Journal of Machine Learning Research 19 (2018) 1-68 Submitted 11/17; Revised 7/18; Published 12/18

# Improved Asynchronous Parallel Optimization Analysis for Stochastic Incremental Methods

**Rémi Leblond** REMI.LEBLOND@INRIA.FR
*INRIA - Sierra Project-Team*
*École Normale Supérieure, Paris*

**Fabian Pedregosa** F@BIANP.NET
*INRIA - Sierra Project-Team*
*École Normale Supérieure, Paris*

**Simon Lacoste-Julien** SLACOSTE@IRO.UMONTREAL.CA
*Department of CS & OR (DIRO)*
*Université de Montréal, Montréal*

**Editor:** Tong Zhang

## Abstract

As data sets continue to increase in size and multi-core computer architectures are developed, asynchronous parallel optimization algorithms become more and more essential to the field of Machine Learning. Unfortunately, conducting the theoretical analysis asynchronous methods is difficult, notably due to the introduction of delay and inconsistency in inherently sequential algorithms. Handling these issues often requires resorting to simplifying but unrealistic assumptions. Through a novel perspective, we revisit and clarify a subtle but important technical issue present in a large fraction of the recent convergence rate proofs for asynchronous parallel optimization algorithms, and propose a simplification of the recently introduced "perturbed iterate" framework that resolves it. We demonstrate the usefulness of our new framework by analyzing three distinct asynchronous parallel incremental optimization algorithms: HOGWILD (asynchronous SGD), KROMAGNON (asynchronous SVRG) and ASAGA, a novel asynchronous parallel version of the incremental gradient algorithm SAGA that enjoys fast linear convergence rates. We are able to both remove problematic assumptions and obtain better theoretical results. Notably, we prove that ASAGA and KROMAGNON can obtain a theoretical linear speedup on multi-core systems even without sparsity assumptions. We present results of an implementation on a 40-core architecture illustrating the practical speedups as well as the hardware overhead. Finally, we investigate the overlap constant, an ill-understood but central quantity for the theoretical analysis of asynchronous parallel algorithms. We find that it encompasses much more complexity than suggested in previous work, and often is order-of-magnitude bigger than traditionally thought.

**Keywords:** optimization, machine learning, large scale, asynchronous parallel, sparsity

## 1. Introduction

We consider the unconstrained optimization problem of minimizing a *finite sum* of smooth convex functions:

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

where each $f_i$ is assumed to be convex with $L$-Lipschitz continuous gradient, $f$ is $\mu$-strongly convex and $n$ is large (for example, the number of data points in a regularized empirical risk minimization setting). We define a condition number for this problem as $\kappa := {}^{L}/_{\mu}$, as is standard in the finite sum literature.[1] A flurry of randomized incremental algorithms (which at each iteration select $i$ at random and process only one gradient $f_i'$) have recently been proposed to solve (1) with a fast[2] linear convergence rate, such as Sag (Le Roux et al., 2012), Sdca (Shalev-Shwartz and Zhang, 2013), Svrg (Johnson and Zhang, 2013) and Saga (Defazio et al., 2014). These algorithms can be interpreted as variance reduced versions of the stochastic gradient descent (Sgd) algorithm, and they have demonstrated both theoretical and practical improvements over Sgd (for the *finite sum* optimization problem 1).

In order to take advantage of the multi-core architecture of modern computers, the aforementioned optimization algorithms need to be adapted to the asynchronous parallel setting, where multiple threads work concurrently. Much work has been devoted recently in proposing and analyzing asynchronous parallel variants of algorithms such as Sgd (Niu et al., 2011), Sdca (Hsieh et al., 2015) and Svrg (Reddi et al., 2015; Mania et al., 2017; Zhao and Li, 2016). Among the incremental gradient algorithms with fast linear convergence rates that can optimize (1) in its general form, only Svrg had had an asynchronous parallel version proposed.[3] No such adaptation had been attempted for Saga until Leblond et al. (2017), even though one could argue that it is a more natural candidate as, contrarily to Svrg, it is not epoch-based and thus has no synchronization barriers at all. The present paper is an extended journal version of the conference paper from Leblond et al. (2017).

The usual frameworks for asynchronous analysis are quite intricate (see Section 2.2) and thus require strong simplifying assumptions. They are not well suited to the study of complex algorithms such as Saga. We therefore base our investigation on the newly proposed "perturbed iterate" framework introduced in Mania et al. (2017), which we also improve upon in order to properly analyze Saga. Deriving a framework in which the analysis of Saga is possible enables us to highlight the deficiencies of previous frameworks and to define a better alternative. Our new approach is not limited to Saga but can be used to investigate other algorithms and improve their existing bounds.

***Contributions.*** In Section 2, we propose a simplification of the "perturbed iterate" framework from Mania et al. (2017) as a basis for our asynchronous convergence analysis. At the

1. Since we have assumed that each individual $f_i$ is $L$-smooth, $f$ itself is $L$-smooth – but its smoothness constant $L_f$ could be much smaller. While the more classical condition number is $\kappa_b := {}^{L_f}/_{\mu}$, our rates are in terms of this bigger ${}^{L}/_{\mu}$ in this paper.
2. Their complexity in terms of gradient evaluations to reach an accuracy of $\epsilon$ is $O((n + \kappa) \log({}^{1}/_{\epsilon}))$, in contrast to $O(n\kappa_b \log({}^{1}/_{\epsilon}))$ for batch gradient descent in the worst case.
3. We note that Sdca requires the knowledge of an explicit $\mu$-strongly convex regularizer in (1), whereas Sag / Saga are adaptive to any local strong convexity of $f$ (Schmidt et al., 2016; Defazio et al., 2014). The variant of Svrg from Hofmann et al. (2015) is also adaptive (we review this variant in Section 4.1).

same time, through a novel perspective, we revisit and clarify a technical problem present in a large fraction of the literature on randomized asynchronous parallel algorithms (with the exception of Mania et al. 2017, which also highlights this issue): namely, they all assume unbiased gradient estimates, an assumption that is *inconsistent* with their proof technique without further unpractical synchronization assumptions.

In Section 3.1, we present a novel sparse variant of Saga that is more adapted to the parallel setting than the original Saga algorithm. In Section 3.2, we present Asaga, a lock-free asynchronous parallel version of Sparse Saga that does not require consistent read or write operations. We give a tailored convergence analysis for Asaga. Our main result states that Asaga obtains the same geometric convergence rate per update as Saga when the overlap bound $\tau$ (which scales with the number of cores) satisfies $\tau \leq \mathcal{O}(n)$ and $\tau \leq \mathcal{O}(\frac{1}{\sqrt{\Delta}} \max\{1, \frac{n}{\kappa}\})$, where $\Delta \leq 1$ is a measure of the sparsity of the problem. This notably implies that a linear speedup is theoretically possible even without sparsity in the well-conditioned regime where $n \gg \kappa$. This result is in contrast to previous analysis which always required some sparsity assumptions.

In Section 4, we revisit the asynchronous variant of Svrg from Mania et al. (2017), Kromagnon, while removing their gradient bound assumption (which was inconsistent with the strongly convex setting).[4] We prove that the algorithm enjoys the same fast rates of convergence as Svrg under similar conditions as Asaga – whereas the original paper only provided analysis for slower rates (in both the sequential and the asynchronous case), and thus less meaningful speedup results.

In Section 5, in order to show that our improved "after read" perturbed iterate framework can be used to revisit the analysis of other optimization routines with correct proofs that do not assume homogeneous computation, we provide the analysis of the Hogwild algorithm first introduced in Niu et al. (2011). Our framework allows us to remove the classic gradient bound assumption and to prove speedups in more realistic settings.

In Section 6, we provide a practical implementation of Asaga and illustrate its performance on a 40-core architecture, showing improvements compared to asynchronous variants of Svrg and Sgd. We also present experiments on the overlap bound $\tau$, showing that it encompasses much more complexity than suggested in previous work.

***Related Work.*** The seminal textbook of Bertsekas and Tsitsiklis (1989) provides most of the foundational work for parallel and distributed optimization algorithms. An asynchronous variant of Sgd with constant step size called Hogwild was presented by Niu et al. (2011); part of their framework of analysis was re-used and inspired most of the recent literature on asynchronous parallel optimization algorithms with convergence rates, including asynchronous variants of coordinate descent (Liu et al., 2015), Sdca (Hsieh et al., 2015), Sgd for non-convex problems (De Sa et al., 2015; Lian et al., 2015), Sgd for stochastic optimization (Duchi et al., 2015) and Svrg (Reddi et al., 2015; Zhao and Li, 2016). These papers make use of

4. Although the authors mention that this gradient bound assumption can be enforced through the use of a thresholding operator, they do not explain how to handle the interplay between this non-linear operator and the asynchrony of the algorithm. Their theoretical analysis relies on the linearity of the operations (e.g. to derive (Mania et al., 2017, Eq. (2.6))), and thus this claim is not currently supported by theory (note that a strongly convex function over an unbounded domain always has unbounded gradients).

an unbiased gradient assumption that is not consistent with the proof technique, and thus suffers from technical problems[5] that we highlight in Section 2.2.

The "perturbed iterate" framework presented in Mania et al. (2017) is to the best of our knowledge the only one that does not suffer from this problem, and our convergence analysis builds heavily from their approach, while simplifying it. In particular, the authors assumed that $f$ was both strongly convex and had a bound on the gradient, two *inconsistent* assumptions in the unconstrained setting that they analyzed. We overcome these difficulties by using tighter inequalities that remove the requirement of a bound on the gradient. We also propose a more convenient way to label the iterates (see Section 2.2). The sparse version of Saga that we propose is also inspired from the sparse version of Svrg proposed by Mania et al. (2017).

Reddi et al. (2015) presents a hybrid algorithm called Hsag that includes Saga and Svrg as special cases. Their asynchronous analysis is epoch-based though, and thus does not handle a fully asynchronous version of Saga as we do. Moreover, they require consistent reads and do not propose an efficient sparse implementation for Saga, in contrast to Asaga.

Pan et al. (2016) proposes a black box mini-batch algorithm to parallelize Sgd-like methods while maintaining serial equivalence through smart update partitioning. When the data set is sparse enough, they obtain speedups over "Hogwild" implementations of Svrg and Saga.[6] However, these "Hogwild" implementations appear to be quite suboptimal, as they do not leverage data set sparsity efficiently: they try to adapt the "lazy updates" trick from Schmidt et al. (2016) to the asynchronous parallel setting – which as discussed in Appendix E is extremely difficult – and end up making several approximations which severely penalize the performance of the algorithms. In particular, they have to use much smaller step sizes than in the sequential version, which makes for worse results.

Pedregosa et al. (2017) extend the Asaga algorithm presented in Section 3.2 to the proximal setting.

***Notation.*** We denote by $\mathbb{E}$ a full expectation with respect to all the randomness in the system, and by $\mathbf{E}$ the *conditional* expectation of a random $i$ (the index of the factor $f_i$ chosen in Sgd and other algorithms), conditioned on all the past, where "past" will be clear from the context. $[x]_v$ represents the coordinate $v$ of the vector $x \in \mathbb{R}^d$. For *sequential* algorithms, $x^+$ is the updated parameter vector after one algorithm iteration.

## 2. Revisiting the Perturbed Iterate Framework for Asynchronous Analysis

As most recent parallel optimization contributions, we use a similar hardware model to Niu et al. (2011). We consider multiple cores which all have read and write access to a shared memory. The cores update a central parameter vector in an asynchronous and lock-free fashion. Unlike Niu et al. (2011), we *do not* assume that the vector reads are consistent: multiple cores can read and write different coordinates of the shared vector at the same time. This also implies that a full vector read for a core might not correspond to any consistent state in the shared memory at any specific point in time.

5. Except (Duchi et al., 2015) that can be easily fixed by incrementing their global counter *before* sampling.
6. By "Hogwild", the authors mean asynchronous parallel variants where cores independently run the sequential update rule.

### 2.1. Perturbed Iterate Framework

We first review the "perturbed iterate" framework recently introduced by Mania et al. (2017) which will form the basis of our analysis. In the sequential setting, stochastic gradient descent and its variants can be characterized by the following update rule:

$$x_{t+1} = x_t - \gamma g(x_t, i_t)\,, \tag{2}$$

where $i_t$ is a random variable independent from $x_t$ and we have the unbiasedness condition $\mathbf{E} g(x_t, i_t) = f'(x_t)$ (recall that $\mathbf{E}$ is the relevant-past conditional expectation with respect to $i_t$).

Unfortunately, in the parallel setting, we manipulate stale, inconsistent reads of shared parameters and thus we do not have such a straightforward relationship. Instead, Mania et al. (2017) proposed to distinguish $\hat{x}_t$, the actual value read by a core to compute an update, from $x_t$, a "virtual iterate" that we can analyze and is *defined* by the update equation:

$$x_{t+1} := x_t - \gamma g(\hat{x}_t, i_t)\,. \tag{3}$$

We can thus interpret $\hat{x}_t$ as a noisy (perturbed) version of $x_t$ due to the effect of asynchrony.

We formalize the precise meaning of $x_t$ and $\hat{x}_t$ in the next section. We first note that all references mentioned in the related work section that analyzed asynchronous parallel randomized algorithms assumed that the following unbiasedness condition holds:

$$[\text{unbiasedness condition}] \quad \mathbf{E}[g(\hat{x}_t, i_t)|\hat{x}_t] = f'(\hat{x}_t).\,^{7} \tag{4}$$

This condition is at the heart of most convergence proofs for randomized optimization methods.[8] Mania et al. (2017) correctly pointed out that most of the literature thus made the often implicit assumption that $i_t$ is independent of $\hat{x}_t$. But as we explain below, this assumption is incompatible with a non-uniform asynchronous model in the analysis approach used in most of the recent literature.

### 2.2. On the Difficulty of Labeling the Iterates

Formalizing the meaning of $x_t$ and $\hat{x}_t$ highlights a subtle but important difficulty arising when analyzing *randomized* parallel algorithms: what is the meaning of $t$? This is the problem of *labeling* the iterates for the purpose of the analysis, and this labeling can have randomness itself that needs to be taken in consideration when interpreting the meaning of an expression like $\mathbb{E}[x_t]$. In this section, we contrast three different approaches in a unified framework. We notably clarify the dependency issues that the labeling from Mania et al. (2017) resolves and propose a new, simpler labeling which allows for much simpler proof techniques.

7. We note that to be completely formal and define this conditional expectation more precisely, one would need to define another random vector that describes the entire system randomness, including all the reads, writes, delays, etc. Conditioning on $\hat{x}_t$ in (4) is actually a shorthand to indicate that we are conditioning on all the relevant "past" that defines both the value of $\hat{x}_t$ as well as the fact that it was the $t^{\text{th}}$ labeled element. For clarity of exposition, we will not go into this level of technical detail, but one could define the appropriate sigma fields to condition on in order to make this equation fully rigorous.
8. A notable exception is Sag (Le Roux et al., 2012) which has biased updates and thus requires a significantly more complex convergence proof. Making Sag unbiased leads to Saga (Defazio et al., 2014), with a much simpler convergence proof.

We consider algorithms that execute in parallel the following four steps, where $t$ is a global labeling that needs to be defined:[9]

1. Read the information in shared memory ($\hat{x}_t$).
2. Sample $i_t$. (5)
3. Perform some computations using ($\hat{x}_t, i_t$).
4. Write an update to shared memory.

***The "After Write" Approach.*** We call the "after write" approach the standard global labeling scheme used in Niu et al. (2011) and re-used in all the later papers that we mentioned in the related work section, with the notable exceptions of Mania et al. (2017) and Duchi et al. (2015). In this approach, $t$ is a (virtual) global counter recording the number of *successful writes* to the shared memory $x$ (incremented after step 4 in 5); $x_t$ thus represents the (true) content of the shared memory after $t$ updates. The interpretation of the crucial equation (3) then means that $\hat{x}_t$ represents the (delayed) local copy value of the core that made the $(t+1)^{\text{th}}$ successful update; $i_t$ represents the factor sampled by this core for this update. Notice that in this framework, the value of $\hat{x}_t$ and $i_t$ is unknown at "time $t$"; we have to wait to the later time when the next core writes to memory to finally determine that its local variables are the ones labeled by $t$. We thus see that here $\hat{x}_t$ and $i_t$ are not necessarily independent – they share dependence through the assignment of the $t$ label. In particular, if some values of $i_t$ yield faster updates than others, it will influence the label assignment defining $\hat{x}_t$. We provide a concrete example of this possible dependency in Figure 1.

The only way we can think to resolve this issue and ensure unbiasedness is to assume that the computation time for the algorithm running on a core is independent of the sample $i$ chosen. This assumption seems overly strong in the context of potentially heterogeneous factors $f_i$'s, and is thus a fundamental flaw for analyzing non-uniform asynchronous computation that has mostly been ignored in the recent asynchronous optimization literature.[10]

9. Observe that contrary to most asynchronous algorithms, we choose to read the shared parameter vector *before* sampling the next data point. We made this design choice to emphasize that in order for $\hat{x}_t$ and $i_t$ to be independent – which will prove crucial for the analysis – the reading of the shared parameter has to be independent of the sampled data point. Although in practice one would prefer to only read the necessary parameters *after* sampling the relevant data point, for the sake of the analysis we cannot allow this source of dependence. We note that our analysis could also handle reading the parameter first and then sampling as long as independence is ensured, but for clarity of presentation, we decided to make this independence explicit.

