\min_{x\in[0,1]^{|V|+|E|}} \sum_{v\in V} x_v + \frac{\beta}{2}\sum_{(u,v)\in E}(x_u + x_v - x_{u,v} - 1)^2 + \frac{1}{2\beta}\sum_{v\in V} x_v^2 + \sum_{e\in E} x_e^2$.

the initial starting point has a value of one, and the minimum attained across all algorithms and epochs has a value of zero. Experiments were conducted on a Linux machine with 2 Intel Xeon Processor E5-2670 (2.60GHz, eight cores each) with 250Gb memory.



(a) Linear regression, synthetic

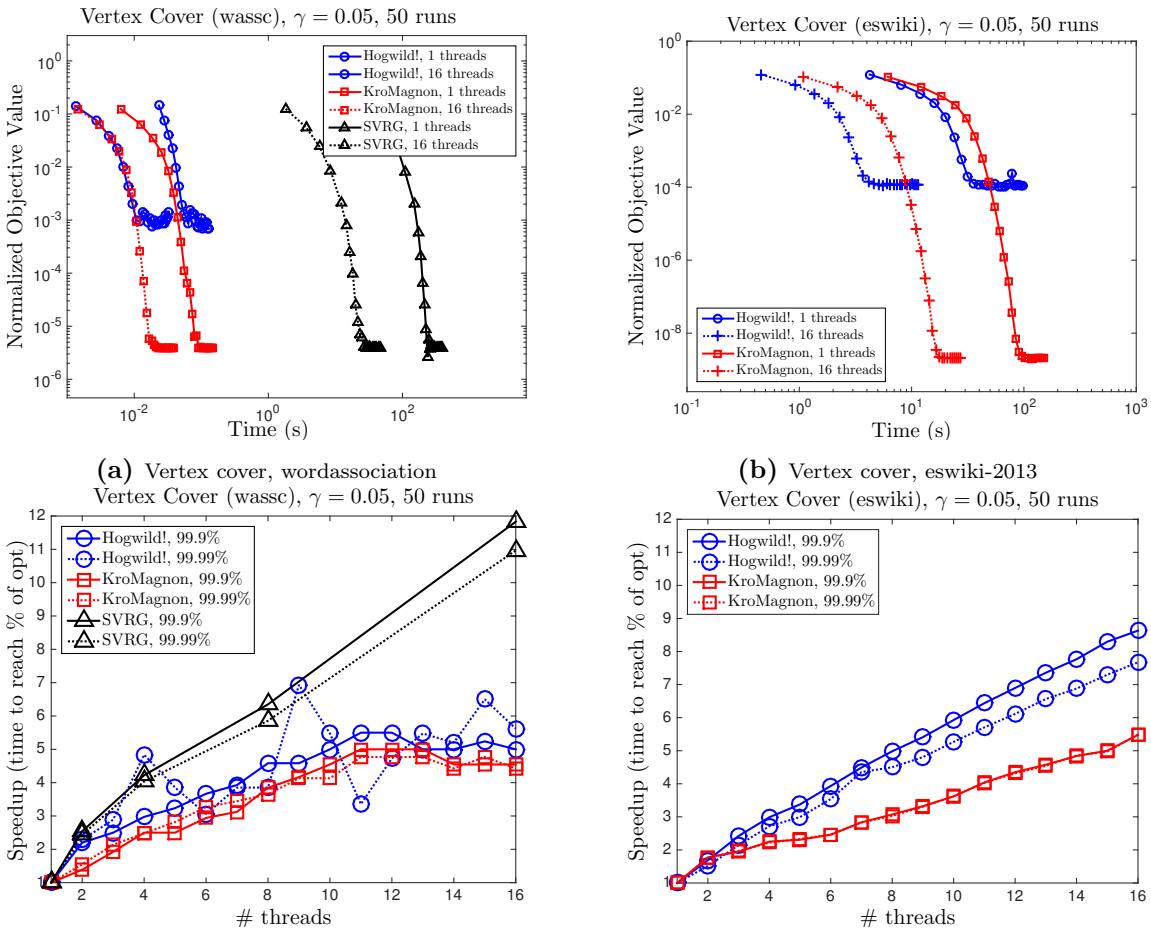
(b) Logistic regression, synthetic

(c) Linear regression, synthetic

(d) Logistic regression, synthetic

Figure 3: Linear and logistic regression on synthetic data. In subfigures (a) and (b) we plot the convergence with respect to normalized objective value as a function of wall-clock time, and in (c) and (d) the speedup with respect to the number of threads. The above experiments are all for linear and logistic regression problems on synthetic data, in which we have 3 million data points, each with 10K features, and each data point with 20 nonzero entries. We observe that KroMagnon is significantly faster than parallel and dense SVRG, while they both can attain better objective values compared to constant step-size Hogwild!. Moreover, we observe that the speedup gains of Hogwild! and KroMagnon are scaling reasonably well for up to 16 threads.

Comparison with dense SVRG We were unable to run dense SVRG on the url and eswiki-2013 datasets due to the large number of features. Figures 3a, 3b, and 5a show that KroMagnon is one-two orders of magnitude faster than dense SVRG. In fact, running dense SVRG on 16 threads is slower than KroMagnon on a single thread. Moreover, as seen in Fig. 4a, KroMagnon on 16 threads can be up to four orders of magnitude faster than serial dense SVRG. Both dense SVRG and KroMagnon attain similar optima.