   Mania et al. (2017) make the opposite presentation choice. In their main analysis, they explicitly assume that $\hat{x}_t$ and $i_t$ are independent, although they explain that it is not the case in practical implementations. The authors then propose a scheme to handle the dependency directly in their appendix. However, this "fix" can only be applied in a restricted setup: only for the Hogwild algorithm, with the assumption that the norm of the gradient is uniformly bounded. Furthermore, even in this restricted setup, the scheme leads to worsened theoretical results (the bound on $\tau$ is $\kappa^2$ worse). Applying it to a more complex algorithm such as Kromagnon or Asaga would mean overcoming several significant hurdles and is thus still an open problem.

   In the absence of a better option, we choose to enforce the independence of $\hat{x}_t$ and $i_t$ with our modified steps ordering.

10. We note that Bertsekas and Tsitsiklis (1989) briefly discussed this issue (see Section 7.8.3), stressing that their analysis for SGD required that the scheduling of computation was independent from the randomness from SGD, but they did not offer any solution if this assumption was not satisfied. Both the "before read" labeling from Mania et al. (2017) and our proposed "after read" labeling resolve this issue.

| | $f_1$ | $f_2$ |
|---|---|---|
| core 1 | $\times$ | |
| core 2 | $\times$ | |
| $f'_{i_0}(\hat{x}_0)$ | $f'_1(\hat{x}_0)$ | |

| | $f_1$ | $f_2$ |
|---|---|---|
| core 1 | $\times$ | |
| core 2 | | $\times$ |
| | $f'_1(\hat{x}_0)$ | |

| | $f_1$ | $f_2$ |
|---|---|---|
| core 1 | | $\times$ |
| core 2 | $\times$ | |
| | $f'_1(\hat{x}_0)$ | |

| | $f_1$ | $f_2$ |
|---|---|---|
| core 1 | | $\times$ |
| core 2 | | $\times$ |
| | | $f'_2(\hat{x}_0)$ |

Figure 1: Suppose that we have two cores and that $f$ has two factors: $f_1$ which has support on only one variable, and $f_2$ which has support on $10^6$ variables and thus yields a gradient step that is significantly more expensive to compute. $x_0$ is the initial content of the memory, and we do not know yet whether $\hat{x}_0$ is the local copy read by the first core or the second core, but we are sure that $\hat{x}_0 = x_0$ as no update can occur in shared memory without incrementing the counter. There are four possibilities for the next step defining $x_1$ depending on which index $i$ was sampled on each core. If any core samples $i = 1$, we know that $x_1 = x_0 - \gamma f'_1(x_0)$ as it will be the first (much faster update) to complete. This happens in 3 out of 4 possibilities; we thus have that $\mathbb{E}x_1 = x_0 - \gamma(\frac{3}{4} f'_1(x_0) + \frac{1}{4} f'_2(x_0))$. We see that this analysis scheme *does not* satisfy the crucial unbiasedness condition (4). To understand this subtle point better, note that in this very simple example, $i_0$ and $i_1$ are not independent. We can show that $P(i_1 = 2 \mid i_0 = 2) = 1$. They share dependency *through the labeling assignment.*

***The "Before Read" Approach.*** Mania et al. (2017) address this issue by proposing instead to increment the global $t$ counter just *before* a new core starts to *read* the shared memory (before step 1 in 5). In their framework, $\hat{x}_t$ represents the (inconsistent) read that was made by this core in this computational block, and $i_t$ represents the chosen sample. The update rule (3) represents a *definition* of the meaning of $x_t$, which is now a "virtual iterate" as it does not necessarily correspond to the content of the shared memory at any point. The real quantities manipulated by the algorithm in this approach are the $\hat{x}_t$'s, whereas $x_t$ is used only for the analysis – consequently, the critical quantity we want to see vanish is $\mathbb{E}\|\hat{x}_t - x^*\|^2$. The independence of $i_t$ with $\hat{x}_t$ can be simply enforced in this approach by making sure that the way the shared memory $x$ is read does not depend on $i_t$ (e.g. by reading all its coordinates in a fixed order). Note that this implies that we have to read all of $x$'s coordinates, regardless of the size of $f_{i_t}$'s support. This is a much weaker condition than the assumption that all the computation in a block does not depend on $i_t$ as required by the "after write" approach, and is thus more reasonable.

***A New Global Ordering: the "After Read" Approach.*** The "before read" approach gives rise to the following complication in the analysis: $\hat{x}_t$ can depend on $i_r$ for $r > t$. This is because $t$ is a global time ordering only on the assignment of computation to a core, not on when $\hat{x}_t$ was finished being read. This means that we need to consider both the "future" and the "past" when analyzing $x_t$. To simplify the analysis, we thus propose a third way to label the iterates that we call "after read": $\hat{x}_t$ represents the $(t+1)^{\text{th}}$ *fully completed read* ($t$ incremented after step 1 in 5). As in the "before read" approach, we can ensure that $i_t$ is independent of $\hat{x}_t$ by ensuring that how we read does not depend on $i_t$. But unlike in the "before read" approach, $t$ here now does represent a global ordering on the $\hat{x}_t$ iterates – and thus we have that $i_r$ is independent of $\hat{x}_t$ for $r > t$. Again using (3) as the definition of the virtual iterate $x_t$ as in the perturbed iterate framework, we then have a very simple form for the value of $x_t$ and $\hat{x}_t$ (assuming atomic writes, see Property 5 below):

$$\begin{aligned} x_t &= x_0 - \gamma \sum_{u=0}^{t-1} g(\hat{x}_u, \hat{\alpha}^u, i_u) \, ; \\ [\hat{x}_t]_v &= [x_0]_v - \gamma \sum_{\substack{u=0 \\ \text{u s.t. coordinate } v \text{ was written} \\ \text{for } u \text{ before } t}}^{t-1} [g(\hat{x}_u, \hat{\alpha}^u, i_u)]_v \, . \end{aligned} \tag{6}$$

This proved crucial for our Asaga proof and allowed us to obtain better bounds for Hogwild and the Kromagnon algorithm presented in Mania et al. (2017).

The main idea of the perturbed iterate framework is to use this handle on $\hat{x}_t - x_t$ to analyze the convergence for $x_t$. As $x_t$ is a virtual quantity, Mania et al. (2017) supposed that there exists an index $T$ such that $x_T$ lives in shared memory ($T$ is a pre-set final iteration number after which all computation is completed, which means $x_T = \hat{x}_T$) and gave their convergence result for this $x_T$.

In this paper, we instead state the convergence results directly in terms of $\hat{x}_t$, thus avoiding the need for an unwieldy pre-set final iteration counter, and also enabling guarantees during the entire course of the algorithm.

**Remark 1** *As mentioned in footnote 9, Mania et al. (2017) choose to sample a data point first and only then read the shared parameter vector in* (5)*. One advantage of this option is that it allows for reading only the relevant dimensions of the parameter vector, although it means losing the crucial independence property between $\hat{x}_t$ and $i_t$.*

*We can thus consider that their labeling approach is "after sampling" rather than "before read" (both are equivalent given their ordering). If we take this view, then by switching the order of the sampling and the reading steps in their setup, the "after sampling" approach becomes equivalent to our proposed "after read" labeling.*

*However, the framework in which they place their analysis is the "before read" approach as described above, which results in having to take into account troublesome "future" terms in* (6)*. These additional terms make the analysis considerably harder and ultimately lead to worse theoretical results.*

## 3. Asynchronous Parallel Sparse Saga

We start by presenting Sparse Saga, a sparse variant of the Saga algorithm that is more adapted to the asynchronous parallel setting. We then introduce Asaga, the asynchronous parallel version of Sparse Saga. Finally, we state both convergence and speedup results for Asaga and give an outline of their proofs.

### 3.1. Sparse Saga

Borrowing our notation from Hofmann et al. (2015), we first present the original Saga algorithm and then describe our novel sparse variant.

***Original* Saga *Algorithm.*** The standard Saga algorithm (Defazio et al., 2014) maintains two moving quantities to optimize (1): the current iterate $x$ and a table (memory) of historical gradients $(\alpha_i)_{i=1}^n$.[11] At every iteration, the Saga algorithm samples uniformly at random an index $i \in \{1, \ldots, n\}$, and then executes the following update on $x$ and $\alpha$ (for the unconstrained optimization version):

$$x^+ = x - \gamma\big(f_i'(x) - \alpha_i + \bar{\alpha}\big); \qquad \alpha_i^+ = f_i'(x), \tag{7}$$

where $\gamma$ is the step size and $\bar{\alpha} := {}^1\!/\!_n \sum_{i=1}^n \alpha_i$ can be updated efficiently in an online fashion. Crucially, $\mathbf{E}\alpha_i = \bar{\alpha}$ and thus the update direction is unbiased ($\mathbf{E}x^+ = x - \gamma f'(x)$). Furthermore, it can be proven (see Defazio et al., 2014) that under a reasonable condition on $\gamma$, the update has vanishing variance, which enables the algorithm to converge linearly with a constant step size.

***Motivation for a Variant.*** In its current form, every Saga update is dense even if the individual gradients are sparse due to the historical gradient ($\bar{\alpha}$) term. Schmidt et al. (2016) introduced an implementation technique denoted lagged updates in which each iteration has a cost proportional to the size of the support of $f_i'(x)$. However, this technique involves keeping track of past updates and is not easily adaptable to the parallel setting (see Appendix E). We therefore introduce Sparse Saga, a novel variant which explicitly takes sparsity into account and is easily parallelizable.

***Sparse* Saga *Algorithm.*** As in the Sparse Svrg algorithm proposed in Mania et al. (2017), we obtain Sparse Saga by a simple modification of the parameter update rule in (7) where $\bar{\alpha}$ is replaced by a sparse version equivalent in expectation:

$$x^+ = x - \gamma(f_i'(x) - \alpha_i + D_i\bar{\alpha}), \tag{8}$$

where $D_i$ is a diagonal matrix that makes a weighted projection on the support of $f_i'$. More precisely, let $S_i$ be the support of the gradient $f_i'$ function (i.e., the set of coordinates where $f_i'$ can be nonzero). Let $D$ be a $d \times d$ diagonal reweighting matrix, with coefficients ${}^1\!/\!_{p_v}$ on the diagonal, where $p_v$ is the probability that dimension $v$ belongs to $S_i$ when $i$ is sampled uniformly at random in $\{1, ..., n\}$. We then define $D_i := P_{S_i} D$, where $P_{S_i}$ is the projection onto $S_i$. The reweighting by $D$ ensures that $\mathbf{E} D_i \bar{\alpha} = \bar{\alpha}$, and thus that the update is still unbiased despite the sparsifying projection.

***Convergence Result for (Serial) Sparse* Saga*.*** For clarity of exposition, we model our convergence result after the simple form of Hofmann et al. (2015, Corollary 3). Note that the rate we obtain for Sparse Saga is the same as the one obtained in the aforementioned reference for Saga.

**Theorem 2** *Let $\gamma = \frac{a}{5L}$ for any $a \leq 1$. Then* Sparse Saga *converges geometrically in expectation with a rate factor of at least $\rho(a) = \frac{1}{5}\min\left\{\frac{1}{n}, a\frac{1}{\kappa}\right\}$, i.e., for $x_t$ obtained after $t$ updates, we have $\mathbb{E}\|x_t - x^*\|^2 \leq (1-\rho)^t\, C_0$, where $C_0 := \|x_0 - x^*\|^2 + \frac{1}{5L^2}\sum_{i=1}^n \|\alpha_i^0 - f_i'(x^*)\|^2$.*

11. For linear predictor models, the memory $\alpha_i^0$ can be stored as a scalar. Following Hofmann et al. (2015), $\alpha_i^0$ can be initialized to any convenient value (typically 0), unlike the prescribed $f_i'(x_0)$ analyzed in (Defazio et al., 2014).

***Proof outline.*** We reuse the proof technique from Hofmann et al. (2015), in which a combination of classical strong convexity and Lipschitz inequalities is used to derive the inequality (Hofmann et al., 2015, Lemma 1):

$$\mathbf{E}\|x^+ - x^*\|^2 \leq (1-\gamma\mu)\|x - x^*\|^2 + 2\gamma^2 \mathbf{E}\|\alpha_i - f_i'(x^*)\|^2 + (4\gamma^2 L - 2\gamma)\big(f(x) - f(x^*)\big). \quad (9)$$

This gives a contraction term. A Lyapunov function is then defined to control the two other terms. To ensure our variant converges at the same rate as regular Saga, we only need to prove that the above inequality (Hofmann et al., 2015, Lemma 1) is still verified. To prove this, we derive close variants of equations (6) and (9) in their paper. The rest of the proof can be reused without modification. The full details can be found in Appendix A.

***Comparison with Lagged Updates.*** The lagged updates technique in Saga is based on the observation that the updates for component $[x]_v$ need not be applied until this coefficient needs to be accessed, that is, until the next iteration $t$ such that $v \in S_{i_t}$. We refer the reader to Schmidt et al. (2016) for more details.

Interestingly, the expected number of iterations between two steps where a given dimension $v$ is in the support of the partial gradient is $p_v^{-1}$, where $p_v$ is the probability that $v$ is in the support of the partial gradient at a given step. $p_v^{-1}$ is precisely the term which we use to multiply the update to $[x]_v$ in Sparse Saga. Therefore one may see the updates in Sparse Saga as *anticipated* updates, whereas those in the Schmidt et al. (2016) implementation are *lagged*.

The two algorithms appear to be very close, even though Sparse Saga uses an expectation to multiply a given update whereas the lazy implementation uses a random variable (with the same expectation). Sparse Saga therefore uses a slightly more aggressive strategy, which may explain the result of our experiments (see Section 6.3): both Sparse Saga and Saga with lagged updates had similar convergence in terms of number of iterations, with the Sparse Saga scheme being slightly faster in terms of runtime.

Although Sparse Saga requires the computation of the $p_v$ probabilities, this can be done during a first pass throughout the data (during which constant step size Sgd may be used) at a negligible cost.

## 3.2. Asynchronous Parallel Sparse Saga

We describe Asaga, a sparse asynchronous parallel implementation of Sparse Saga, in Algorithm 1 in the theoretical form that we analyze, and in Algorithm 2 as its practical implementation. We state our convergence result and analyze our algorithm using the improved perturbed iterate framework.

In the specific case of (Sparse) Saga, we have to add the additional read memory argument $\hat\alpha^t$ to our perturbed update (3):

$$\begin{aligned} x_{t+1} &:= x_t - \gamma g(\hat x_t, \hat\alpha^t, i_t); \\ g(\hat x_t, \hat\alpha^t, i_t) &:= f'_{i_t}(\hat x_t) - \hat\alpha^t_{i_t} + D_{i_t}\left( {}^1\!/_n \textstyle\sum_{i=1}^n \hat\alpha^t_i \right). \end{aligned} \quad (10)$$

Before stating our convergence result, we highlight some properties of Algorithm 1 and make one central assumption.

**Algorithm 1** Asaga (analyzed algorithm)

```
1:  Initialize shared variables $x$ and $(\alpha_i)_{i=1}^n$
2:  keep doing in parallel
3:    $\hat{x}$ = inconsistent read of $x$
4:    $\forall j$, $\hat{\alpha}_j$ = inconsistent read of $\alpha_j$
5:    Sample $i$ uniformly in $\{1, ..., n\}$
6:    Let $S_i$ be $f_i$'s support
7:    $[\bar{\alpha}]_{S_i} = {}^1\!/_n \sum_{k=1}^n [\hat{\alpha}_k]_{S_i}$
8:
9:    $[\delta x]_{S_i} = -\gamma(f_i'(\hat{x}) - \hat{\alpha}_i + D_i[\bar{\alpha}]_{S_i})$
10:   for $v$ in $S_i$ do
11:     $[x]_v \leftarrow [x]_v + [\delta x]_v$          // atomic
12:     $[\alpha_i]_v \leftarrow [f_i'(\hat{x})]_v$
13:     // '$\leftarrow$' denotes a shared memory update.
14:   end for
15: end parallel loop
```

**Algorithm 2** Asaga (implementation)

```
1:  Initialize shared $x$, $(\alpha_i)_{i=1}^n$ and $\bar{\alpha}$
2:  keep doing in parallel
3:    Sample $i$ uniformly in $\{1, ..., n\}$
4:    Let $S_i$ be $f_i$'s support
5:    $[\hat{x}]_{S_i}$ = inconsistent read of $x$ on $S_i$
6:    $\hat{\alpha}_i$ = inconsistent read of $\alpha_i$
7:    $[\bar{\alpha}]_{S_i}$ = inconsistent read of $\bar{\alpha}$ on $S_i$
8:    $[\delta\alpha]_{S_i} = f_i'([\hat{x}]_{S_i}) - \hat{\alpha}_i$
9:    $[\delta x]_{S_i} = -\gamma([\delta\alpha]_{S_i} + D_i[\bar{\alpha}]_{S_i})$
10:   for $v$ in $S_i$ do
11:     $[x]_v \leftarrow [x]_v + [\delta x]_v$              // atomic
12:     $[\alpha_i]_v \leftarrow [\alpha_i]_v + [\delta\alpha]_v$        // atomic
13:     $[\bar{\alpha}]_v \leftarrow [\bar{\alpha}]_v + {}^1\!/_n[\delta\alpha]_v$      // atomic
14:   end for
15: end parallel loop
```

**Property 3 (independence)** *Given the "after read" global ordering, $i_r$ is independent of $\hat{x}_t$ $\forall r \geq t$.*

The independence property for $r = t$ is assumed in most of the parallel optimization literature, even though it is not verified in case the "after write" labeling is used. We emulate Mania et al. (2017) and *enforce* this independence in Algorithm 1 by having the core read all the shared data parameters and historical gradients before starting their iterations. Although this is too expensive to be practical if the data is sparse, this is required by the theoretical Algorithm 1 that we can analyze. The independence for $r > t$ is a consequence of using the "after read" global ordering instead of the "before read" one.

**Property 4 (unbiased estimator)** *The update, $g_t := g(\hat{x}_t, \hat{\alpha}^t, i_t)$, is an unbiased estimator of the true gradient at $\hat{x}_t$, i.e. (10) yields (4) in conditional expectation.*

This property is crucial for the analysis, as in most related literature. It follows by the independence of $i_t$ with $\hat{x}_t$ and from the computation of $\bar{\alpha}$ on line 7 of Algorithm 1, which ensures that $\mathbb{E}\hat{\alpha}_i = 1/n \sum_{k=1}^n [\hat{\alpha}_k]_{S_i} = [\bar{\alpha}]_{S_i}$, making the update unbiased. In practice, recomputing $\bar{\alpha}$ is not optimal, but storing it instead introduces potential bias issues in the proof (as detailed in Appendix F.3).

**Property 5 (atomicity)** *The shared parameter coordinate update of $[x]_v$ on line 11 is atomic.*

Since our updates are additions, there are no overwrites, even when several cores compete for the same resources. In practice, this is enforced by using *compare-and-swap* semantics, which are heavily optimized at the processor level and have minimal overhead. Our experiments with non-thread safe algorithms (i.e. where this property is not verified, see Figure 7 of Appendix F) show that compare-and-swap is necessary to optimize to high accuracy.

Finally, as is standard in the literature, we make an assumption on the maximum delay that asynchrony can cause – this is the *partially asynchronous* setting as defined in Bertsekas and Tsitsiklis (1989):

**Assumption 6 (bounded overlaps)** *We assume that there exists a uniform bound, called $\tau$, on the maximum number of iterations that can overlap together. We say that iterations $r$ and $t$ overlap if at some point they are processed concurrently. One iteration is being processed from the start of the reading of the shared parameters to the end of the writing of its update. The bound $\tau$ means that iterations $r$ cannot overlap with iteration $t$ for $r \geq t + \tau + 1$, and thus that every coordinate update from iteration $t$ is successfully written to memory before the iteration $t + \tau + 1$ starts.*

Our result will give us conditions on $\tau$ subject to which we have linear speedups. $\tau$ is usually seen as a proxy for $p$, the number of cores (which lowerbounds it). However, though $\tau$ appears to depend linearly on $p$, it actually depends on several other factors (notably the data sparsity distribution) and can be orders of magnitude bigger than $p$ in real-life experiments. We can upper bound $\tau$ by $(p-1)R$, where $R$ is the ratio of the maximum over the minimum iteration time (which encompasses theoretical aspects as well as hardware overhead). More details can be found in Section 6.7.

***Explicit effect of asynchrony.*** By using the overlap Assumption 6 in the expression (6) for the iterates, we obtain the following explicit effect of asynchrony that is crucially used in our proof:

$$\hat{x}_t - x_t = \gamma \sum_{u=(t-\tau)_+}^{t-1} G_u^t g(\hat{x}_u, \hat{\alpha}^u, i_u), \tag{11}$$

where $G_u^t$ are $d \times d$ diagonal matrices with terms in $\{0, +1\}$. From our definition of $t$ and $x_t$, it is clear that every update in $\hat{x}_t$ is already in $x_t$ – this is the 0 case. Conversely, some updates might be late: this is the +1 case. $\hat{x}_t$ may be lacking some updates from the "past" in some sense, whereas given our global ordering definition, it cannot contain updates from the "future".

## 3.3. Convergence and Speedup Results

We now state our main theoretical results. We give a detailed outline of the proof in Section 3.3.2 and its full details in Appendix B.

We first define a notion of problem sparsity, as it will appear in our results.

**Definition 7 (Sparsity)** *As in Niu et al. (2011), we introduce $\Delta_r := \max_{v=1..d} |\{i : v \in S_i\}|$. $\Delta_r$ is the maximum right-degree in the bipartite graph of the factors and the dimensions, i.e., the maximum number of data points with a specific feature. For succinctness, we also define $\Delta := \Delta_r/n$. We have $1 \leq \Delta_r \leq n$, and hence $1/n \leq \Delta \leq 1$.*

### 3.3.1. Convergence and Speedup Statements

**Theorem 8 (Convergence guarantee and rate of Asaga)** *Suppose $\tau < n/10$.*[12] *Let*

$$a^*(\tau) := \frac{1}{32\left(1+\tau\sqrt{\Delta}\right)\xi(\kappa,\Delta,\tau)} \quad \text{where } \xi(\kappa,\Delta,\tau) := \sqrt{1+\frac{1}{8\kappa}\min\{\frac{1}{\sqrt{\Delta}},\tau\}} \quad \text{(note that } \xi(\kappa,\Delta,\tau)\approx 1 \text{ unless } \kappa < 1/\sqrt{\Delta}\ (\leq \sqrt{n})\text{)}. \tag{12}$$

*For any step size $\gamma = \frac{a}{L}$ with $a \leq a^*(\tau)$, the inconsistent read iterates of Algorithm 1 converge in expectation at a geometric rate of at least: $\rho(a) = \frac{1}{5}\min\left\{\frac{1}{n}, a\frac{1}{\kappa}\right\}$, i.e., $\mathbb{E}f(\hat{x}_t) - f(x^*) \leq (1-\rho)^t\,\tilde{C}_0$, where $\tilde{C}_0$ is a constant independent of $t$ ($\approx \frac{n}{\gamma}C_0$ with $C_0$ as defined in Theorem 2).*

This result is very close to Saga's original convergence theorem, but with the maximum step size divided by an extra $1+\tau\sqrt{\Delta}$ factor. Referring to Hofmann et al. (2015) and our own Theorem 2, the rate factor for Saga is $\min\{1/n, a/\kappa\}$ up to a constant factor. Comparing this rate with Theorem 8 and inferring the conditions on the maximum step size $a^*(\tau)$, we get the following conditions on the overlap $\tau$ for Asaga to have the same rate as Saga (comparing upper bounds).

**Corollary 9 (Speedup condition)** *Suppose $\tau \leq \mathcal{O}(n)$ and $\tau \leq \mathcal{O}(\frac{1}{\sqrt{\Delta}}\max\{1, \frac{n}{\kappa}\})$. Then using the step size $\gamma = a^*(\tau)/L$ from (12), Asaga converges geometrically with rate factor $\Omega(\min\{\frac{1}{n},\frac{1}{\kappa}\})$ (similar to Saga), and is thus linearly faster than its sequential counterpart up to a constant factor. Moreover, if $\tau \leq \mathcal{O}(\frac{1}{\sqrt{\Delta}})$, then a universal step size of $\Theta(\frac{1}{L})$ can be used for Asaga to be adaptive to local strong convexity with a similar rate to Saga (i.e., knowledge of $\kappa$ is not required).*

Interestingly, in the well-conditioned regime ($n > \kappa$, where Saga enjoys a range of step sizes which all give the same contraction ratio), Asaga enjoys the same rate as Saga even in the non-sparse regime ($\Delta = 1$) for $\tau < \mathcal{O}(n/\kappa)$. This is in contrast to the previous work on asynchronous incremental gradient methods which required some kind of sparsity to get a theoretical linear speedup over their sequential counterpart (Niu et al., 2011; Mania et al., 2017). In the ill-conditioned regime ($\kappa > n$), sparsity is required for a linear speedup, with a bound on $\tau$ of $\mathcal{O}(\sqrt{n})$ in the best-case (though degenerate) scenario where $\Delta = 1/n$.

The proof for Corollary 9 can be found in Appendix B.9.

***Comparison to related work.***

- We give the first convergence analysis for an asynchronous parallel version of Saga (note that Reddi et al. (2015) only covers an epoch based version of Saga with random stopping times, a fairly different algorithm).
- Theorem 8 can be directly extended to the a parallel extension of the Svrg version from Hofmann et al. (2015) which is adaptive to the local strong convexity with similar rates (see Section 4.2).
- In contrast to the parallel Svrg analysis from Reddi et al. (2015, Thm. 2), our proof technique handles inconsistent reads and a non-uniform processing speed across $f_i$'s.

12. Asaga can actually converge for any $\tau$, but the maximum step size then has a term of $\exp(\tau/n)$ in the denominator with much worse constants. See Appendix B.7.

Our bounds are similar (noting that $\Delta$ is equivalent to theirs), except for the adaptivity to local strong convexity: Asaga does not need to know $\kappa$ for optimal performance, contrary to parallel Svrg (see Section 4 for more details).

- In contrast to the Svrg analysis from Mania et al. (2017, Thm. 14), we obtain a better dependence on the condition number in our rate ($1/\kappa$ vs. $1/\kappa^2$ on their work) and on the sparsity (they obtain $\tau \leq \mathcal{O}(\Delta^{-1/3})$), while we furthermore remove their gradient bound assumption. We also give our convergence guarantee on $\hat{x}_t$ *during* the algorithm, whereas they only bound the error for the "last" iterate $x_T$.

### 3.3.2. Proof Outline of Theorem 8

We give here an extended outline of the proof. We detail key lemmas in Section 3.3.3.

***Initial recursive inequality.*** Let $g_t := g(\hat{x}_t, \hat{\alpha}^t, i_t)$. By expanding the update equation (10) defining the virtual iterate $x_{t+1}$ and introducing $\hat{x}_t$ in the inner product term, we obtain:

$$\begin{aligned}\|x_{t+1} - x^*\|^2 &= \|x_t - \gamma g_t - x^*\|^2 \\ &= \|x_t - x^*\|^2 + \gamma^2\|g_t\|^2 - 2\gamma\langle x_t - x^*, g_t\rangle \\ &= \|x_t - x^*\|^2 + \gamma^2\|g_t\|^2 - 2\gamma\langle \hat{x}_t - x^*, g_t\rangle + 2\gamma\langle \hat{x}_t - x_t, g_t\rangle\,. \end{aligned} \tag{13}$$

Note that we introduce $\hat{x}_t$ in the inner product because $g_t$ is a function of $\hat{x}_t$, not $x_t$.

In the sequential setting, we require $i_t$ to be independent of $x_t$ to obtain unbiasedness. In the perturbed iterate framework, we instead require that $i_t$ is independent of $\hat{x}_t$ (see Property 3). This crucial property enables us to use the unbiasedness condition (4) to write: $\mathbb{E}\langle \hat{x}_t - x^*, g_t\rangle = \mathbb{E}\langle \hat{x}_t - x^*, f'(\hat{x}_t)\rangle$. Taking the expectation of (13) and using this unbiasedness condition we obtain an expression that allows us to use the $\mu$-strong convexity of $f$:[13]

$$\langle \hat{x}_t - x^*, f'(\hat{x}_t)\rangle \geq f(\hat{x}_t) - f(x^*) + \frac{\mu}{2}\|\hat{x}_t - x^*\|^2. \tag{14}$$

With further manipulations on the expectation of (13), including the use of the standard inequality $\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$ (see Appendix B.1), we obtain our basic recursive contraction inequality:

$$a_{t+1} \leq (1 - \frac{\gamma\mu}{2})a_t + \gamma^2\mathbb{E}\|g_t\|^2 \underbrace{+\gamma\mu\mathbb{E}\|\hat{x}_t - x_t\|^2 + 2\gamma\mathbb{E}\langle \hat{x}_t - x_t, g_t\rangle}_{\text{additional asynchrony terms}} -2\gamma e_t\,, \tag{15}$$

where $a_t := \mathbb{E}\|x_t - x^*\|^2$ and $e_t := \mathbb{E}f(\hat{x}_t) - f(x^*)$.

Inequality (15) is a midway point between the one derived in the proof of Lemma 1 in Hofmann et al. (2015) and Equation (2.5) in Mania et al. (2017), because we use the tighter strong convexity bound (14) than in the latter (giving us the important extra term $-2\gamma e_t$).

In the sequential setting, one crucially uses the negative suboptimality term $-2\gamma e_t$ to cancel the variance term $\gamma^2\mathbb{E}\|g_t\|^2$ (thus deriving a condition on $\gamma$). In our setting, we need

13. Note that here is our departure point with Mania et al. (2017) who replaced the $f(\hat{x}_t) - f(x^*)$ term with the lower bound $\frac{\mu}{2}\|\hat{x}_t - x^*\|^2$ in this relationship (see their Equation (2.4)), thus yielding an inequality too loose afterwards to get the fast rates for Svrg.

to bound the additional asynchrony terms using the same negative suboptimality in order to prove convergence and speedup for our parallel algorithm – this will give stronger constraints on the maximum step size.

The rest of the proof then proceeds as follows:

1. By using the expansion (11) for $\hat{x}_t - x_t$, we can bound the additional asynchrony terms in (15) in terms of the past updates $(\mathbb{E}\|g_u\|^2)_{u \leq t}$. This gives Lemma 10 below.

2. We then bound the updates $\mathbb{E}\|g_t\|^2$ in terms of past suboptimalities $(e_v)_{v \leq u}$ by using standard Saga inequalities and carefully analyzing the update rule for $\alpha_i^+$ (7) in expectation. This gives Lemma 13 below.

3. By applying Lemma 13 to the result of Lemma 10, we obtain a master contraction inequality (27) in terms of $a_{t+1}$, $a_t$ and $(e_u)_{u \leq t}$.

4. We define a novel Lyapunov function $\mathcal{L}_t = \sum_{u=0}^{t} (1-\rho)^{t-u} a_u$ and manipulate the master inequality to show that $\mathcal{L}_t$ is bounded by a contraction, subject to a maximum step size condition on $\gamma$ (given in Lemma 14 below).

5. Finally, we unroll the Lyapunov inequality to get the convergence Theorem 8.

### 3.3.3. Details

We list the key lemmas below with their proof sketch, and pointers to the relevant parts of Appendix B for detailed proofs.

**Lemma 10 (Inequality in terms of $g_t := g(\hat{x}_t, \hat{\alpha}^t, i_t)$)** *For all $t \geq 0$:*

$$a_{t+1} \leq (1 - \frac{\gamma\mu}{2}) a_t + \gamma^2 C_1 \mathbb{E}\|g_t\|^2 + \gamma^2 C_2 \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\|g_u\|^2 - 2\gamma e_t \,, \tag{16}$$

$$\textit{where} \quad C_1 := 1 + \sqrt{\Delta}\tau \quad \textit{and} \quad C_2 := \sqrt{\Delta} + \gamma\mu C_1 \,. \quad ^{14} \tag{17}$$

To prove this lemma we need to bound both $\mathbb{E}\|\hat{x}_t - x^*\|^2$ and $\mathbb{E}\langle \hat{x}_t - x_t, g_t \rangle$ with respect to $(\mathbb{E}\|g_u\|^2)_{u \leq t}$. We achieve this by crucially using Equation (11), together with the following proposition, which we derive by a combination of Cauchy-Schwartz and our sparsity definition (see Section B.2).

**Proposition 11** *For any $u \neq t$,*

$$\mathbb{E}|\langle g_u, g_t \rangle| \leq \frac{\sqrt{\Delta}}{2} (\mathbb{E}\|g_u\|^2 + \mathbb{E}\|g_t\|^2) \,. \tag{18}$$

To derive this essential inequality for both the right-hand-side terms of Eq. (18), we start by proving a relevant property of $\Delta$. We reuse the sparsity constant introduced in Reddi et al. (2015) and relate it to the one we have defined earlier, $\Delta_r$:

14. Note that $C_2$ depends on $\gamma$. In the rest of the paper, we write $C_2(\gamma)$ instead of $C_2$ when we want to draw attention to that dependency.

**Remark 12** *Let $D$ be the smallest constant such that:*

$$\mathbf{E}\|x\|_i^2 = \frac{1}{n}\sum_{i=1}^n \|x\|_i^2 \leq D\|x\|^2 \quad \forall x \in \mathbb{R}^d, \tag{19}$$

*where $\|.\|_i$ is defined to be the $\ell_2$-norm restricted to the support $S_i$ of $f_i$. We have:*

$$D = \frac{\Delta_r}{n} = \Delta\,. \tag{20}$$

**Proof** We have:

$$\mathbf{E}\|x\|_i^2 = \frac{1}{n}\sum_{i=1}^n \|x\|_i^2 = \frac{1}{n}\sum_{i=1}^n \sum_{v\in S_i} [x]_v^2 = \frac{1}{n}\sum_{v=1}^d \sum_{i|v\in S_i} [x]_v^2 = \frac{1}{n}\sum_{v=1}^d \delta_v [x]_v^2\,, \tag{21}$$

where $\delta_v := |(i \mid v \in S_i)|$. This implies:

$$D \geq \frac{1}{n}\sum_{v=1}^d \delta_v \frac{[x]_v^2}{\|x\|^2}\,. \tag{22}$$

Since $D$ is the minimum constant satisfying this inequality, we have:

$$D = \max_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{v=1}^d \delta_v \frac{[x]_v^2}{\|x\|^2}\,. \tag{23}$$

We need to find $x$ such that it maximizes the right-hand side term. Note that the vector $([x]_v^2/\|x\|^2)_{v=1..d}$ is in the unit probability simplex, which means that an equivalent problem is the maximization over all convex combinations of $(\delta_v)_{v=1..d}$. This maximum is found by putting all the weight on the maximum $\delta_v$, which is $\Delta_r$ by definition.