(a) Vertex cover, wordassociation

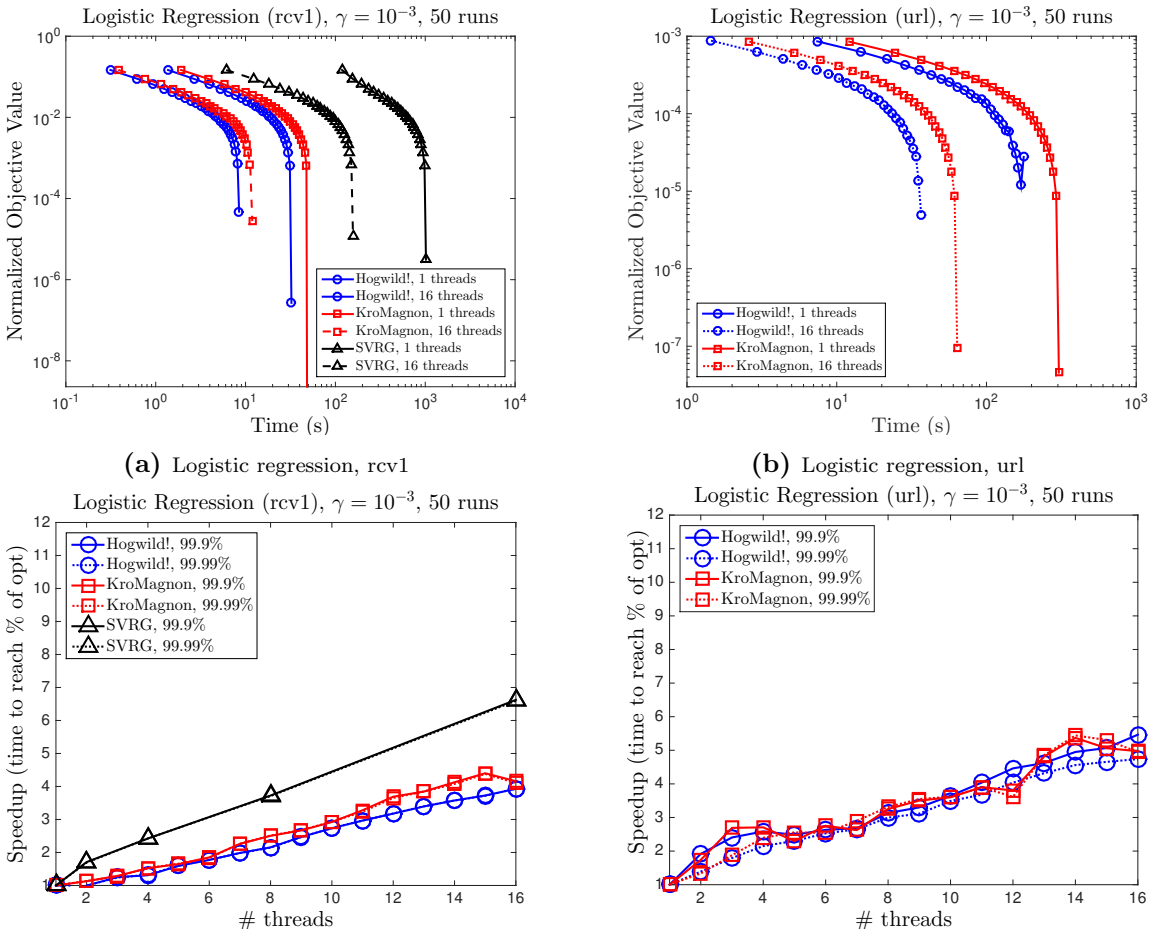
(b) Vertex cover, eswiki-2013

(c) Vertex cover, wordassociation

(d) Vertex cover, eswiki-2013

Figure 4: Vertex cover on the wordassociation-2011 and eswiki-2013 datasets. Subfigure (a) shows the convergence of the algorithms on wordassociation-2011, a small graph with less than 11,000 vertices. KroMagnon on a single thread is 3-4 orders of magnitude faster than desnse SVRG on this dataset. Convergence of KroMagnon and Hogwild! on the eswiki-2013 dataset is shown in subfigure (b); we were unable to run dense SVRG on this larger graph. Subfigures (c) and (d) show the speedups of the algorithms on the two datasets. In subfigure (c), both Hogwild! and KroMagnon exhibit poorer speedups than dense SVRG because of the rapid conve on the smaller wordassociation-2011 dataset. In subfigure (d) we observe that Hogwild! achieves a speedup of up to 8x and KroMagnon up to 5x.

Speedups We measured the time each algorithm takes to achieve 99.9% and 99.99% of the minimum achieved by that algorithm. Speedups are computed relative to the runtime of the algorithm on a single thread. Although the speedup of KroMagnon varies across datasets, we find that KroMagnon has comparable speedups with Hogwild! on all datasets, as shown in Figure 3c, 3d, 4c, 4d, 5c, 5d. We further observe that dense SVRG has better speedup scaling. This happens because the per iteration complexity of Hogwild! and KroMagnon is significantly cheaper to the extent that the additional overhead associated with having extra threads leads to some speedup loss; this is not the case for dense SVRG as the per iteration cost is higher.



(a) Logistic regression, rcv1

(b) Logistic regression, url

(c) Logistic regression, rcv1

(d) Logistic regression, url

Figure 5: Logistic regression on the rcv1 and url datasets. Subfigure (a) shows the convergence of the algorithms on the rcv1 dataset. For a given objective value, KroMagnon is 1-2 orders of magnitude faster than dense SVRG. On the larger url dataset (subfigure (b)), we were unable to run dense SVRG. Note that some of the effect of asynchrony can be observed in these experiments: the objective values obtained by KroMagnon, Hogwild!, and dense SVRG are slightly different on 1 thread compared to 16 threads. Speedups of the algorithms are shown in subfigures (c) and (d)–KroMagnon has a slightly better speedup than Hogwild! on rcv1, and the same speedup on url.