This implies that $\Delta = \Delta_r/n$ is indeed the smallest constant satisfying (19). ■

***Proof of Proposition 11*** Let $u \neq t$. Without loss of generality, $u < t$.[15] Then:

$$\begin{aligned}
\mathbb{E}|\langle g_u, g_t\rangle| &\leq \mathbb{E}\|g_u\|_{i_t}\|g_t\| && \text{(Sparse inner product; support of } g_t \text{ is } S_{i_t}\text{)}\\
&\leq \sqrt{\mathbb{E}\|g_u\|_{i_t}^2}\sqrt{\mathbb{E}\|g_t\|^2} && \text{(Cauchy-Schwarz for expectations)}\\
&\leq \sqrt{\Delta\mathbb{E}\|g_u\|^2}\sqrt{\mathbb{E}\|g_t\|^2} && \text{(Remark 12 and } i_t \perp\!\!\!\perp g_u, \forall u < t\text{)}\\
&\leq \frac{\sqrt{\Delta}}{2}(\mathbb{E}\|g_u\|^2 + \mathbb{E}\|g_t\|^2)\,. && \text{(AM-GM inequality)}
\end{aligned}$$

All told, we have:

$$\mathbb{E}|\langle g_u, g_t\rangle| \leq \frac{\sqrt{\Delta}}{2}(\mathbb{E}\|g_u\|^2 + \mathbb{E}\|g_t\|^2)\,. \tag{24}$$

15. One only has to switch $u$ and $t$ if $u > t$.

**Lemma 13 (Suboptimality bound on $\mathbb{E}\|g_t\|^2$)** *For all $t \geq 0$,*

$$\mathbb{E}\|g_t\|^2 \leq 4Le_t + \frac{4L}{n}\sum_{u=1}^{t-1}(1-\frac{1}{n})^{(t-2\tau-u-1)_+}e_u + 4L(1-\frac{1}{n})^{(t-\tau)_+}\tilde{e}_0\,, \tag{25}$$

*where* $\tilde{e}_0 := \frac{1}{2L}\mathbb{E}\|\alpha_i^0 - f_i'(x^*)\|^2$.[16]

From our proof of convergence for Sparse Saga we know that (see Appendix A):

$$\mathbb{E}\|g_t\|^2 \leq 2\mathbb{E}\|f'_{i_t}(\hat{x}_t) - f'_{i_t}(x^*)\|^2 + 2\mathbb{E}\|\hat{\alpha}^t_{i_t} - f'_{i_t}(x^*)\|^2. \tag{26}$$

We can handle the first term by taking the expectation over a Lipschitz inequality (Hofmann et al. (2015, Equations 7 and 8). All that remains to prove the lemma is to express the $\mathbb{E}\|\hat{\alpha}^t_{i_t} - f'_{i_t}(x^*)\|^2$ term in terms of past suboptimalities. We note that it can be seen as an expectation of past first terms with an adequate probability distribution which we derive and bound.

From our algorithm, we know that each dimension of the memory vector $[\hat{\alpha}_i]_v$ contains a partial gradient computed at some point in the past $[f_i'(\hat{x}_{u^t_{i,v}})]_v$[17] (unless $u = 0$, in which case we replace the partial gradient with $\alpha_i^0$). We then derive bounds on $P(u^t_{i,v} = u)$ and sum on all possible $u$. Together with clever conditioning, we obtain Lemma 13 (see Section B.3).

***Master inequality.*** Let $H_t$ be defined as $H_t := \sum_{u=1}^{t-1}(1-\frac{1}{n})^{(t-2\tau-u-1)_+}e_u$. Then, by setting (25) into Lemma 10, we get (see Section B.5):

$$\begin{aligned} a_{t+1} \leq& (1-\frac{\gamma\mu}{2})a_t - 2\gamma e_t + 4L\gamma^2 C_1\big(e_t + (1-\frac{1}{n})^{(t-\tau)_+}\tilde{e}_0\big) + \frac{4L\gamma^2 C_1}{n}H_t \\ &+ 4L\gamma^2 C_2\sum_{u=(t-\tau)_+}^{t-1}\big(e_u + (1-\frac{1}{n})^{(u-\tau)_+}\tilde{e}_0\big) + \frac{4L\gamma^2 C_2}{n}\sum_{u=(t-\tau)_+}^{t-1}H_u\,. \end{aligned} \tag{27}$$

***Lyapunov function and associated recursive inequality.*** We now have the beginning of a contraction with additional positive terms which all converge to 0 as we near the optimum, as well as our classical negative suboptimality term. This is not unusual in the variance reduction literature. One successful approach in the sequential case is then to define a Lyapunov function which encompasses all terms and is a true contraction (see Defazio et al., 2014; Hofmann et al., 2015). We emulate this solution here. However, while all terms in the sequential case only depend on the current iterate, $t$, in the parallel case we have terms "from the past" in our inequality. To resolve this issue, we define a more involved Lyapunov function which also encompasses past iterates:

$$\mathcal{L}_t = \sum_{u=0}^{t}(1-\rho)^{t-u}a_u, \quad 0 < \rho < 1, \tag{28}$$

where $\rho$ is a target contraction rate that we define later.

16. We introduce this quantity instead of $e_0$ so as to be able to handle the arbitrary initialization of the $\alpha_i^0$.
17. More precisely: $\forall t, i, v \;\; \exists u^t_{i,v} < t$ s.t. $[\hat{\alpha}^t_i]_v = [f_i'(\hat{x}_{u^t_{i,v}})]_v$.

Using the master inequality (27), we get (see Appendix B.6):

$$\begin{aligned}\mathcal{L}_{t+1} &= (1-\rho)^{t+1}a_0 + \sum_{u=0}^{t}(1-\rho)^{t-u}a_{u+1} \\ &\leq (1-\rho)^{t+1}a_0 + (1-\frac{\gamma\mu}{2})\mathcal{L}_t + \sum_{u=1}^{t} r_u^t e_u + r_0^t \tilde{e}_0\,. \end{aligned} \tag{29}$$

The aim is to prove that $\mathcal{L}_t$ is bounded by a contraction. We have two promising terms at the beginning of the inequality, and then we need to handle the last term. Basically, we can rearrange the sums in (27) to expose a simple sum of $e_u$ multiplied by factors $r_u^t$.

Under specific conditions on $\rho$ and $\gamma$, we can prove that $r_u^t$ is negative for all $u \geq 1$, which coupled with the fact that each $e_u$ is positive means that we can safely drop the sum term from the inequality. The $r_0^t$ term is a bit trickier and is handled separately.

In order to obtain a bound on $e_t$ directly rather than on $\mathbb{E}\|\hat{x}_t - x^*\|^2$, we then introduce an additional $\gamma e_t$ term on both sides of (29). The bound on $\gamma$ under which the modified $r_t^t + \gamma$ is negative is then twice as small (we could have used any multiplier between 0 and $2\gamma$, but chose $\gamma$ for simplicity's sake). This condition is given in the following Lemma.

**Lemma 14 (Sufficient condition for convergence)** *Suppose $\tau < n/10$ and $\rho \leq 1/4n$. If*

$$\gamma \leq \gamma^* = \frac{1}{32L(1+\sqrt{\Delta}\tau)\sqrt{1+\frac{1}{8\kappa}\min(\tau, \frac{1}{\sqrt{\Delta}})}} \tag{30}$$

*then for all $u \geq 1$, the coefficients $r_u^t$ from* (29) *are negative. Furthermore, we have $r_t^t + \gamma \leq 0$ and thus:*

$$\gamma e_t + \mathcal{L}_{t+1} \leq (1-\rho)^{t+1}a_0 + (1-\frac{\gamma\mu}{2})\mathcal{L}_t + r_0^t \tilde{e}_0\,. \tag{31}$$

We obtain this result after carefully deriving the $r_u^t$ terms. We find a second-order polynomial inequality in $\gamma$, which we simplify down to (30) (see Appendix B.7).

We can then finish the argument to bound the suboptimality error $e_t$. We have:

$$\mathcal{L}_{t+1} \leq \gamma e_t + \mathcal{L}_{t+1} \leq (1-\frac{\gamma\mu}{2})\mathcal{L}_t + (1-\rho)^{t+1}(a_0 + A\tilde{e}_0)\,. \tag{32}$$

We have two linearly contracting terms. The sum contracts linearly with the worst rate between the two (the smallest geometric rate factor). If we define $\rho^* := \nu \min(\rho, \gamma\mu/2)$, with $0 < \nu < 1$,[18] then we get:

$$\gamma e_t + \mathcal{L}_{t+1} \leq (1-\frac{\gamma\mu}{2})^{t+1}\mathcal{L}_0 + (1-\rho^*)^{t+1}\frac{a_0 + A\tilde{e}_0}{1-\eta} \tag{33}$$

$$\gamma e_t \leq (1-\rho^*)^{t+1}\big(\mathcal{L}_0 + \frac{1}{1-\eta}(a_0 + A\tilde{e}_0)\big)\,, \tag{34}$$

where $\eta := \frac{1-M}{1-\rho^*}$ with $M := \max(\rho, \gamma\mu/2)$. Our geometric rate factor is thus $\rho^*$ (see Appendix B.8).

18. $\nu$ is introduced to circumvent the problematic case where $\rho$ and $\gamma\mu/2$ are too close together.

## 4. Asynchronous Parallel Svrg with the "After Read" Labeling

**Asaga *vs. asynchronous* Svrg.** There are several scenarios in which Asaga can be practically advantageous over its closely related cousin, asynchronous Svrg (note though that "asynchronous" Svrg still requires one synchronization step per epoch to compute a full gradient).

First, while Saga trades memory for less computation, in the case of generalized linear models the memory cost can be reduced to $\mathcal{O}(n)$, compared to $\mathcal{O}(d)$ for Svrg (Johnson and Zhang, 2013). This is of course also true for their asynchronous counterparts.

Second, as Asaga does not require any synchronization steps, it is better suited to heterogeneous computing environments (where cores have different clock speeds or are shared with other applications).

Finally, Asaga does not require knowing the condition number $\kappa$ for optimal convergence in the sparse regime. It is thus adaptive to local strong convexity, whereas Svrg is not. Indeed, Svrg and its asynchronous variant require setting an additional hyper-parameter – the epoch size $m$ – which needs to be at least $\Omega(\kappa)$ for convergence but yields a slower effective convergence rate than Asaga if it is set much bigger than $\kappa$. Svrg thus requires tuning this additional hyper-parameter or running the risk of either slower convergence (if the epoch size chosen is much bigger than the condition number) or even not converging at all (if $m$ is chosen to be much smaller than $\kappa$).[19]

***Motivation for analyzing asynchronous* Svrg.** Despite the advantages that we have just listed, in the case of complex models, the storage cost of Saga may become too expensive for practical use. Svrg (Johnson and Zhang, 2013) trades off more computation for less storage and does not suffer from this drawback. It can thus be applied to cases where Saga cannot (e.g. deep learning models, see Reddi et al., 2016).

Another advantage of Kromagnon is that the historical gradient term $f'(\tilde{x})$ is fixed during an epoch, while its Asaga equivalent, $\bar{\alpha}$, has to be updated at each iteration, either by recomputing if from the $\hat{\alpha}$ – which is costly – or by updating a maintained quantity – which is cheaper but may ultimately result in introducing some bias in the update (see Appendix F.3 for more details on this subtle issue).

It is thus worthwhile to carry out the analysis of Kromagnon (Mania et al., 2017)[20], the asynchronous parallel version of Svrg, although it has to be noted that since Svrg requires regularly computing batch gradients, Kromagnon will present regular synchronization steps as well as coordinated computation – making it less attractive for the asynchronous parallel setting.

We first extend our Asaga analysis to analyze the convergence of a variant of Svrg presented in Hofmann et al. (2015), obtaining exactly the same bounds. This variant improves upon the initial algorithm because it does not require tuning the epoch size hyperparameter and is thus adaptive to local strong convexity (see Section 4.1). Furthermore, it allows for a

19. Note that as Saga (and contrary to the original Svrg) the Svrg variant from Hofmann et al. (2015) does not require knowledge of $\kappa$ and is thus adaptive to local strong convexity, which carries over to its asynchronous adaptation that we analyze in Section 4.2.
20. The speedup analysis presented in Mania et al. (2017) is not fully satisfactory as it does not achieve state-of-the-art convergence results for either Svrg or Kromagnon. Furthermore, we are able to remove their uniform gradient bound assumption, which is inconsistent with strong convexity.

cleaner analysis where – contrary to Svrg – we do not have to replace the final parameters of an epoch by one of its random iterates.

Then, using our "after read" labeling, we are also able to derive a convergence and speedup proof for Kromagnon, with comparable results to our Asaga analysis. In particular, we prove that as for Asaga in the "well-conditioned" regime Kromagnon can achieve a linear speedup even without sparsity assumptions.

## 4.1. Svrg Algorithms

We start by describing the original Svrg algorithm, the variant given in Hofmann et al. (2015) and the sparse asynchronous parallel adaptation, Kromagnon.

***Original* Svrg *algorithm.*** The standard Svrg algorithm (Johnson and Zhang, 2013) is very similar to Saga. The main difference is that instead of maintaining a table of historical gradients, Svrg uses a "reference" batch gradient $f'(\tilde{x})$, updated at regular intervals (typically every $m$ iterations, where $m$ is a hyper-parameter). Svrg is thus an epoch-based algorithm, where at the beginning of every epoch a reference iterate $\tilde{x}$ is chosen and its gradient is computed. Then, at every iteration in the epoch, the algorithm samples uniformly at random an index $i \in \{1, \dots, n\}$, and then executes the following update on $x$:

$$x^+ = x - \gamma\big(f_i'(x) - f_i'(\tilde{x}) + f'(\tilde{x})\big)\,. \tag{35}$$

As for Saga the update direction is unbiased ($\mathbf{E}x^+ = x - \gamma f'(x)$) and it can be proven (see Johnson and Zhang, 2013) that under a reasonable condition on $\gamma$ and $m$ (the epoch size), the update has vanishing variance, which enables the algorithm to converge linearly with a constant step size.

***Hofmann's* Svrg *variant.*** Hofmann et al. (2015) introduce a variant where the size of the epoch is a random variable. At each iteration $t$, a first Bernoulli random variable $B_t$ with $p = 1/n$ is sampled. If $B_t = 1$, then the algorithm updates the reference iterate, $\tilde{x} = x_t$ and computes its full gradient as its new "reference gradient". If $B_t = 0$, the algorithm executes the normal Svrg inner update. Note that this variant is adaptive to local strong convexity, as it does not require the inner loop epoch size $m = \Omega(\kappa)$ as a hyperparameter. In that respect it is closer to Saga than the original Svrg algorithm which is not adaptive.

**Kromagnon.** Kromagnon, introduced in Mania et al. (2017) is obtained by using the same sparse update technique as Sparse Saga, and then running the resulting algorithm in parallel (see Algorithm 3).

## 4.2. Extension to the Svrg Variant from Hofmann et al. (2015)

We introduce Ahsvrg – a sparse asynchronous parallel version for the Svrg variant from Hofmann et al. (2015) – in Algorithm 4. Every core runs stochastic updates independently as long as they are all sampling inner updates, and coordinate whenever one of them decides to do a batch gradient computation. The one difficulty of this approach is that each core needs to be able to communicate to every other core that they should stop doing inner updates and start computing a synchronized batch gradient instead.

To this end, we introduce a new shared variable, $s$, which represents the "state" of the computation. This variable is checked by each core $c$ before each update. If $s = 1$, then

**Algorithm 3** Kromagnon (Mania et al., 2017)

```
1: Initialize shared $x$ and $x_0$
2: while True do
3:   Compute in parallel $g = f'(x_0)$ (synchronously)
4:   for $i = 1..m$ do in parallel (asynchronously)
5:     Sample $i$ uniformly in $\{1, ..., n\}$
6:     Let $S_i$ be $f_i$'s support
7:     $[\hat{x}]_{S_i}$ = inconsistent read of $x$ on $S_i$
8:     $[\delta x]_{S_i} = -\gamma([f_i'(\hat{x}_t) - f_i'(x_0)]_{S_i} + D_i[g]_{S_i})$
9:     for $v$ in $S_i$ do
10:      $[x]_v = [x]_v + [\delta x]_v$                // atomic
11:    end for
12:  end parallel loop
13:  $x_0 = x$
14: end while
```

**Algorithm 4** Ahsvrg

```
1: Initialize shared $x$, $s$ and $x_0$
2: while True do
3:   Compute in parallel $g = f'(x_0)$ (synchronously)
4:   $s = 0$
5:   while $s = 0$ do in parallel (asynchronously)
6:     Sample $B$ with $p = 1/n$
7:     if $B = 1$ then
8:       $s = 1$
9:     else
10:      Sample $i$ uniformly in $\{1, ..., n\}$
11:      Let $S_i$ be $f_i$'s support
12:      $[\hat{x}]_{S_i}$ = inconsistent read of $x$ on $S_i$
13:      $[\delta x]_{S_i} = -\gamma([f_i'(\hat{x}_t) - f_i'(x_0)]_{S_i} + D_i[g]_{S_i})$
14:      for $v$ in $S_i$ do
15:        $[x]_v = [x]_v + [\delta x]_v$              // atomic
16:      end for
17:    end if
18:  end parallel loop
19:  $x_0 = x$
20: end while
```

another core has called for a batch gradient computation and core $c$ starts computing its allocated part of this computation. If $s = 0$, core $c$ proceeds to sample a first random variable. Then it either samples and performs an inner update and keeps going, or it samples a full gradient computation, in which case it updates $s$ to 1 and starts computing its allocated part of the computation. Once a full gradient is computed, $s$ is set to 0 once again and every core resume their loop.

Our Asaga convergence and speedup proofs can easily be adapted to accommodate Ahsvrg since it is closer to Saga than the initial Svrg algorithm. To prove convergence, all one has to do is to modify Lemma 13 very slightly (the only difference is that the $(t-2\tau-u-1)_+$ exponent is replaced by $(t-\tau-u-1)_+$ and the rest of the proof can be used as is). The justification for this small tweak is that the batch steps in Svrg are fully synchronized. More details can be found in Appendix B.4.