7 Conclusions and Open Problems

We have introduced a novel framework for analyzing parallel asynchronous stochastic gradient optimization algorithms. The main advantage of our framework is that it is straightforward to apply to a range of first-order stochastic algorithms, while it involves elementary derivations. Moreover, in our analysis we lift, or relax, many of the assumptions made in prior art, *e.g.*, we do not assume consistent reads, and we analyze full stochastic gradient updates. We use our framework to analyze Hogwild! and ASCD, and further introduce and analyze KroMagnon, a new asynchronous sparse SVRG algorithm.

We conclude with some open problems:

1. It would be interesting to obtain tighter bounds for the convergence of function values of the algorithms presented. How do the "errors" due to asynchrony influence the convergence

rate of function values? In this case the number of iterations required to reach a target accuracy should scale with the condition number of the objective, not its square. Moreover, the literature on stochastic coordinate descent establishes convergence results in terms of coordinate-wise Lipschitz constants—a more refined smoothness quantity than the full-function smoothness. It would be worthwhile to know if our framework can be adapted to take these parameters into account.

2. Our perturbed iterates framework relies fundamentally on the strong convexity assumption. However, asynchronous algorithms are known to perform well on non-strongly convex (and even nonconvex) objectives. Can we generalize our framework to simply convex, or smooth functions? Under what assumptions, or simple families of functions, can we show convergence for nonconvex problems?

3. As previously explained, we believe that the upper bounds on τ—the proxy for the number of cores—in our ASCD and KroMagnon analyses are amenable to improvements. It is an open problem to explore the extent of such improvements.

4. Our analysis offers sensible upper bounds only in the presence of sparsity. It seems, however, that to obtain speedup results for Hogwild!, it is only necessary to have small correlation between randomly sampled gradients. In what practical setups do randomly selected gradients have sufficiently small correlation? Does that immediately imply linear speedups in the same way that update sparsity does?

5. In this work we analyzed three similar stochastic first-order methods. It is an open problem to apply our framework and provide an elementary analysis for a greater variety of stochastic gradient type optimization algorithms, such as AdaGrad-type schemes (similar to [6]), or stochastic dual coordinate methods (similar to [8]).

6. Capturing the effects of asynchrony as noise on the algorithmic input seems to be applicable to settings beyond stochastic optimization. As shown recently for a combinatorial graph problem, a similar viewpoint enables the analysis of an asynchronous graph clustering algorithm [**?**]. It is an interesting endeavor to explore the extent to which a perturbed iterate viewpoint is suitable for analyzing general asynchronous iterative algorithms.

References

[1] Feng Niu, Benjamin Recht, Christopher Re, and Stephen Wright. Hogwild: a lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

[2] Ben Recht, Christopher Re, Joel Tropp, and Victor Bittorf. Factoring nonnegative matrices with linear programs. In *Advances in Neural Information Processing Systems*, pages 1214–1222, 2012.

[3] Yong Zhuang, Wei-Sheng Chin, Yu-Chin Juan, and Chih-Jen Lin. A fast parallel SGD for matrix factorization in shared memory systems. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 249–256, 2013.

[4] Hyokun Yun, Hsiang-Fu Yu, Cho-Jui Hsieh, SVN Vishwanathan, and Inderjit Dhillon. Nomad: Non-locking, stochastic multi-machine algorithm for asynchronous and decentralized matrix completion. *Proceedings of the VLDB Endowment*, 7(11):975–986, 2014.

[5] Ji Liu, Steve Wright, Christopher Re, Victor Bittorf, and Srikrishna Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. In *Proceedings of the 31st International Conference on Machine Learning*, pages 469–477, 2014.

[6] John Duchi, Michael I Jordan, and Brendan McMahan. Estimation, optimization, and parallelism when data is sparse. In *Advances in Neural Information Processing Systems*, pages 2832–2840, 2013.

[7] Yu-Xiang Wang, Veeranjaneyulu Sadhanala, Wei Dai, Willie Neiswanger, Suvrit Sra, and Eric P Xing. Asynchronous parallel block-coordinate Frank-Wolfe. *arXiv preprint arXiv:1409.6086*, 2014.

[8] Cho-Jui Hsieh, Hsiang-Fu Yu, and Inderjit S Dhillon. Passcode: parallel asynchronous stochastic dual coordinate descent. *arXiv preprint arXiv:1504.01365*, 2015.

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

[10] Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. Project Adam: building an efficient and scalable deep learning training system. In *11th USENIX Symposium on Operating Systems Design and Implementation*, pages 571–582, 2014.

[11] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

[12] Daniel Chazan and Willard Miranker. Chaotic relaxation. *Linear algebra and its applications*, 2(2):199–222, 1969.

[13] Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

[14] John N Tsitsiklis, Dimitri P Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812, 1986.

[15] Martin Zinkevich, John Langford, and Alex J Smola. Slow learners are fast. In *Advances in Neural Information Processing Systems*, pages 2331–2339, 2009.

[16] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 2595–2603, 2010.

[17] Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77, 2011.

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

[19] Peter Richtárik and Martin Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1-2):433–484, 2016.

[20] Martin Jaggi, Virginia Smith, Martin Takác, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 3068–3076, 2014.

[21] Ji Liu and Stephen J Wright. Asynchronous stochastic coordinate descent: parallelism and convergence properties. *SIAM Journal on Optimization*, 25(1):351–376, 2015.

[22] Mingyi Hong. A distributed, asynchronous and incremental algorithm for nonconvex optimization: an ADMM based approach. *arXiv preprint arXiv:1412.6058*, 2014.

[23] Hamid Reza Feyzmahdavian, Arda Aytekin, and Mikael Johansson. An asynchronous mini-batch algorithm for regularized stochastic optimization. In *IEEE Conference on Decision and Control*, pages 1384–1389. IEEE, 2015.

[24] Ji Liu, Stephen J Wright, and Srikrishna Sridhar. An asynchronous parallel randomized Kaczmarz algorithm. *arXiv preprint arXiv:1401.4780*, 2014.

[25] Haim Avron, Alex Druinsky, and Anshul Gupta. Revisiting asynchronous linear solvers: provable convergence rate through randomization. In *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pages 198–207. IEEE, 2014.

[26] Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm. In *Advances in Neural Information Processing Systems*, pages 1017–1025, 2014.

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

[28] Zhimin Peng, Yangyang Xu, Ming Yan, and Wotao Yin. Arock: an algorithmic framework for asynchronous parallel coordinate updates. *arXiv preprint arXiv:1506.02396*, 2015.

[29] Elad Hazan and Satyen Kale. Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization. *The Journal of Machine Learning Research*, 15(1):2489–2512, 2014.

[30] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015.

[31] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. RCV1: a new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

[32] LIBSVM Data: Classification (binary class). `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`.

[33] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Identifying suspicious URLs: an application of large-scale online learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 681–688, 2009.

[34] Paolo Boldi and Sebastiano Vigna. The WebGraph framework I: compression techniques. In *Proceedings of the 13th international conference on World Wide Web*, pages 595–602, 2004.

[35] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *Proceedings of the 20th international conference on World Wide Web*, pages 587–596, 2011.

[36] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. Ubicrawler: a scalable fully distributed web crawler. *Software: Practice & Experience*, 34(8):711–726, 2004.

A Removing the Independence Assumption

Our main analysis for HOGWILD! relied on the independence between $\hat{\mathbf{x}}_i$ and s_i (Assumption 1). Here, we show how to lift this assumption, conditional on the fact that each of the f_{e_i} terms is at least L_{s} smooth. Observe that the following is no longer true: $\mathbb{E}\langle \hat{\mathbf{x}}_i - \mathbf{x}^*, \mathbf{g}(\hat{\mathbf{x}}_i, s_i)\rangle = \mathbb{E}\langle \hat{\mathbf{x}}_i - \mathbf{x}^*, \nabla f(\hat{\mathbf{x}}_i)\rangle$ since we cannot use iterated expectations, precisely due to the lack of independence of the samples and the read variables. However, $\overline{\mathbf{x}}_i$, defined in § 3, is still independent of s_i. Therefore, we expand our derivation in (2.3) in the following way:

$$\begin{aligned} a_{j+1} \leq a_j - 2\gamma\mathbb{E}\langle \overline{\mathbf{x}}_i - \mathbf{x}^*, \mathbf{g}(\overline{\mathbf{x}}_j, s_j)\rangle + \gamma^2\mathbb{E}\|g(\hat{\mathbf{x}}_j, \xi_j)\|^2 + 2\gamma\mathbb{E}\langle \overline{\mathbf{x}}_j - \mathbf{x}_j, \mathbf{g}(\overline{\mathbf{x}}_j, \xi_j)\rangle \\ + 2\gamma\mathbb{E}\langle \mathbf{x}_j - \mathbf{x}^*, \mathbf{g}(\overline{\mathbf{x}}_j, s_j) - \mathbf{g}(\hat{\mathbf{x}}_j, s_j)\rangle. \end{aligned}$$

Since $\overline{\mathbf{x}}_j$ and s_j are independent by construction, we use iterated expectations to get: $\mathbb{E}\langle \overline{\mathbf{x}}_i - \mathbf{x}^*, \mathbf{g}(\overline{\mathbf{x}}_j, s_j)\rangle = \mathbb{E}\langle \overline{\mathbf{x}}_i - \mathbf{x}^*, \nabla f(\overline{\mathbf{x}}_j)\rangle$. As before, the strong convexity of f and the triangle inequality imply that $\langle \overline{\mathbf{x}}_j - \mathbf{x}^*, \nabla f(\overline{\mathbf{x}}_j)\rangle \geq \frac{m}{2}\|\mathbf{x}_j - \mathbf{x}^*\|^2 - m\|\overline{\mathbf{x}}_j - \mathbf{x}_j\|^2$. Putting everything together we get the following recursion for the sequence a_j.

$$\begin{aligned} a_{j+1} \leq (1-\gamma m)a_j + \gamma^2 \underbrace{\mathbb{E}\|g(\hat{\mathbf{x}}_j, \xi_j)\|^2}_{R_0^j} + 2\gamma m \underbrace{\mathbb{E}\|\overline{\mathbf{x}}_j - \mathbf{x}_j\|^2}_{R_1^j} + 2\gamma \underbrace{\mathbb{E}\langle \overline{\mathbf{x}}_j - \mathbf{x}_j, \mathbf{g}(\overline{\mathbf{x}}_j, \xi_j)\rangle}_{R_2^j} \\ + 2\gamma \underbrace{\mathbb{E}\langle \mathbf{x}_j - \mathbf{x}^*, \mathbf{g}(\overline{\mathbf{x}}_j, s_j) - \mathbf{g}(\hat{\mathbf{x}}_j, s_j)\rangle}_{R_3^j}. \end{aligned}$$