## 4.3. Fast Convergence and Speedup Rates for Kromagnon

We now state our main theoretical results. We give a detailed outline of the proof in Section 4.4 and its full details in Appendix C.

**Theorem 15 (Convergence guarantee and rate of Kromagnon)** *Suppose the step size $\gamma$ and epoch size $m$ are chosen such that the following condition holds:*

$$0 < \theta := \frac{\frac{1}{\mu\gamma m} + 2L(1+2\sqrt{\Delta}\tau)(\gamma + \tau\mu\gamma^2)}{1 - 2L(1+2\sqrt{\Delta}\tau)(\gamma+\tau\mu\gamma^2)} < 1\,. \tag{36}$$

*Then the inconsistent read iterates of* Kromagnon *converge in expectation at a geometric rate, i.e.*

$$\mathbb{E}f(\tilde{x}_k) - f(x^*) \leq \theta^t(f(x_0) - f(x^*))\,, \tag{37}$$

*where $\tilde{x}_k$ is the initial iterate for epoch $k$, which is obtained by choosing uniformly at random among the inconsistent read iterates from the previous epoch.*

This result is similar to the theorem given in the original Svrg paper (Johnson and Zhang, 2013). Indeed, if we remove the asynchronous part (i.e. if we set $\tau = 0$), we get exactly the same rate and condition. It also has the same form as the one given in Reddi et al. (2015), which was derived for dense asynchronous Svrg in the easier setting of consistent read and writes (and in the flawed "after write" framework), and gives essentially the same conditions on $\gamma$ and $m$.

In the canonical example presented in most Svrg papers, with $\kappa = n$, $m = \mathcal{O}(n)$ and $\gamma = 1/10L$, Svrg obtains a convergence rate of 0.5. Reddi et al. (2015) get the same rate by setting $\gamma = 1/20\max(1,\sqrt{\Delta}\tau)L$ and $m = \mathcal{O}\big(n(1+\sqrt{\Delta}\tau)\big)$. Following the same line of reasoning (setting $\gamma = 1/20\max(1,\sqrt{\Delta}\tau)L$, $\tau = \mathcal{O}(n)$ and $\theta = 0.5$ and computing the resulting condition on $m$), these values for $\gamma$ and $m$ also give us a convergence rate of 0.5. Therefore, as in Reddi et al. (2015), when $\kappa = n$ we get a linear speedup for $\tau < 1/\sqrt{\Delta}$ (which can be as big as $\sqrt{n}$ in the degenerate case where no data points share any feature with each other). Note that this is the same speedup condition as Asaga in this regime.

Svrg theorems are usually similar to Theorem 15, which does not give an optimal step size or epoch size. This makes the analysis of a parallel speedup difficult, prompting authors to compare rates in specific cases with most parameters fixed, as we have just done. In order to investigate the speedup and step size conditions more precisely and thus derive a more general theorem, we now give Svrg and Kromagnon results modeled on Theorem 8.

**Corollary 16 (Convergence guarantee and rate for serial Svrg)** *Let $\gamma = \frac{a}{4L}$ for any $a \leq \frac{1}{4}$ and $m = \frac{32\kappa}{a}$. Then* Svrg *converges geometrically in expectation with a rate factor per gradient computation of at least $\rho(a) = \frac{1}{4}\min\left\{\frac{1}{n}, \frac{a}{64\kappa}\right\}$, i.e.*

$$\mathbb{E}f(\tilde{x}_k) - f(x^*) \leq (1-\rho)^{k(2m+n)}\left(f(x_0) - f(x^*)\right) \qquad \forall k \geq 0\,. \tag{38}$$

Due to Svrg's special structure, we cannot write $\mathbb{E}f(x_t) - f(x^*) \leq (1-\rho)^t\left(f(x_0) - f(x^*)\right)$ for all $t \geq 0$. However, expressing the convergence properties of this algorithm in terms of a rate factor per gradient computation (of which there are $2m + n$ per epoch) makes it easier to compare convergence rates, either to similar algorithms such as Saga or to its parallel variant Kromagnon – and thus to study the speedup obtained by parallelizing Svrg.

Compared to Saga, this result is very close. The main difference is that the additional hyper-parameter $m$ has to be set and requires knowledge of $\mu$. This illustrates the fact that Svrg is not adaptive to local strong convexity, whereas both Saga and Hofmann's Svrg are.

**Corollary 17 (Simplified convergence guarantee and rate for Kromagnon)** *Let*

$$a^*(\tau) = \frac{1}{4(1 + 2\sqrt{\Delta}\tau)(1 + \frac{\tau}{16\kappa})}\,. \tag{39}$$

*For any step size $\gamma = \frac{a}{4L}$ with $a \leq a^*(\tau)$ and $m = \frac{32\kappa}{a}$,* Kromagnon *converges geometrically in expectation with a rate factor per gradient computation of at least $\rho(a) = \frac{1}{4}\min\left\{\frac{1}{n}, \frac{a}{64\kappa}\right\}$, i.e.*

$$\mathbb{E}f(\tilde{x}_k) - f(x^*) \leq (1-\rho)^{k(2m+n)}\left(f(x_0) - f(x^*)\right) \qquad \forall k \geq 0\,. \tag{40}$$

This result is again quite close to Corollary 16 derived in the serial case. We see that the maximum step size is divided by an additional $(1 + 2\tau\sqrt{\Delta})$ term, while the convergence rate is the same. Comparing the rates and the maximum allowable step sizes in both settings give us the sufficient condition on $\tau$ to get a linear speedup.

**Corollary 18 (Speedup condition)** *Suppose $\tau \leq \mathcal{O}(n)$ and $\tau \leq \mathcal{O}(\frac{1}{\sqrt{\Delta}}\max\{1, \frac{n}{\kappa}\})$. If $n \geq \kappa$, also suppose $\tau \leq \sqrt{n\Delta^{-1/2}}$. Then using the step size $\gamma = a^*(\tau)/L$ from* (39), Kromagnon *converges geometrically with rate factor $\Omega(\min\{\frac{1}{n}, \frac{1}{\kappa}\})$ (similar to* Svrg*), and is thus linearly faster than its sequential counterpart up to a constant factor.*

This result is almost the same as Asaga, with the additional condition that $\tau \leq \mathcal{O}(\sqrt{n})$ in the well-conditioned regime. We see that in this regime Kromagnon can also get the same rate as Svrg even without sparsity, which had not been observed in previous work.

Furthermore, one has to note that $\tau$ is generally smaller for Kromagnon than for Asaga since it is reset to 0 at the beginning of each new epoch (where all cores are synchronized once more).

***Comparison to related work.***

- Corollary 16 provides a rate of convergence *per gradient computation* for Svrg, contrary to most of the literature on this algorithm (including the seminal paper Johnson and Zhang, 2013). This result allows for easy comparison with Saga and other algorithms (in contrast, Konečný and Richtárik 2013 is more involved).
- In contrast to the Svrg analysis from Reddi et al. (2015, Thm. 2), our proof technique handles inconsistent reads and a non-uniform processing speed across $f_i$'s. While Theorem 15 is similar to theirs, Corollary 16 and 17 are more precise results. They enable a finer analysis of the speedup conditions (Corollary 18) – including the possible speedup without sparsity regime.
- In contrast to the Kromagnon analysis from Mania et al. (2017, Thm. 14), Theorem 15 gives a better dependence on the condition number in the rate ($1/\kappa$ vs. $1/\kappa^2$ for them) and on the sparsity (they get $\tau \leq \mathcal{O}(\Delta^{-1/3})$), while we remove their gradient bound assumption. Our results are state-of-the-art for Svrg (contrary to theirs) and so our speedup comparison is more meaningful. Finally, Theorem 15 gives convergence guarantees on $\hat{x}_t$ *during* the algorithm, whereas they only bound the error for the "last" iterate $x_T$.

### 4.4. Proof of Theorem 15

We now give a detailed outline of the proof. Its full details can be found in Appendix C.

Our proof technique begins as our Asaga analysis. In particular, Properties 3, 4, 5 are also verified for Kromagnon[21], and as in our Asaga analysis, we make Assumption 6 (bounded overlaps). Consequently, the basic recursive contraction inequality (15) and Lemma 10 also hold. However, when we derive the equivalent of Lemma 13, we get a slightly different form, which prompts a difference in the rest of the proof technique.

**Lemma 19 (Suboptimality bound on $\mathbb{E}\|g_t\|^2$)**

$$\mathbb{E}\|g_t\|^2 \leq 4Le_t + 4L\tilde{e}_k \qquad \forall k \geq 0, km \leq t \leq (k+1)m\,, \tag{41}$$

*where $\tilde{e}_k := \mathbb{E}f(\tilde{x}_k) - f(x^*)$ and $\tilde{x}_k$ is the initial iterate for epoch $k$.*

We give the proof in Appendix C.1. To derive both terms, we use the same technique as for the first term of Lemma 13. Although this is a much simpler result than Lemma 13 in the case of Asaga, two key differences prevent us from reusing the same Lyapunov function proof technique. First, the $e_0$ term in Lemma 13 is replaced by $\tilde{e}_k$ which depends on the epoch number. Second, this term is not multiplied by a geometrically decreasing quantity, which means the $-2\gamma e_0$ term is not sufficient to cancel out all of the $e_0$ terms coming from subsequent inequalities. To solve this issue, we go to more traditional Svrg techniques.

The rest of the proof is as follows:

1. By substituting Lemma 19 into Lemma 10, we get a master contraction inequality (42) in terms of $a_{t+1}$, $a_t$ and $e_u, u \leq t$.

21. Note that similarly to Asaga, the Kromagnon algorithm which we analyze reads the parameters first and then samples. This is necessary in order for Property 3 to be verified at $r = t$, although not practical when it comes to actual implementation.

2. As in Johnson and Zhang (2013), we sum the master contraction inequality over a whole epoch, and then use the same randomization trick (44) to relate $(e_t)_{km \leq t \leq (k+1)m-1}$ to $\tilde{e}_k$.

3. We thus obtain a contraction inequality between $\tilde{e}_k$ and $\tilde{e}_{k-1}$, which finishes the proof for Theorem 15.

4. We then only have to derive the conditions on $\gamma, \tau$ and $m$ under which we contractions and compare convergence rates to finish the proofs for Corollary 16, Corollary 17 and Corollary 18.

We list the key points below with their proof sketch, and give the detailed proof in Appendix C.

***Master inequality.*** As in our Asaga analysis, we apply (41) to the result of Lemma 10, which gives us that for all $k \geq 0, km \leq t \leq (k+1)m-1$ (see Appendix C.2):

$$a_{t+1} \leq (1-\frac{\gamma\mu}{2})a_t + (4L\gamma^2 C_1 - 2\gamma)e_t + 4L\gamma^2 C_2 \sum_{u=\max(km,t-\tau)}^{t-1} e_u + (4L\gamma^2 C_1 + 4L\gamma^2 \tau C_2)\tilde{e}_k\,. \tag{42}$$

***Contraction inequality.*** As we previously mentioned, the term in $\tilde{e}_k$ is not multiplied by a geometrically decreasing factor, so using the same Lyapunov function as for Asaga cannot work. Instead, we apply the same method as in the original Svrg paper (Johnson and Zhang, 2013): we sum the master contraction inequality over a whole epoch. This gives us (see Appendix C.2):

$$a_{(k+1)m} \leq a_{km} + (4L\gamma^2 C_1 + 4L\gamma^2\tau C_2 - 2\gamma) \sum_{t=km}^{(k+1)m-1} e_t + m(4L\gamma^2 C_1 + 4L\gamma^2 \tau C_2)\tilde{e}_k\,. \tag{43}$$

To cancel out the $\tilde{e}_k$ term, we only have one negative term on the right-hand side of (43): $-2\gamma \sum_{t=km}^{(k+1)m-1} e_t$. This means we need to relate $\sum_{t=km}^{(k+1)m-1} e_t$ to $\tilde{e}_k$. We can do it using the same randomization trick as in Johnson and Zhang (2013): instead of choosing the last iterate of the $k^{\text{th}}$ epoch as $\tilde{x}_k$, we pick one of the iterates of the epoch uniformly at random. This means we get:

$$\tilde{e}_k = \mathbb{E}f(\tilde{x}_k) - f(x^*) = \frac{1}{m}\sum_{t=(k-1)m}^{km-1} e_t \tag{44}$$

We now have: $\sum_{t=km}^{(k+1)m-1} e_t = m\tilde{e}_{k+1}$. Combined with the fact that $a_{km} \leq \frac{2}{\mu}\tilde{e}_k$ and that we can remove the positive $a_{(k+1)m}$ term from the left-hand-side of (43), this gives us our final recursion inequality:

$$\big(2\gamma m - 4L\gamma^2 C_1 m - 4L\gamma^2 \tau C_2 m\big)\tilde{e}_{k+1} \leq \big(\frac{2}{\mu} + 4L\gamma^2 C_1 m + 4L\gamma^2\tau C_2 m\big)\tilde{e}_k \tag{45}$$

Replacing $C_1$ and $C_2$ by their values (defined in 17) in (45) directly leads to Theorem 15.

**Algorithm 5** Hogwild

```
1:  Initialize shared variable $x$
2:  keep doing in parallel
3:      $\hat{x}$ = inconsistent read of $x$
4:      Sample $i$ uniformly in $\{1, ..., n\}$
5:      Let $S_i$ be $f_i$'s support
6:      $[\delta x]_{S_i} := -\gamma f_i'(\hat{x})$
7:      for $v$ in $S_i$ do
8:          $[x]_v \leftarrow [x]_v + [\delta x]_v$        // atomic
9:      end for
10: end parallel loop
```

## 5. Hogwild Analysis

In order to show that our improved "after read" perturbed iterate framework can be used to revisit the analysis of other optimization routines with correct proofs that do not assume homogeneous computation, we now provide the analysis of the Hogwild algorithm (i.e. asynchronous parallel constant step size Sgd) first introduced in Niu et al. (2011).

We start by describing Hogwild in Algorithm 5, and then give our theoretical convergence and speedup results and their proofs. Note that our framework allows us to easily remove the classical bounded gradient assumption, which is used in one form or another in most of the literature (Niu et al., 2011; De Sa et al., 2015; Mania et al., 2017) – although it is inconsistent with strong convexity in the unconstrained regime. This allows for better bounds where the uniform bound on $\|f_i'(x)\|^2$ is replaced by its variance at the optimum.

### 5.1. Theoretical Results

We now state the theoretical results of our analysis of Hogwild with inconsistent reads and writes in the "after read framework". We give an outline of the proof in Section 5.2 and its full details in Appendix D. We start with a useful definition.

**Definition 20** *Let $\sigma^2 = \mathbf{E}\|f_i'(x^*)\|^2$ be the variance of the gradient estimator at the optimum.*

For reference, we start by giving the rate of convergence of serial Sgd (see e.g. Schmidt, 2014).

**Theorem 21 (Convergence guarantee and rate of Sgd)** *Let $a \le \frac{1}{2}$. Then for any step size $\gamma = \frac{a}{L}$, Sgd converges in expectation to b-accuracy at a geometric rate of at least: $\rho(a) = a/\kappa$, i.e., $\mathbb{E}\|x_t - x^*\|^2 \le (1-\rho)^t \|x_0 - x^*\|^2 + b$, where $b = 2\frac{\gamma\sigma^2}{\mu}$.*

As Sgd only converges linearly up to a ball around the optimum, to make sure we reach $\epsilon$-accuracy, it is necessary that $\frac{2\gamma\sigma^2}{\mu} \le \epsilon$, i.e. $\gamma \le \frac{\epsilon\mu}{2\sigma^2}$. All told, in order to get linear convergence to $\epsilon$-accuracy, serial Sgd requires $\gamma \le \min\left\{\frac{1}{2L}, \frac{\epsilon\mu}{2\sigma^2}\right\}$. The proof can be found in Appendix D.3.

**Theorem 22 (Convergence guarantee and rate of Hogwild)** *Let*

$$a^*(\tau) := \frac{1}{5\left(1 + 2\tau\sqrt{\Delta}\right)\xi(\kappa, \Delta, \tau)} \qquad \text{where } \xi(\kappa, \Delta, \tau) := \sqrt{1 + \frac{1}{2\kappa}\min\{\frac{1}{\sqrt{\Delta}}, \tau\}} \qquad (46)$$

*(note that $\xi(\kappa, \Delta, \tau) \approx 1$ unless $\kappa < 1/\sqrt{\Delta}$ $(\leq \sqrt{n})$).*

*For any step size $\gamma = \frac{a}{L}$ with $a \leq \min\left\{a^*(\tau), \frac{\kappa}{\tau}\right\}$, the inconsistent read iterates of Algorithm 5 converge in expectation to b-accuracy at a geometric rate of at least: $\rho(a) = a/\kappa$, i.e., $\mathbb{E}\|\hat{x}_t - x^*\|^2 \leq (1-\rho)^t(2\|x_0 - x^*\|^2) + b$, where $b = (\frac{8\gamma(C_1 + \tau C_2)}{\mu} + 4\gamma^2 C_1 \tau)\sigma^2$ and $C_1$ and $C_2(\gamma)$ are defined in* (17).

Once again this result is quite close to the one obtained for serial Sgd. Note that we recover this exact condition (up to a small constant factor) if we set $\tau = 0$, i.e. if we force our asynchronous algorithm to be serial.

The condition $a \leq \frac{\kappa}{\tau}$ is equivalent to $\gamma\mu\tau \leq 1$ and should be thought of as a condition on $\tau$. We will see that it is always verified in the regime we are interested in, that is the linear speed-up regime (where more stringent conditions are imposed on $\tau$).