The reader can verify that although R_1^j and R_2^j are defined now in terms of $\overline{\mathbf{x}}_j$, the upper bounds derived in § 3 still hold. We bound R_3^j as follows

$$\mathbb{E}\langle \mathbf{x}_j - \mathbf{x}^*, \mathbf{g}(\overline{\mathbf{x}}_j, s_j) - \mathbf{g}(\hat{\mathbf{x}}_j, s_j)\rangle \leq \mathbb{E}\|\mathbf{x}_j - \mathbf{x}^*\|\|\mathbf{g}(\overline{\mathbf{x}}_j, s_j) - \mathbf{g}(\hat{\mathbf{x}}_j, s_j)\| \leq \frac{ma_j}{4} + \frac{L_{\mathrm{s}}^2}{m}\mathbb{E}\|\overline{\mathbf{x}}_j - \hat{\mathbf{x}}_j\|^2,$$

where the last inequality follows by the smoothness of the gradient steps and the arithmetic-geometric mean inequality. Therefore

$$\begin{aligned} a_{j+1} \leq \left(1 - \frac{\gamma m}{2}\right) a_j + \gamma^2 \underbrace{\mathbb{E}\|g(\hat{\mathbf{x}}_j, \xi_j)\|^2}_{R_0^j} + 2\gamma m \underbrace{\mathbb{E}\|\overline{\mathbf{x}}_j - \mathbf{x}_j\|^2}_{R_1^j} + 2\gamma \underbrace{\mathbb{E}\langle \overline{\mathbf{x}}_j - \mathbf{x}_j, \mathbf{g}(\overline{\mathbf{x}}_j, \xi_j)\rangle}_{R_2^j} \\ + 2\gamma\frac{L_{\mathrm{s}}^2}{m} \underbrace{\mathbb{E}\|\overline{\mathbf{x}}_j - \hat{\mathbf{x}}_j\|^2}_{R_4^j}. \end{aligned}$$

Since we can upper bound R_4^j by the same bound derived for R_3^j, we obtain the following convergence result for HOGWILD!:

Theorem 19. *If the number of samples that overlap in time with a single sample during the execution of* HOGWILD! *is bounded as* $\tau = \mathcal{O}\left(\min\left\{n/\overline{\Delta}_C, M^2/\epsilon L^2\right\}\right)$, *then* HOGWILD! *with step-size* $\gamma = \mathcal{O}(1)\frac{\epsilon m}{M^2}$, *reaches an accuracy of* $\mathbb{E}\|\mathbf{x}_k - \mathbf{x}^*\|^2 \leq \epsilon$ *after* $T \geq \mathcal{O}(1)\frac{M^2\log(a_0/\epsilon)}{\epsilon m^2}$ *iterations.*

The only difference between this result and the one in our main section is that we can guarantee speedup for a smaller range of τ. Similar ideas could be applied to the analysis of ASCD and KroMagnon.

B Omitted Proofs

B.1 ASCD

B.1.1 Bounding R_0^j and R_1^j

Lemma 8. *Let $\tau \leq \frac{\kappa\sqrt{d}}{\ell}$ and set $\gamma = \frac{\theta}{6dL\kappa}$, for any $\theta \leq 1$ and $\ell \geq 1$. Then,*

$$R_0^j \leq \mathcal{O}(1)\left(dL^2 a_j + \theta^{2\ell} dM^2\right) \text{ and } R_1^j \leq \mathcal{O}(1)\left(\theta^2 a_j + \theta^{2\ell}\frac{M^2}{L^2}\right).$$

Proof. Let $A = 2dL^2 a_j$, $B = 2dL^2$ and $C = (\gamma\tau)^2$. Then, we can rewrite the bounds of Lemma 7 as $G_r \leq A + B \cdot \Delta_r$ and $\Delta_r \leq 3^2(r+1)^2 \cdot C \cdot G_{r+1}$, which implies $G_r \leq A + 3^2(r+1)^2 BC \cdot G_{r+1}$. We can now upper bound $R_0^j = G_0$, by applying the previous inequality ℓ times. If we expand the formulas, we get

$$R_0^j = G_0 \leq A\sum_{i=0}^{\ell-1}(3^i \cdot i!)^2 (BC)^i + (3^\ell \cdot \ell!)^2 (BC)^\ell G_\ell. \tag{B.1}$$

Since $\gamma = \frac{\theta}{6dL\kappa}$ and $\tau \leq \frac{\kappa\sqrt{d}}{\ell}$ (the choice of τ is made so that the sum in (B.1) is significantly small), we have $BC = 2dL^2\gamma^2\tau^2 \leq 2dL^2 \frac{\theta^2}{6^2 d^2 L^2 \kappa^2}\frac{\kappa^2 d}{\ell^2} \leq \frac{1}{2\cdot 3^2\cdot \ell^2}$. Using the upper bound $k! \leq k^k$ on each term of the sum (B.1), and plugging in the upper bound on BC, we get

$$\sum_{i=0}^{\ell-1}(3^i \cdot i!)^2 (BC)^i \leq \sum_{i=0}^{\ell-1}\frac{(i!)^2}{2^i \ell^{2i}} \leq \sum_{i=0}^{\ell-1}\frac{1}{2^i}\left(i/\ell\right)^{2i} \leq 2.$$

Similarly, we obtain the following upper bound on the last term of Eq. B.1 $(3^\ell \cdot \ell!)^2 (BC)^\ell \leq 2^{-\ell}\theta^{2\ell}$. Finally, $G_\ell \leq dM^2$, and combining the above gives us $R_0^j \leq \mathcal{O}(1)\left(dL^2 a_j + \theta^{2\ell} dM^2\right)$.