We now investigate the conditions under which Hogwild is linearly faster than Sgd. Note that to derive these conditions we need not only compare their respective convergence rates, but also the size of the ball around the optimum to which both algorithms converge. These quantities are provided in Theorems 21 and 22.

**Corollary 23 (Speedup condition)** *Suppose $\tau = \mathcal{O}(\min\{\frac{1}{\sqrt{\Delta}}, \kappa\})$. Then for any step size $\gamma \leq \frac{a^*(\tau)}{L} = \mathcal{O}(\frac{1}{L})$ (i.e., any allowable step size for* Sgd*),* Hogwild *converges geometrically to a ball of radius $r_h = \mathcal{O}(\frac{\gamma\sigma^2}{\mu})$ with rate factor $\rho = \frac{\gamma\mu}{2}$ (similar to* Sgd*), and is thus linearly faster than its sequential counterpart up to a constant factor.*

*Moreover, a universal step size of $\Theta(\frac{1}{L})$ can be used for* Hogwild *to be adaptive to local strong convexity with a similar rate to* Sgd *(i.e., knowledge of $\kappa$ is not required).*

If $\gamma = \mathcal{O}(1/L)$, Hogwild obtains the same convergence rate as Sgd and converges to a ball of equivalent radius. Since the maximum step size guaranteeing linear convergence for Sgd is also $\mathcal{O}(1/L)$, Hogwild is linearly faster than Sgd for any reasonable step size – under the condition that $\tau = \mathcal{O}(\min\{\frac{1}{\sqrt{\Delta}}, \kappa\})$. We also remark that since $\gamma \leq 1/L$ and $\tau \leq \kappa$, we have $\gamma\mu\tau \leq 1$, which means the condition $a \leq \frac{\kappa}{\tau}$ is superseded by $a \leq a^*(\tau)$ in Theorem 22.

We note that the condition on $\tau$ is much more restrictive if the condition number is small than for Asaga and Kromagnon. This can be explained by the fact that both Saga and Svrg have a composite rate factor which is not directly proportional to the step size. As a result, in the well-conditioned setting these algorithms enjoy a range of step sizes that all give the same contraction rate. This allows their asynchronous variants to use smaller step sizes while maintaining linear speedups. Sgd, on the other hand, has a rate factor that is directly proportional to its step size, hence the more restrictive condition on $\tau$.

***Function values results.*** Our results are derived directly on iterates, that is, we bound the distance between $\hat{x}_t$ and $x^*$. We can easily obtain results on function values to bound $\mathbb{E}f(\hat{x}_t) - f(x^*)$ by adapting the classical smoothness inequality:[22] $\mathbb{E}f(x_t) - f(x^*) \leq \frac{L}{2}\mathbb{E}\|x_t - x^*\|^2$ to the asynchronous parallel setting.

***Convergence to $\epsilon$-accuracy.*** As noted in Mania et al. (2017), for our algorithm to converge to $\epsilon$-accuracy for some $\epsilon > 0$, we require an additional bound on the step size to make sure that the radius of the ball to which we converge is small enough. For Sgd, this means using a step size $\gamma = \mathcal{O}(\frac{\epsilon\mu}{\sigma^2})$ (see Appendix D.3). We can also prove that under the conditions that $\tau = \mathcal{O}(\frac{1}{\sqrt{\Delta}})$ and $\gamma\mu\tau \leq 1$, Hogwild requires the same bound on the step size to converge to $\epsilon$-accuracy (see Appendix D.4).

If $\epsilon$ is small enough, the active upper bound on the step size is $\gamma = \mathcal{O}(\frac{\epsilon\mu}{\sigma^2})$ for both algorithms. In this regime, we obtain a relaxed condition on $\tau$ for a linear speedup. The condition $\tau \leq \kappa$ which came from comparing maximum allowable step sizes is removed. Instead, we enforce $\gamma\mu\tau \leq 1$, which gives us the weaker condition $\tau = \mathcal{O}(\frac{\sigma^2}{\epsilon\mu^2})$. Our condition on the overlap is then: $\tau = \mathcal{O}(\min\{\frac{1}{\sqrt{\Delta}}, \frac{\sigma^2}{\epsilon\mu^2}\})$. We see that this is similar to the condition obtained by Mania et al. (2017, Theorem 4) in their Hogwild analysis, although we have the variance at the optimum $\sigma^2$ instead of a squared global bound on the gradient.

***Comparison to related work.***

- We give the first convergence analysis for Hogwild with no assumption on a global bound on the gradient ($M$). This allows us to replace the usual dependence in $M^2$ by a term in $\sigma^2$ which is potentially significantly smaller. This means improved upper bounds on the step size and the allowed overlap.
- We obtain the same condition on the step size for linear convergence to $\epsilon$-accuracy of Hogwild as previous analysis for serial Sgd (e.g. Needell et al., 2014) – given $\tau \leq {}^1\!/\!_{\gamma\mu}$.
- In contrast to the Hogwild analysis from Niu et al. (2011); De Sa et al. (2015), our proof technique handles inconsistent reads and a non-uniform processing speed across $f_i$'s. Further, Corollary 23 gives a better dependence on the sparsity than in Niu et al. (2011), where $\tau \leq \mathcal{O}(\Delta^{-1/4})$, and does not require various bounds on the gradient assumptions.
- In contrast to the Hogwild analysis from Mania et al. (2017, Thm. 3), removing their gradient bound assumption enables us to get a (potentially) significantly better upper bound condition on $\tau$ for a linear speedup. We also give our convergence guarantee on $\hat{x}_t$ *during* the algorithm, whereas they only bound the error for the "last" iterate $x_T$.

## 5.2. Proof of Theorem 22 and Corollary 23

Here again, our proof technique begins as our Asaga analysis, with Properties 3, 4, 5 also verified for Hogwild[23]. As in our Asaga analysis, we make Assumption 6. Consequently, the basic recursive contraction inequality (15) and Lemma 10 also hold. As for Kromagnon, the proof diverges when we derive the equivalent of Lemma 13.

22. See e.g. Moulines and Bach (2011).
23. Once again, in our analysis the Hogwild algorithms reads the parameters before sampling, so that Property 3 is verified for $r = t$.

**Lemma 24 (Suboptimality bound on $\mathbb{E}\|g_t\|^2$)** *For all $t \geq 0$,*

$$\mathbb{E}\|g_t\|^2 \leq 4Le_t + 2\sigma^2 \,. \tag{47}$$

We give the proof in Appendix D.1. To derive both terms, we use the same technique as for the first term of Lemma 13. This result is simpler than both Lemma 13 (for Asaga) and Lemma 19 (for Kromagnon). The second term in this case does not even vanish as $t$ grows. This reflects the fact that constant step size Sgd does not converge to the optimum but rather to a ball around it. However, this simpler form allows us to simply unroll the resulting master inequality to get our convergence result.

The rest of the proof is as follows:

1. By substituting Lemma 24 into Lemma 10, we get a master contraction inequality (48) in terms of $a_{t+1}$, $a_t$, $(e_u, u \leq t)$ and $\sigma^2$.

2. We then unroll this master inequality and cleverly regroup terms to obtain a contraction inequality (49) between $a_t$, $a_0$ and $\sigma^2$.

3. By using that $\|\hat{x}_t - x^*\|^2 \leq 2a_t + 2\|\hat{x}_t - x_t\|^2$, we obtain a contraction inequality directly on the "real" iterates (as opposed to the "virtual" iterates as in Mania et al., 2017), subject to a maximum step size condition on $\gamma$. This finishes the proof for Theorem 22.

4. Finally, we only have to derive the conditions on $\gamma$ and $\tau$ under which Hogwild converges with a similar convergence rate to a ball with a similar radius than serial Sgd to finish the proof for Corollary 23.

We list the key points below with their proof sketch, and give the detailed proof in Appendix D.

***Master inequality.*** As in our Asaga analysis, we plug (47) in Lemma 10, which gives us that (see Appendix D.2):

$$a_{t+1} \leq (1 - \frac{\gamma\mu}{2})a_t + (4L\gamma^2 C_1 - 2\gamma)e_t + 4L\gamma^2 C_2 \sum_{u=(t-\tau)_+}^{t-1} e_u + 2\gamma^2\sigma^2(C_1 + \tau C_2) \,. \tag{48}$$

***Contraction inequality on $x_t$.*** As we previously mentioned, the term in $\sigma^2$ does not vanish so we cannot use either our Asaga or our Kromagnon proof technique. Instead, we unroll Equation (48) all the way to $t = 0$. This gives us (see Appendix D.2):

$$a_{t+1} \leq (1 - \frac{\gamma\mu}{2})^{t+1} a_0 + (4L\gamma^2 C_1 + 8L\gamma^2 \tau C_2 - 2\gamma) \sum_{u=0}^{t} (1 - \frac{\gamma\mu}{2})^{t-u} e_u + \frac{4\gamma\sigma^2}{\mu}(C_1 + \tau C_2) \,. \tag{49}$$

***Contraction inequality on $\hat{x}_t$.*** We now use that $\|\hat{x}_t - x^*\|^2 \leq 2a_t + 2\|\hat{x}_t - x_t\|^2$ together with our previous bound (65). Together with (49), we get (see Appendix D.2):

$$\mathbb{E}\|\hat{x}_t - x^*\|^2 \leq (1 - \frac{\gamma\mu}{2})^{t+1} 2a_0 + \Big(\frac{8\gamma(C_1 + \tau C_2)}{\mu} + 4\gamma^2 C_1 \tau\Big)\sigma^2$$
$$+ (24L\gamma^2 C_1 + 16L\gamma^2 \tau C_2 - 4\gamma) \sum_{u=0}^{t} (1 - \frac{\gamma\mu}{2})^{t-u} e_u \,. \quad (50)$$

To get our final contraction inequality, we need to safely remove all the $e_u$ terms, so we enforce $16L\gamma^2 C_1 + 16L\gamma^2\tau C_2 - 4\gamma \leq 0$. This leads directly to Theorem 22.

***Convergence rate and ball-size comparison.*** To prove Corollary 23, we simply show that under the condition $\tau = \mathcal{O}(\min\{\frac{1}{\sqrt{\Delta}}, \kappa\})$, the biggest allowable step size for Hogwild to converge linearly is $\mathcal{O}(1/L)$, as is also the case for Sgd; and that the size of the ball to which both algorithms converge is of the same order. The proof is finished by remarking that for both algorithms, the rates of convergence are directly proportional to the step size.

## 6. Empirical Results

We now present the results of our experiments. We first compare our new sequential algorithm, Sparse Saga, to its existing alternative, Saga with lagged updates and to the original Saga algorithm as a baseline. We then move on to our main results, the empirical comparison of Asaga, Kromagnon and Hogwild. Finally, we present additional results, including convergence and speedup figures with respect to the number of iteration (i.e. "theoretical speedups") and measures on the $\tau$ constant.

### 6.1. Experimental Setup

***Models.*** Although Asaga can be applied more broadly, we focus on logistic regression, a model of particular practical importance. The associated objective function takes the following form:

$$\frac{1}{n}\sum_{i=1}^{n} \log\big(1 + \exp(-b_i a_i^\intercal x)\big) + \frac{\mu}{2}\|x\|^2, \quad (51)$$

where $a_i \in \mathbb{R}^d$ and $b_i \in \{-1, +1\}$ are the data samples.

***Data sets.*** We consider two sparse data sets: RCV1 (Lewis et al., 2004) and URL (Ma et al., 2009); and a dense one, Covtype (Collobert et al., 2002), with statistics listed in the table below. As in Le Roux et al. (2012), Covtype is standardized, thus 100% dense. $\Delta$ is $\mathcal{O}(1)$ in all data sets, hence not very insightful when relating it to our theoretical results. Deriving a less coarse sparsity bound remains an open problem.

***Hardware and software.*** Experiments were run on a 40-core machine with 384GB of memory. All algorithms were implemented in Scala. We chose this high-level language despite its typical 20x slowdown compared to C (when using standard libraries, see Appendix F.2) because our primary concern was that the code may easily be reused and extended for research purposes (to this end, we have made all our code available at `http://www.di.ens.fr/sierra/research/asaga/`).

Table 1: Basic data set statistics.

| | $n$ | $d$ | density | $L$ |
|---|---|---|---|---|
| **RCV1** | 697,641 | 47,236 | 0.15% | 0.25 |
| **URL** | 2,396,130 | 3,231,961 | 0.004% | 128.4 |
| **Covtype** | 581,012 | 54 | 100% | 48428 |

### 6.2. Implementation Details

**Regularization.** Following Schmidt et al. (2016), the amount of regularization used was set to $\mu = 1/n$. In each update, we project the gradient of the regularization term (we multiply it by $D_i$ as we also do with the vector $\bar{\alpha}$) to preserve the sparsity pattern while maintaining an unbiased estimate of the gradient. For squared $\ell_2$, the Sparse Saga updates becomes:

$$x^+ = x - \gamma(f_i'(x) - \alpha_i + D_i\bar{\alpha} + \mu D_i x). \tag{52}$$

**Comparison with the theoretical algorithm.** The algorithm we used in the experiments is fully detailed in Algorithm 2. There are two differences with Algorithm 1. First, in the implementation we choose $i_t$ at random *before* we read the feature vector $a_{i_t}$. This enables us to only read the necessary data for a given iteration (i.e. $[\hat{x}_t]_{S_i}, [\hat{\alpha}_i^t], [\bar{\alpha}^t]_{S_i}$). Although this violates Property 3, it still performs well in practice.

Second, we maintain $\bar{\alpha}^t$ in memory. This saves the cost of recomputing it at every iteration (which we can no longer do since we only read a subset data). Again, in practice the implemented algorithm enjoys good performance. But this design choice raises a subtle point: the update is not guaranteed to be unbiased in this setup (see Appendix F.3 for more details).

***Step sizes.*** For each algorithm, we picked the best step size among 10 equally spaced values in a grid, and made sure that the best step size was never at the boundary of this interval. For Covtype and RCV1, we used the interval $[\frac{1}{10L}, \frac{10}{L}]$, whereas for URL we used the interval $[\frac{1}{L}, \frac{100}{L}]$ as it admitted larger step sizes. It turns out that the best step size was fairly constant for different number of cores for both Asaga and Kromagnon, and both algorithms had similar best step sizes (0.7 for RCV1, 0.05 for URL and $5 \times 10^{-5}$ for Covtype).

### 6.3. Comparison of Sequential Algorithms: Sparse Saga vs Lagged updates

We compare the Sparse Saga variant proposed in Section 3.1 to two other approaches: the naive (i.e., dense) update scheme and the lagged updates implementation described in Defazio et al. (2014). Note that we use different datasets from the parallel experiments, including a subset of the RCV1 data set and the Realsim data set (see description in Appendix F.1). Figure 2 reveals that sparse and lagged updates have a lower cost per iteration than their dense counterpart, resulting in faster convergence for sparse data sets. Furthermore, while the two approaches had similar convergence in terms of number of iterations, the Sparse Saga scheme is slightly faster in terms of runtime (and as previously pointed out, sparse updates are better adapted for the asynchronous setting). For the dense data set (Covtype), the three approaches exhibit similar performance.

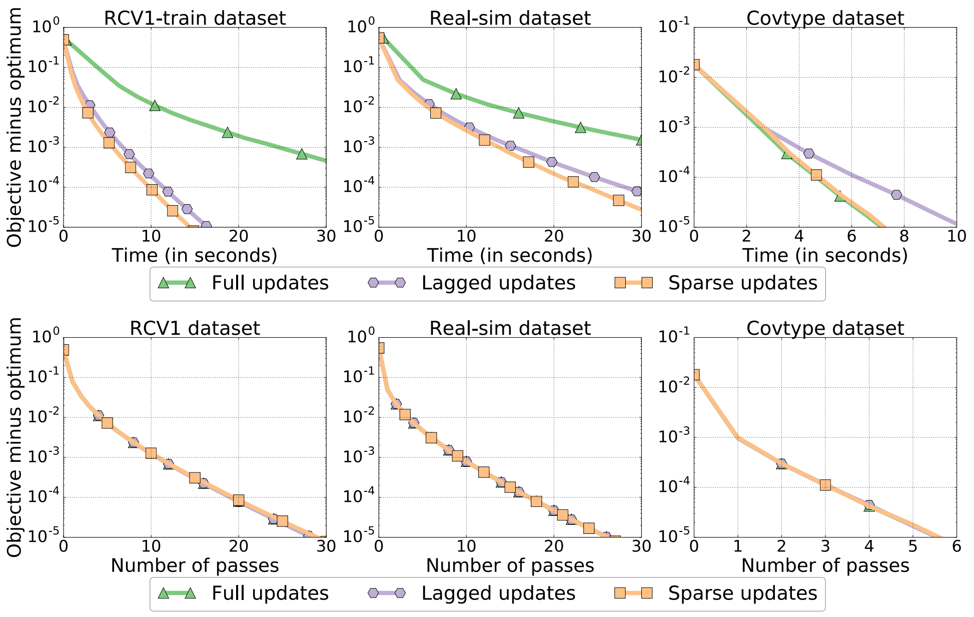
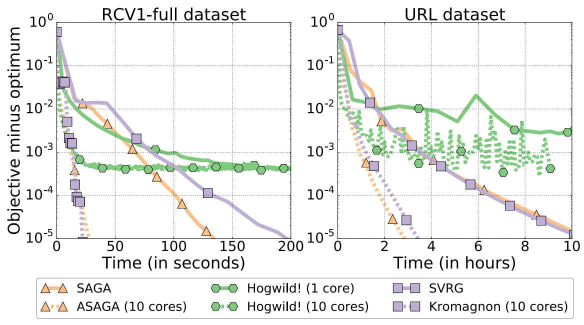
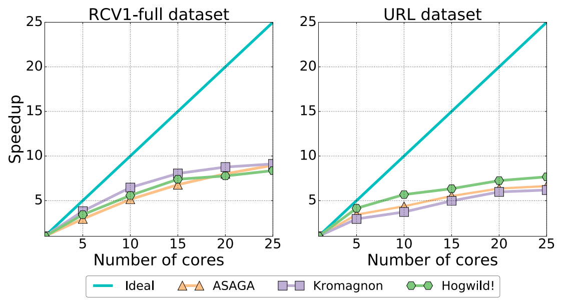


Figure 2: **Lagged vs Sparse Saga updates**. Suboptimality with respect to time for different Saga update schemes on various data sets. First row: suboptimality as a function of time. Second row: suboptimality as a the number of passes over the data set. For sparse data sets (RCV1 and Real-sim), lagged and sparse updates have a lower cost per iteration which result in faster convergence.



(a) Suboptimality as a function of time.



(b) Speedup as a function of the number of cores

Figure 3: **Convergence and speedup for asynchronous stochastic gradient descent methods**. We display results for RCV1 and URL. Results for Covtype can be found in Section 6.6.

## 6.4. Asaga vs. Kromagnon vs. Hogwild

We compare three different asynchronous variants of stochastic gradient methods on the aforementioned data sets: Asaga, presented in this work, Kromagnon, the asynchronous sparse Svrg method described in Mania et al. (2017) and Hogwild (Niu et al., 2011). Each method had its step size chosen so as to give the fastest convergence (up to a suboptimality of $10^{-3}$ in the special case of Hogwild). The results can be seen in Figure 3a: for each method we consider its asynchronous version with both one (hence sequential) and ten processors. This figure reveals that the asynchronous version offers a significant speedup over its sequential counterpart.

We then examine the speedup relative to the increase in the number of cores. The speedup is measured as time to achieve a suboptimality of $10^{-5}$ ($10^{-3}$ for Hogwild) with one core divided by time to achieve the same suboptimality with several cores, averaged over 3 runs. Again, we choose step size leading to fastest convergence[24] (see Appendix F.2 for information about the step sizes). Results are displayed in Figure 3b.

As predicted by our theory, we observe linear "theoretical" speedups (i.e. in terms of number of iterations, see Section 6.6). However, with respect to running time, the speedups seem to taper off after 20 cores. This phenomenon can be explained by the fact that our hardware model is by necessity a simplification of reality. As noted in Duchi et al. (2015), in a modern machine there is no such thing as *shared memory*. Each core has its own levels of cache (L1, L2, L3) in addition to RAM. These faster pools of memory are fully leveraged when using a single core. Unfortunately, as soon as several cores start writing to common locations, cache coherency protocols have to be deployed to ensure that the information is consistent across cores. These protocols come with computational overheads. As more and more cores are used, the shared information goes lower and lower in the memory stack, and the overheads get more and more costly. It may be the case that on much bigger data sets, where the cache memory is unlikely to provide benefits even for a single core (since

24. Although we performed grid search on a large interval, we observed that the best step size was fairly constant for different number of cores, and similar for Asaga and Kromagnon.

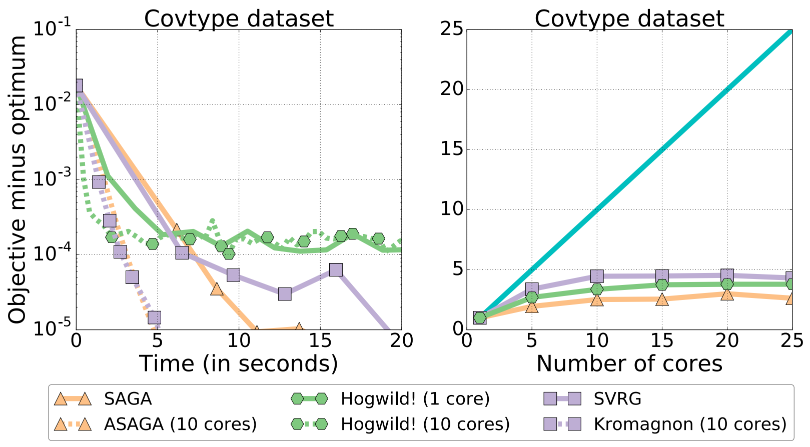


Figure 4: Comparison on the Covtype data set. Left: suboptimality. Right: speedup. The number of cores in the legend only refers to the left plot.

sampling items repeatedly becomes rare), the running time speedups actually improve. More experimentation is needed to quantify these effects and potentially increase performance.

### 6.5. Effect of Sparsity

Sparsity plays an important role in our theoretical results, where we find that while it is necessary in the "ill-conditioned" regime to get linear speedups, it is not in the "well-conditioned" regime. We confront this to real-life experiments by comparing the convergence and speedup performance of our three asynchronous algorithms on the Covtype data set, which is fully dense after standardization. The results appear in Figure 4.

While we still see a significant improvement in speed when increasing the number of cores, this improvement is smaller than the one we observe for sparser data sets. The speedups we observe are consequently smaller, and taper off earlier than on our other data sets. However, since the observed "theoretical" speedup is linear (see Section 6.6), we can attribute this worse performance to higher hardware overhead. This is expected because each update is fully dense and thus the shared parameters are much more heavily contended for than in our sparse datasets.

One thing we notice when computing the $\Delta$ constant for our data sets is that it often fails to capture the full sparsity distribution, being essentially a maximum: for all three data sets, we obtain $\Delta = \mathcal{O}(1)$. This means that $\Delta$ can be quite big even for very sparse data sets. Deriving a less coarse bound remains an open problem.

### 6.6. Theoretical Speedups

In the previous experimental sections, we have shown experimental speedup results where suboptimality was a function of the running time. This measure encompasses both theoretical algorithmic properties and hardware overheads (such as contention of shared memory) which are not taken into account in our analysis.

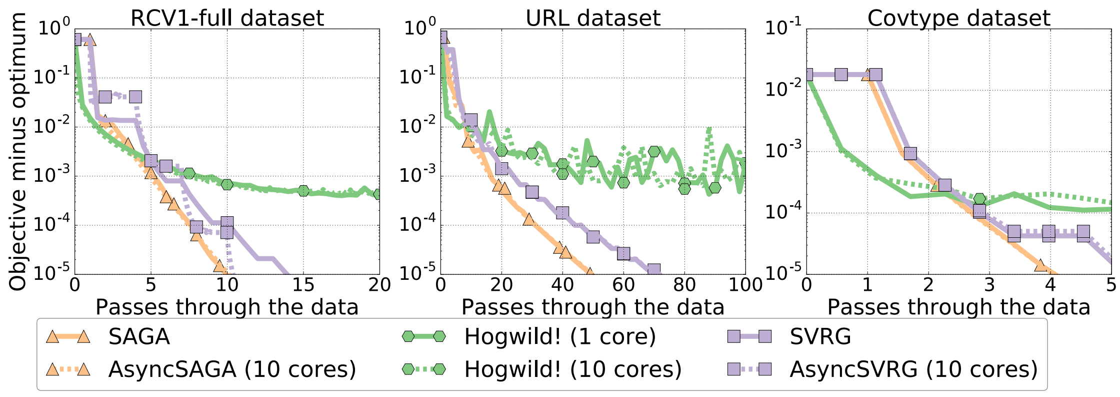


Figure 5: **Theoretical speedups**. Suboptimality with respect to number of iterations for Asaga, Kromagnon and Hogwild with 1 and 10 cores. Curves almost coincide, which means the theoretical speedup is almost the number of cores $p$, hence linear.

In order to isolate these two effects, we now plot our convergence experiments where suboptimality is a function of the number of iterations; thus, we abstract away any potential hardware overhead.[25] The experimental results can be seen in Figure 5.

For all three algorithms and all three data sets, the curves for 1 and 10 cores almost coincide, which means that we are indeed in the "theoretical linear speedup" regime. Indeed, when we plotted the amount of iterations required to converge to a given accuracy as a function of the number of cores, we obtained straight horizontal lines for our three algorithms.

The fact that the speedups we observe in running time are less than linear can thus be attributed to various hardware overheads, including shared variable contention – the compare-and-swap operations are more and more expensive as the number of competing requests increases – and cache effects as mentioned in Section 6.4.

## 6.7. A Closer Look at the $\tau$ Constant

### 6.7.1. Theory

In the parallel optimization literature, $\tau$ is often referred to as a proxy for the number of cores. However, intuitively as well as in practice, it appears that there are a number of other factors that can influence this quantity. We will now attempt to give a few qualitative arguments as to what these other factors might be and how they relate to $\tau$.

***Number of cores.*** The first of these factors is indeed the number of cores. If we have $p$ cores, $\tau \geq p - 1$. Indeed, in the best-case scenario where all cores have exactly the same execution speed for a single iteration, $\tau = p - 1$.

***Length of an iteration.*** To get more insight into what $\tau$ really encompasses, let us now try to define the worst-case scenario in the preceding example. Consider 2 cores. In the worst

25. To do so, we implement a global counter which is sparsely updated (every 100 iterations for example) in order not to modify the asynchrony of the system. This counter is used only for plotting purposes and is not needed otherwise.

case, one core runs while the other is stuck. Then the overlap is $t$ for all $t$ and eventually grows to $+\infty$. If we assume that one core runs twice as fast as the other, then $\tau = 2$. If both run at the same speed, $\tau = 1$.

It appears then that a relevant quantity is $R$, the ratio between the fastest execution time and the slowest execution time for a single iteration. We have $\tau \leq (p-1)R$, which can be arbitrarily bigger than $p$.

There are several factors at play in $R$ itself. These include:

- the speed of execution of the cores themselves (i.e. clock time).
- the data matrix itself. Different support sizes for $f_i$ means different gradient computation times. If one $f_i$ has support of size $n$ while all the others have support of size 1 for example, $R$ may eventually become very big.
- the length of the computation itself. The longer our algorithm runs, the more likely it is to explore the potential corner cases of the data matrix.

The overlap is then upper bounded by the number of cores multiplied by the ratio of the maximum iteration time over the minimum iteration time (which is linked to the sparsity distribution of the data matrix). This is an upper bound, which means that in some cases it will not really be useful. For example, if one factor has support size 1 and all others have support size $d$, the probability of the event which corresponds to the upper bound is exponentially small in $d$. We conjecture that a more useful indicator could be ratio of the maximum iteration time over the expected iteration time.

To sum up this preliminary theoretical exploration, the $\tau$ term encompasses much more complexity than is usually implied in the literature. This is reflected in the experiments we ran, where the constant was orders of magnitude bigger than the number of cores.

### 6.7.2. Experimental Results

In order to verify our intuition about the $\tau$ variable, we ran several experiments on all three data sets, whose characteristics are reminded in Table 2. $\delta_l^i$ is the support size of $f_i$.

Table 2: Density measures including minimum, average and maximum support size $\delta_l^i$ of the factors.

| | $n$ | $d$ | density | $\max(\delta_l^i)$ | $\min(\delta_l^i)$ | $\bar{\delta}_l$ | $\max(\delta_l^i)/\bar{\delta}_l$ |
|---|---:|---:|---:|---:|---:|---:|---:|
| **RCV1** | 697,641 | 47,236 | 0.15% | 1,224 | 4 | 73.2 | 16.7 |
| **URL** | 2,396,130 | 3,231,961 | 0.003% | 414 | 16 | 115.6 | 3.58 |
| **Covtype** | 581,012 | 54 | 100% | 12 | 8 | 11.88 | 1.01 |

To estimate $\tau$, we compute the average overlap over 100 iterations, i.e. the difference in labeling between the end of the hundredth iteration and the start of the first iteration on, divided by 100. This quantity is a lower bound on the actual overlap (which is a maximum, not an average). We then take its maximum observed value. The reason why we use an average is that computing the overlap requires using a global counter, which we do not want

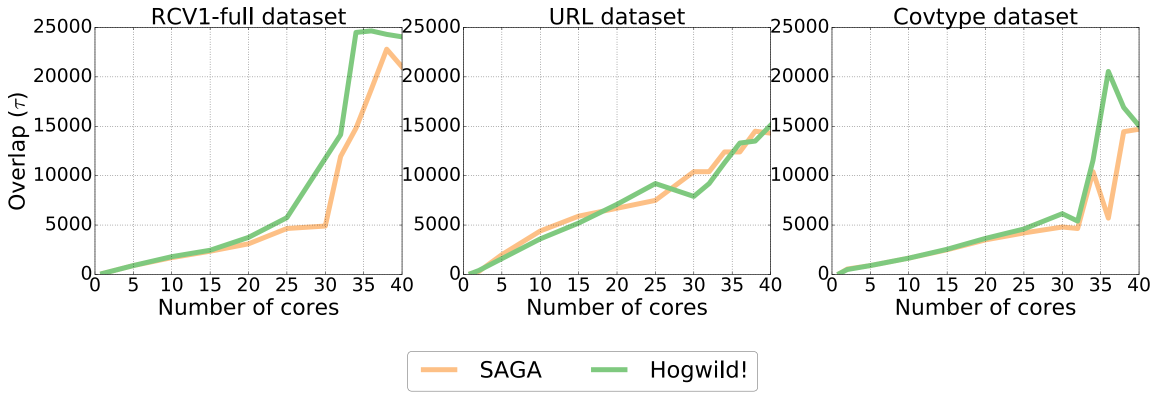


Figure 6: **Overlap**. Overlap as a function of the number of cores for both Asaga and Hogwild on all three data sets.

to update every iteration since it would make it a heavily contentious quantity susceptible of artificially changing the asynchrony pattern of our algorithm.

The results we observe are order of magnitude bigger than $p$, indicating that $\tau$ can indeed not be dismissed as a mere proxy for the number of cores, but has to be more carefully analyzed.

First, we plot the maximum observed $\tau$ as a function of the number of cores (see Figure 6). We observe that the relationship does indeed seem to be roughly linear with respect to the number of cores until 30 cores. After 30 cores, we observe what may be a phase transition where the slope increases significantly.

Second, we measured the maximum observed $\tau$ as a function of the number of epochs. We omit the figure since we did not observe any dependency; that is, $\tau$ does not seem to depend on the number of epochs. We know that it must depend on the number of iterations (since it cannot be bigger, and is an increasing function with respect to that number for example), but it appears that a stable value is reached quite quickly (before one full epoch is done).

If we allowed the computations to run forever, we would eventually observe an event such that $\tau$ would reach the upper bound mentioned in Section 6.7.1, so it may be that $\tau$ is actually a very slowly increasing function of the number of iterations.

## 7. Conclusions and Future Work

Building on the recently proposed “perturbed iterate” framework, we have proposed a novel perspective to clarify an important technical issue present in a large fraction of the recent convergence rate proofs for asynchronous parallel optimization algorithms. To resolve it, we have introduced a novel “after read” framework and demonstrated its usefulness by analyzing three asynchronous parallel incremental optimization algorithms, including Asaga, a novel sparse and fully asynchronous variant of the incremental gradient algorithm Saga. Our proof technique accommodates more realistic settings than is usually the case in the literature (such as inconsistent reads and writes and an unbounded gradient); we obtain tighter conditions than in previous work. In particular, we show that Asaga is linearly faster than Saga under mild conditions, and that sparsity is not always necessary to get linear speedups. Our empirical benchmarks confirm speedups up to 10x.

Schmidt et al. (2016) have shown that Sag enjoys much improved performance when combined with non-uniform sampling and line-search. We have also noticed that our $\Delta_r$ constant (being essentially a maximum) sometimes fails to accurately represent the full sparsity distribution of our data sets. Finally, while our algorithm can be directly ported to a distributed master-worker architecture, its communication pattern would have to be optimized to avoid prohibitive costs. Limiting communications can be interpreted as artificially increasing the delay, yielding an interesting trade-off between delay influence and communication costs.

These constitute interesting directions for future analysis, as well as a further exploration of the $\tau$ term, which we have shown encompasses more complexity than previously thought.

## Acknowledgments

We would like to thank Xinghao Pan for sharing with us their implementation of Kromagnon, as well as Alberto Chiappa for spotting a typo in the proof. This work was partially supported by a Google Research Award and the MSR-Inria Joint Center. FP acknowledges financial support from the chaire *Économie des nouvelles données* with the *data science* joint research initiative with the *fonds AXA pour la recherche*.

***Appendix Outline:***

- In Appendix A, we adapt the proof from Hofmann et al. (2015) to prove Theorem 2, our convergence result for serial Sparse Saga.
- In Appendix B, we give the complete details for the proof of convergence for Asaga (Theorem 8) as well as its linear speedup regimes (Corollary 9).
- In Appendix C, we give the full details for the proof of convergence for Kromagnon (Theorem 15) as well as a simpler convergence result for both Svrg (Corollary 16) and Kromagnon (Corollary 17) and finally the latter's linear speedup regimes (Corollary 18)
- In Appendix D, we give the full details for the proof of convergence for Hogwild (Theorem 22) as well as its linear speedup regimes (Corollary 23).
- In Appendix E, we explain why adapting the lagged updates implementation of Saga to the asynchronous setting is difficult.
- In Appendix F, we give additional details about the data sets and our implementation.

## Appendix A. Sparse Saga – Proof of Theorem 2

***Proof sketch for Hofmann et al. (2015).*** As we will heavily reuse the proof technique from Hofmann et al. (2015), we start by giving its sketch.