We can now bound R_1^j. By definition $R_1^j = \Delta_0$, and from Lemma 7 we have $\Delta_0 \leq 3^2 \cdot C \cdot G_1$. We can bound G_1 similarly to G_0 as

$$G_1 \leq A\sum_{i=0}^{\ell-1}(3^i \cdot (i+1)!)^2 (BC)^i + (3^\ell \cdot (\ell+1)!)^2 (BC)^\ell G_{\ell+1}.$$

As before $BC \leq \frac{\theta^2}{2\cdot 3^2 \ell^2}$. Since $(i+1)! \leq 2\ell^i$ for any $0 \leq i \leq \ell$, it follows that $\sum_{i=0}^{\ell-1}(3^i \cdot (i+1)!)^2 (BC)^i \leq \mathcal{O}(1)$, and $(3^\ell \cdot (\ell+1)!)^2 (BC)^\ell \leq \mathcal{O}(1)\theta^{2\ell}$. Therefore, because $G_{\ell+1} \leq dM^2$, we obtain $G_1 \leq \mathcal{O}(1)(dL^2 a_j + \theta^{2\ell} dM^2)$. Since $C = (\gamma\tau)^2 \leq \frac{\theta^2}{dL^2}$, it follows that $R_1 \leq \mathcal{O}(1)\left(\theta^2 a_j + \theta^{2\ell}(M/L)^2\right)$. ∎

B.1.2 Bounding R_2^j

Lemma 9. *Let $\tau \leq \frac{\kappa\sqrt{d}}{\ell}$ and $\tau = \mathcal{O}(\sqrt[6]{d})$. Then, $R_2^j \leq \mathcal{O}(1)\left(\theta m a_j + \theta^{2\ell}\frac{M^2}{L\kappa}\right)$.*

Proof. From (4.1) we can upper bound R_2^j as follows.

$$R_2^j = \mathbb{E}\langle \hat{\mathbf{x}}_j - \mathbf{x}_j, \mathbf{g}(\hat{\mathbf{x}}_j, s_j)\rangle \leq \gamma \cdot \sum_{\substack{i=j-\tau \\ i\neq j}}^{j+\tau} \mathbb{E}\|\mathbf{g}(\hat{\mathbf{x}}_i, s_i)\| \cdot \|\mathbf{g}(\hat{\mathbf{x}}_j, s_j)\| \cdot \mathbb{1}(s_i = s_j). \tag{B.2}$$

The random variable $\mathbb{1}(s_i = s_j)$ encodes the sparsity of the gradient steps. To take advantage of this sparsity we use smoothness to replace the iterates $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$, by $\hat{\mathbf{x}}_{j-3\tau}$. The latter iterate is independent of both s_i and s_j by our assumption that no more than τ coordinates can be updated while a core is processing a single coordinate. This independence will allows us to "untangle" the expectation of $\mathbb{1}(s_i = s_j)$ from the inner products in the above sum, which will result in a significantly improved bound on R_2^j compared to applying Cauchy-Schwartz directly on it.

For clarity, we note that when $j < 3\tau$, we have $\hat{\mathbf{x}}_{j-3\tau} = \mathbf{x}_0$. From the L-Lipschitz assumption on the gradient $\nabla f(\mathbf{x})$, we get the following bounds

$$\begin{aligned}\|\mathbf{g}(\hat{\mathbf{x}}_j, s_j)\| &\leq \|\mathbf{g}(\hat{\mathbf{x}}_{j-3\tau}, s_j)\| + dL\|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_j\| \\ \|\mathbf{g}(\hat{\mathbf{x}}_i, s_i)\| &\leq \|\mathbf{g}(\hat{\mathbf{x}}_{j-3\tau}, s_i)\| + dL\|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_i\|.\end{aligned}$$

Then, the expectation of a term $\|\mathbf{g}(\hat{\mathbf{x}}_i, s_i)\| \cdot \|\mathbf{g}(\hat{\mathbf{x}}_j, s_j)\| \cdot \mathbb{1}(s_i = s_j)$ in the sum (B.2) is upper bounded by

$$\begin{aligned}\mathbb{E}\Big\{\Big(&\|\mathbf{g}(\hat{\mathbf{x}}_{j-3\tau}, s_i)\|\ \|\mathbf{g}(\hat{\mathbf{x}}_{j-3\tau}, s_j)\| + (dL)^2\|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_i\|\ \|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_j\| \\ &+dL\|\mathbf{g}(\hat{\mathbf{x}}_{j-3\tau}, s_j)\|\ \|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_i\| + dL\|\mathbf{g}(\hat{\mathbf{x}}_{j-3\tau}, s_i)\|\ \|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_j\|\Big) \cdot \mathbb{1}(s_i = s_j)\Big\}.\end{aligned}$$

We first bound the second term using Cauchy-Schwartz and the property of iterated expectation, to exploit the expectation of the $\mathbb{1}(s_i = s_j)$ term

$$\begin{aligned}&\mathbb{E}\{\|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_i\| \cdot \|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_j\| \cdot \mathbb{1}(s_i = s_j)\} \\ &\qquad \leq \sqrt{\mathbb{E}\{\|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_i\|^2\} \cdot \mathbb{E}\{\|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_j\|^2 \cdot \mathbb{1}(s_i = s_j)\}} \\ &\qquad = \sqrt{\mathbb{E}\{\|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_i\|^2\} \cdot \mathbb{E}_{\sim s_j}\{\|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_j\|^2 \cdot \mathbb{E}_{s_j}\{\mathbb{1}(s_i = s_j)\}\}} \\ &\qquad = \sqrt{\frac{1}{d}}\sqrt{\mathbb{E}\{\|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_i\|^2\} \cdot \mathbb{E}\{\|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_j\|^2\}} \\ &\qquad \leq \mathcal{O}(1)\sqrt{\frac{1}{d}} \cdot \gamma^2\tau^2 \underbrace{\max_{k\in\mathcal{S}_4^j} \mathbb{E}\{\|\mathbf{g}(\hat{\mathbf{x}}_k, s_k)\|^2\}}_{G_4},\end{aligned}$$

where the first equality follows due to $\hat{\mathbf{x}}_j$ being independent of s_j; hence the expectation with respect to s_j can be applied to the indicator function. The last inequality follows from our arguments in the proof of Lemma 7 because both mismatches $\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_j$ can be written as linear combinations of gradient steps indexed by $\mathcal{S}_4^j$ as in (4.1). Similarly the third term satisfies the inequality

$$\begin{aligned}&\mathbb{E}\{\|\mathbf{g}(\hat{\mathbf{x}}_{j-3\tau}, s_j)\| \cdot \|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_i\| \cdot \mathbb{1}(s_i = s_j)\} \\ &\qquad \leq \sqrt{\mathbb{E}\{\|\mathbf{g}(\hat{\mathbf{x}}_{j-3\tau}, s_j)\|^2\} \cdot \mathbb{E}\{\|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_i\|^2 \cdot \mathbb{1}(s_i = s_j)\}} \\ &\qquad = \mathcal{O}(1)\sqrt{\frac{1}{d}} \cdot \gamma\tau G_4.\end{aligned}$$

The same bound applies for the fourth term $\mathbb{E}\{\|\mathbf{g}(\hat{\mathbf{x}}_{j-3\tau}, s_i)\| \cdot \|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_j\| \cdot \mathbb{1}(s_i = s_j)\}$, while the first term can be easily bounded as

$$\mathbb{E}\{\|\mathbf{g}(\hat{\mathbf{x}}_{j-3\tau}, s_j)\| \cdot \|\mathbf{g}(\hat{\mathbf{x}}_{j-3\tau}, s_i)\| \cdot \mathbb{1}(s_i = s_j)\} \leq \sqrt{\frac{1}{d}} G_4.$$

Putting all pieces together, and using the prescribed value of $\gamma = \frac{\theta}{6dL\kappa}$, we have that

$$R_2 \leq \mathcal{O}(1)\sqrt{\frac{1}{d}}(\gamma\tau)\left(1 + dL\gamma\tau + (dL\gamma\tau)^2\right) G_4 \leq \mathcal{O}(1)\sqrt{\frac{1}{d}} \cdot \gamma \cdot \tau^3 \cdot G_4.$$

The first inequality follows because we are summing over 2τ terms in (B.2). To see why the second inequality is true, note that $dL\gamma \leq \frac{\theta}{6\kappa} \leq 1$ (it is always true that the condition number $\kappa \geq 1$). Therefore $1 + dL\gamma\tau + (dL\gamma\tau)^2 \leq 1 + \tau + \tau^2 \leq 3\tau^2$. As in the proof of Lemma 8, we can bound G_4 by

$$\begin{aligned} G_4 &\leq \mathcal{O}(1)A\sum_{i=0}^{\ell-1}(3^i \cdot (i+4)!)^2(BC)^i + \mathcal{O}(1)(3^\ell \cdot (\ell+4)!)^2(BC)^\ell G_{\ell+4} \\ &\leq \mathcal{O}(1)(dL^2 a_j + \theta^{2\ell} dM^2). \end{aligned}$$

The result follows assuming $\tau = \mathcal{O}(\sqrt[6]{d})$ and $\gamma = \frac{\theta}{6dL\kappa}$. □

Remark 4. *We believe that if we use the same bounding technique that we applied for R_2^j on R_0^j and R_1^j, then we can significantly improve the restrictive bound on τ.*

B.2 KroMagnon

B.2.1 Bounding R_0^j and R_1^j

Lemma 17. *Suppose $\tau \leq \frac{\kappa}{\ell}$ and $\gamma = \frac{\theta}{12L\kappa}$. Then the error terms R_0^j and R_1^j of* KroMagnon *satisfy the following inequalities:*

$$R_0^j \leq \mathcal{O}(1)\left(L^2(a_j + a_0) + \theta^{2\ell}M^2\right) \quad \textit{and} \quad R_1^j \leq \mathcal{O}(1)\left(\theta^2(a_j + a_0) + \theta^{2\ell}M^2/L^2\right).$$

Proof. Let $A = 4L^2(a_j + a_0)$, $B = 4L^2$, and $C = (\gamma\tau)^2$. Then, the inequalities derived above can be rewritten as

$$G_r \leq A + B\Delta_r \text{ and } \Delta_r \leq 3^2(r+1)^2 C G_{r+1}. \tag{B.3}$$

If we expand the formulas, we get for R_0^j the following upper bound

$$R_0^j = G_0 \leq A\sum_{i=0}^{\ell-1}(3^i \cdot i!)^2(BC)^i + (3^\ell \cdot \ell!)^2(BC)^\ell G_\ell.$$

We chose $\gamma = \frac{\theta}{12L\kappa}$ and assumed that $\tau \leq \kappa/\ell$, where $\kappa = \frac{L}{m}$ is the condition number and $\theta \leq 1$. We chose γ to be proportional to the step-size of the serial SVRG, and the assumption on τ is made so that the sum in the above inequality is significantly is significantly small. Then,

$$(3^i \cdot i!)^2(BC)^i \leq (3i)^{2i}\left(4L^2\frac{\theta^2}{4^2 \cdot 3^2 L^2\kappa^2}\frac{\kappa^2}{\ell^2}\right)^i \leq \frac{\theta^2}{4^i}\left(\frac{i}{\ell}\right)^{2i}$$

and hence

$$\sum_{i=0}^{l-1}(3^i \cdot i!)^2(BC)^i \leq \sum_{i=0}^{\infty} 2^{-2i} \leq 2.$$

As in the previous sections we assume a uniform upper bound $M > 0$ on the size of the gradient steps: $\max_j \mathbb{E}\|\mathbf{v}(\hat{\mathbf{x}}_j, s_j)\|^2 = M^2$. Therefore

$$R_0^j = G_0 \leq \mathcal{O}(1)\left(L^2(a_j + a_0) + \theta^{2\ell}M^2\right).$$

After an analogous derivation one can see that

$$R_1^j = \Delta_0 \leq \mathcal{O}(1)\left(\theta^2(a_j + a_0) + \theta^{2\ell}\frac{M^2}{L^2}\right),$$

and thus we obtain the result. □

B.2.2 Bounding R_2^j

Lemma 18. *Suppose $\tau \leq \frac{\kappa}{\ell}$ and $\tau = \mathcal{O}\left(\sqrt[6]{\frac{n}{\overline{\Delta}_C}}\right)$, and let $\gamma = \frac{\theta}{12L\kappa}$. Then,*

$$R_2^j \leq \mathcal{O}(1)\left(\theta \cdot m \cdot (a_j + a_0) + \theta^{2\ell}\frac{M^2}{L\kappa}\right).$$

Proof. From (5.5) we can upper bound R_2^j as follows.

$$R_2^j = \mathbb{E}\langle \hat{\mathbf{x}}_j - \mathbf{x}_j, \mathbf{v}(\hat{\mathbf{x}}_j, s_j)\rangle \leq \gamma \cdot \sum_{i=j-\tau, i\neq j}^{j+\tau} \mathbb{E}\|\mathbf{v}(\hat{\mathbf{x}}_i, s_i)\| \cdot \|\mathbf{v}(\hat{\mathbf{x}}_j, s_j)\| \cdot \mathbb{1}(s_i \cap s_j \neq \emptyset). \tag{B.4}$$

The random variable $\mathbb{1}(s_i \cap s_j \neq \emptyset)$ encodes the sparsity of the gradient steps. As in the proof of Lemma 9, we replace $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$ in the above sum by $\hat{\mathbf{x}}_{j-3\tau}$. When $j < 3\tau$ we define $\hat{\mathbf{x}}_{j-3\tau} = \mathbf{x}_0$. Since f_{e_i} are L-smooth, we have

$$\|\mathbf{v}(\hat{\mathbf{x}}_j, s_j)\| \leq \|\mathbf{v}(\hat{\mathbf{x}}_{j-3\tau}, s_j)\| + L\|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_j\|$$
$$\|\mathbf{v}(\hat{\mathbf{x}}_i, s_i)\| \leq \|\mathbf{v}(\hat{\mathbf{x}}_{j-3\tau}, s_i)\| + L\|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_i\|.$$

Then, the expectation of a term $\|\mathbf{v}(\hat{\mathbf{x}}_i, s_i)\| \cdot \|\mathbf{v}(\hat{\mathbf{x}}_j, s_j)\| \cdot \mathbb{1}(s_i \cap s_j)$ in the sum (B.2) is upper bounded by

$$\mathbb{E}\Bigg\{\Bigg(\|\mathbf{v}(\hat{\mathbf{x}}_{j-3\tau}, s_i)\|\ \|\mathbf{v}(\hat{\mathbf{x}}_{j-3\tau}, s_j)\| + L^2\|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_i\|\ \|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_j\|$$
$$+L\|\mathbf{v}(\hat{\mathbf{x}}_{j-3\tau}, s_j)\|\ \|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_i\| + L\|\mathbf{v}(\hat{\mathbf{x}}_{j-3\tau}, s_i)\|\ \|\hat{\mathbf{x}}_{j-3\tau} - \hat{\mathbf{x}}_j\|\Bigg) \cdot \mathbb{1}(s_i \cap s_j)\Bigg\}.$$

Then, since $\mathbb{E}\mathbb{1}(s_i \cap s_j \neq \emptyset) \leq \frac{2\overline{\Delta}_{\mathrm{C}}}{n}$ (recall that $\overline{\Delta}_{\mathrm{C}}$ is the average conflict degrees), R_2^j can be shown to satisfy the inequality

$$R_2^j \leq \mathcal{O}(1)\sqrt{\frac{\overline{\Delta}_{\mathrm{C}}}{n}}\gamma\tau^3\left(L^2(a_j + a_0) + \theta^{2\ell}M^2\right)$$

as in the proof of Lemma 9. The conclusion follows because $\tau = \mathcal{O}\left(\sqrt[6]{\frac{n}{\overline{\Delta}_{\mathrm{C}}}}\right)$ and $\gamma = \frac{\theta}{12L\kappa} = \frac{m\theta}{12L^2}$. □

Remark 5. *Similar to ASCD, by using the same bounding technique of R_2^j on R_0^j and R_1^j, we should significantly improve the restrictive bound on τ in the convergence result of* KroMagnon.