First, the authors combine classical strong convexity and Lipschitz inequalities to derive the following inequality (Hofmann et al., 2015, Lemma 1):

$$\mathbf{E}\|x^+ - x^*\|^2 \leq (1-\gamma\mu)\|x-x^*\|^2 + 2\gamma^2\mathbf{E}\|\alpha_i - f_i'(x^*)\|^2 + (4\gamma^2 L - 2\gamma)\big(f(x) - f(x^*)\big). \quad (53)$$

This gives a contraction term, as well as two additional terms; $2\gamma^2\mathbf{E}\|\alpha_i - f_i'(x^*)\|^2$ is a positive variance term, but $(4\gamma^2 L - 2\gamma)\big(f(x) - f(x^*)\big)$ is a negative suboptimality term (provided $\gamma$ is small enough). The suboptimality term can then be used to cancel the variance one.

Second, the authors use a classical smoothness upper bound to control the variance term and relate it to the suboptimality. However, since the $\alpha_i$ are partial gradients computed at previous time steps, the upper bounds of the variance involve suboptimality at previous time steps, which are not directly relatable to the current suboptimality.

Third, to circumvent this issue, a Lyapunov function is defined to encompass both current and past terms. To finish the proof, Hofmann et al. (2015) show that the Lyapunov function is a contraction.

***Proof outline.*** Fortunately, we can reuse most of the proof from Hofmann et al. (2015) to show that Sparse Saga converges at the same rate as regular Saga. In fact, once we establish that Hofmann et al. (2015, Lemma 1) is still verified we are done.

To prove this, we show that the gradient estimator is unbiased, and then derive close variants of equations (6) and (9) in their paper, which we remind the reader of here:

$$\mathbf{E}\|f_i'(x) - \bar{\alpha}_i\|^2 \leq 2\mathbf{E}\|f_i'(x) - f_i'(x^*)\|^2 + 2\mathbf{E}\|\bar{\alpha}_i - f_i'(x^*)\|^2 \quad \text{Hofmann et al. (2015, Eq. 6)}$$

$$\mathbf{E}\|\bar{\alpha}_i - f_i'(x^*)\|^2 \leq \mathbf{E}\|\alpha_i - f_i'(x^*)\|^2 . \quad \text{Hofmann et al. (2015, Eq. 9)}$$

***Unbiased gradient estimator.*** We first show that the update estimator is unbiased. The estimator is unbiased if:

$$\mathbf{E} D_i \bar{\alpha} = \mathbf{E} \alpha_i = \frac{1}{n} \sum_{i=1}^{n} \alpha_i \,. \tag{54}$$

We have:

$$\mathbf{E} D_i \bar{\alpha} = \frac{1}{n} \sum_{i=1}^{n} D_i \bar{\alpha} = \frac{1}{n} \sum_{i=1}^{n} P_{S_i} D \bar{\alpha} = \frac{1}{n} \sum_{i=1}^{n} \sum_{v \in S_i} \frac{[\bar{\alpha}]_v e_v}{p_v} = \sum_{v=1}^{d} \left( \sum_{i \,|\, v \in S_i} 1 \right) \frac{[\bar{\alpha}]_v e_v}{n p_v} \,,$$

where $e_v$ is the vector whose only nonzero component is the $v$ component which is equal to 1.

By definition, $\sum_{i|v \in S_i} 1 = n p_v$, which gives us Equation (54).

***Deriving Hofmann et al. (2015, Equation 6).*** We define $\bar{\alpha}_i := \alpha_i - D_i \bar{\alpha}$ (contrary to Hofmann et al. (2015) where the authors define $\bar{\alpha}_i := \alpha_i - \bar{\alpha}$ since they do not concern themselves with sparsity). Using the inequality $\|a + b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$, we get:

$$\mathbf{E} \| f_i'(x) - \bar{\alpha}_i \|^2 \leq 2 \mathbf{E} \| f_i'(x) - f_i'(x^*) \|^2 + 2 \mathbf{E} \| \bar{\alpha}_i - f_i'(x^*) \|^2, \tag{55}$$

which is our equivalent to Hofmann et al. (2015, Eq. 6), where only our definition of $\bar{\alpha}_i$ differs.

***Deriving Hofmann et al. (2015, Equation 9).*** We want to prove Hofmann et al. (2015, Eq. 9):

$$\mathbf{E} \| \bar{\alpha}_i - f_i'(x^*) \|^2 \leq \mathbf{E} \| \alpha_i - f_i'(x^*) \|^2. \tag{56}$$

We have:

$$\mathbf{E} \| \bar{\alpha}_i - f_i'(x^*) \|^2 = \mathbf{E} \| \alpha_i - f_i'(x^*) \|^2 - 2 \mathbf{E} \langle \alpha_i - f_i'(x^*), D_i \bar{\alpha} \rangle + \mathbf{E} \| D_i \bar{\alpha} \|^2. \tag{57}$$

Let $D_{\neg i} := P_{S_i^c} D$; we then have the orthogonal decomposition $D\alpha = D_i \alpha + D_{\neg i} \alpha$ with $D_i \alpha \perp D_{\neg i} \alpha$, as they have disjoint support. We now use the orthogonality of $D_{\neg i} \alpha$ with any vector with support in $S_i$ to simplify the expression (57) as follows:

$$\begin{aligned}
\mathbf{E} \langle \alpha_i - f_i'(x^*), D_i \bar{\alpha} \rangle &= \mathbf{E} \langle \alpha_i - f_i'(x^*), D_i \bar{\alpha} + D_{\neg i} \bar{\alpha} \rangle && (\alpha_i - f_i'(x^*) \perp D_{\neg i} \bar{\alpha}) \\
&= \mathbf{E} \langle \alpha_i - f_i'(x^*), D \bar{\alpha} \rangle \\
&= \langle \mathbf{E} \big( \alpha_i - f_i'(x^*) \big), D \bar{\alpha} \rangle \\
&= \langle \mathbf{E} \alpha_i, D \bar{\alpha} \rangle && (f'(x^*) = 0) \\
&= \bar{\alpha}^\intercal D \bar{\alpha} \,. && (58)
\end{aligned}$$

Similarly,

$$\begin{aligned}
\mathbf{E} \| D_i \bar{\alpha} \|^2 &= \mathbf{E} \langle D_i \bar{\alpha}, D_i \bar{\alpha} \rangle \\
&= \mathbf{E} \langle D_i \bar{\alpha}, D \bar{\alpha} \rangle && (D_i \bar{\alpha} \perp D_{\neg i} \bar{\alpha}) \\
&= \langle \mathbf{E} D_i \bar{\alpha}, D \bar{\alpha} \rangle \\
&= \bar{\alpha}^\intercal D \bar{\alpha} \,. && (59)
\end{aligned}$$

Putting it all together,

$$\mathbf{E}\|\bar{\alpha}_i - f_i'(x^*)\|^2 = \mathbf{E}\|\alpha_i - f_i'(x^*)\|^2 - \bar{\alpha}^\intercal D\bar{\alpha} \leq \mathbf{E}\|\alpha_i - f_i'(x^*)\|^2. \tag{60}$$

This is our version of Hofmann et al. (2015, Equation 9), which finishes the proof of Hofmann et al. (2015, Lemma 1). The rest of the proof from Hofmann et al. (2015) can then be reused without modification to obtain Theorem 2. ■

## Appendix B. Asaga – Proof of Theorem 8 and Corollary 9

### B.1. Initial Recursive Inequality Derivation

We start by proving Equation (15). Let $g_t := g(\hat{x}_t, \hat{\alpha}^t, i_t)$. From (10), we get:

$$\begin{aligned}\|x_{t+1} - x^*\|^2 = \|x_t - \gamma g_t - x^*\|^2 &= \|x_t - x^*\|^2 + \gamma^2\|g_t\|^2 - 2\gamma\langle x_t - x^*, g_t\rangle \\ &= \|x_t - x^*\|^2 + \gamma^2\|g_t\|^2 - 2\gamma\langle \hat{x}_t - x^*, g_t\rangle + 2\gamma\langle \hat{x}_t - x_t, g_t\rangle.\end{aligned}$$

In order to prove Equation (15), we need to bound the $-2\gamma\langle \hat{x}_t - x^*, g_t\rangle$ term. Thanks to Property 3, we can write:

$$\mathbb{E}\langle \hat{x}_t - x^*, g_t\rangle = \mathbb{E}\langle \hat{x}_t - x^*, \mathbf{E}g_t\rangle = \mathbb{E}\langle \hat{x}_t - x^*, f'(\hat{x}_t)\rangle\,.$$

We can now use a classical strong convexity bound as well as a squared triangle inequality to get:

$$\begin{aligned}-\langle \hat{x}_t - x^*, f'(\hat{x}_t)\rangle &\leq -\big(f(\hat{x}_t) - f(x^*)\big) - \frac{\mu}{2}\|\hat{x}_t - x^*\|^2 && \text{(Strong convexity bound)}\\ -\|\hat{x}_t - x^*\|^2 &\leq \|\hat{x}_t - x_t\|^2 - \frac{1}{2}\|x_t - x^*\|^2 && (\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2)\\ -2\gamma\mathbb{E}\langle \hat{x}_t - x^*, g_t\rangle &\leq -\frac{\gamma\mu}{2}\mathbb{E}\|x_t - x^*\|^2 + \gamma\mu\mathbb{E}\|\hat{x}_t - x_t\|^2 - 2\gamma\big(\mathbb{E}f(\hat{x}_t) - f(x^*)\big)\,. && (61)\end{aligned}$$

Putting it all together, we get the initial recursive inequality (15), rewritten here explicitly:

$$a_{t+1} \leq (1 - \frac{\gamma\mu}{2})a_t + \gamma^2\mathbb{E}\|g_t\|^2 + \gamma\mu\mathbb{E}\|\hat{x}_t - x_t\|^2 + 2\gamma\mathbb{E}\langle \hat{x}_t - x_t, g_t\rangle - 2\gamma e_t\,, \tag{62}$$

where $a_t := \mathbb{E}\|x_t - x^*\|^2$ and $e_t := \mathbb{E}f(\hat{x}_t) - f(x^*)$.

### B.2. Proof of Lemma 10 (inequality in terms of $g_t := g(\hat{x}_t, \hat{\alpha}^t, i_t)$)

To prove Lemma 10, we now bound both $\mathbb{E}\|\hat{x}_t - x_t\|^2$ and $\mathbb{E}\langle \hat{x}_t - x_t, g_t\rangle$ with respect to $(\mathbb{E}\|g_u\|^2)_{u\leq t}$.

***Bounding*** $\mathbb{E}\langle \hat{x}_t - x_t, g_t \rangle$ ***in terms of*** $g_u$.

$$\begin{aligned}
\frac{1}{\gamma}\mathbb{E}\langle \hat{x}_t - x_t, g_t \rangle &= \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\langle G_u^t g_u, g_t \rangle && \text{(by Equation (11))} \\
&\leq \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}|\langle g_u, g_t \rangle| && \text{($G_u^t$ diagonal matrices with terms in $\{0,1\}$)} \\
&\leq \sum_{u=(t-\tau)_+}^{t-1} \frac{\sqrt{\Delta}}{2}(\mathbb{E}\|g_u\|^2 + \mathbb{E}\|g_t\|^2) && \text{(by Proposition 11)} \\
&\leq \frac{\sqrt{\Delta}}{2} \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\|g_u\|^2 + \frac{\sqrt{\Delta}\tau}{2}\mathbb{E}\|g_t\|^2. && (63)
\end{aligned}$$

***Bounding*** $\mathbb{E}\|\hat{x}_t - x_t\|^2$ ***with respect to*** $g_u$ Thanks to the expansion for $\hat{x}_t - x_t$ (11), we get:

$$\|\hat{x}_t - x_t\|^2 \leq \gamma^2 \sum_{u,v=(t-\tau)_+}^{t-1} |\langle G_u^t g_u, G_v^t g_v \rangle| \leq \gamma^2 \sum_{u=(t-\tau)_+}^{t-1} \|g_u\|^2 + \gamma^2 \sum_{\substack{u,v=(t-\tau)_+ \\ u\neq v}}^{t-1} |\langle G_u^t g_u, G_v^t g_v \rangle| \,.$$

Using (18) from Proposition 11, we have that for $u \neq v$:

$$\mathbb{E}|\langle G_u^t g_u, G_v^t g_v \rangle| \leq \mathbb{E}|\langle g_u, g_v \rangle| \leq \frac{\sqrt{\Delta}}{2}(\mathbb{E}\|g_u\|^2 + \mathbb{E}\|g_v\|^2)\,. \tag{64}$$

By taking the expectation and using (64), we get:

$$\begin{aligned}
\mathbb{E}\|\hat{x}_t - x_t\|^2 &\leq \gamma^2 \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\|g_u\|^2 + \gamma^2\sqrt{\Delta}(\tau-1)_+ \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\|g_u\|^2 \\
&= \gamma^2\big(1 + \sqrt{\Delta}(\tau-1)_+\big) \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\|g_u\|^2 \\
&\leq \gamma^2\big(1 + \sqrt{\Delta}\tau\big) \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\|g_u\|^2. && (65)
\end{aligned}$$

We can now rewrite (15) in terms of $\mathbb{E}\|g_t\|^2$, which finishes the proof for Lemma 10 (by introducing $C_1$ and $C_2$ as specified by 17 in Lemma 10):

$$\begin{aligned}
a_{t+1} &\leq (1 - \frac{\gamma\mu}{2})a_t - 2\gamma e_t + \gamma^2\mathbb{E}\|g_t\|^2 + \gamma^3\mu(1 + \sqrt{\Delta}\tau) \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\|g_u\|^2 \\
&\quad + \gamma^2\sqrt{\Delta} \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\|g_u\|^2 + \gamma^2\sqrt{\Delta}\tau\mathbb{E}\|g_t\|^2 \\
&\leq (1 - \frac{\gamma\mu}{2})a_t - 2\gamma e_t + \gamma^2 C_1\mathbb{E}\|g_t\|^2 + \gamma^2 C_2 \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\|g_u\|^2. && (66)
\end{aligned}$$

■

## B.3. Proof of Lemma 13 (suboptimality bound on $\mathbb{E}\|g_t\|^2$)

We now derive our bound on $g_t$ with respect to suboptimality. From Appendix A, we know that:

$$\mathbb{E}\|g_t\|^2 \leq 2\mathbb{E}\|f'_{i_t}(\hat{x}_t) - f'_{i_t}(x^*)\|^2 + 2\mathbb{E}\|\hat{\alpha}^t_{i_t} - f'_{i_t}(x^*)\|^2 \tag{67}$$

$$\mathbb{E}\|f'_{i_t}(\hat{x}_t) - f'_{i_t}(x^*)\|^2 \leq 2L\big(\mathbb{E}f(\hat{x}_t) - f(x^*)\big) = 2Le_t\,. \tag{68}$$

**N. B.: In the following, $i_t$ is a random variable picked uniformly at random in $\{1, ..., n\}$, whereas $i$ is a fixed constant.**

We still have to handle the $\mathbb{E}\|\hat{\alpha}^t_{i_t} - f'_{i_t}(x^*)\|^2$ term and express it in terms of past suboptimalities. We know from our definition of $t$ that $i_t$ and $\hat{x}_u$ are independent $\forall u < t$. Given the "after read" global ordering, $\mathbf{E}$ – the expectation on $i_t$ conditioned on $\hat{x}_t$ and all "past" $\hat{x}_u$ and $i_u$ – is well defined, and we can rewrite our quantity as:

$$\begin{aligned}\mathbb{E}\|\hat{\alpha}^t_{i_t} - f'_{i_t}(x^*)\|^2 = \mathbb{E}\big(\mathbf{E}\|\hat{\alpha}^t_{i_t} - f'_{i_t}(x^*)\|^2\big) &= \mathbb{E}\frac{1}{n}\sum_{i=1}^n \|\hat{\alpha}^t_i - f'_i(x^*)\|^2 \\ &= \frac{1}{n}\sum_{i=1}^n \mathbb{E}\|\hat{\alpha}^t_i - f'_i(x^*)\|^2.\end{aligned}$$

Now, with $i$ fixed, let $u^t_{i,l}$ be the time of the iterate last used to write the $[\hat{\alpha}^t_i]_l$ quantity, i.e. $[\hat{\alpha}^t_i]_l = [f'_i(\hat{x}_{u^t_{i,l}})]_l$. We know[26] that $0 \leq u^t_{i,l} \leq t-1$. To use this information, we first need to split $\hat{\alpha}_i$ along its dimensions to handle the possible inconsistencies among them:

$$\mathbb{E}\|\hat{\alpha}^t_i - f'_i(x^*)\|^2 = \mathbb{E}\sum_{l=1}^d \big([\hat{\alpha}^t_i]_l - [f'_i(x^*)]_l\big)^2 = \sum_{l=1}^d \mathbb{E}\Big[\big([\hat{\alpha}^t_i]_l - [f'_i(x^*)]_l\big)^2\Big].$$

This gives us:

$$\begin{aligned}\mathbb{E}\|\hat{\alpha}^t_i - f'_i(x^*)\|^2 &= \sum_{l=1}^d \mathbb{E}\Big[\big(f'_i(\hat{x}_{u^t_{i,l}})_l - f'_i(x^*)_l\big)^2\Big] \\ &= \sum_{l=1}^d \mathbb{E}\Big[\sum_{u=0}^{t-1} \mathbb{1}_{\{u^t_{i,l}=u\}}\big(f'_i(\hat{x}_u)_l - f'_i(x^*)_l\big)^2\Big] \\ &= \sum_{u=0}^{t-1}\sum_{l=1}^d \mathbb{E}\Big[\mathbb{1}_{\{u^t_{i,l}=u\}}\big(f'_i(\hat{x}_u)_l - f'_i(x^*)_l\big)^2\Big].\end{aligned} \tag{69}$$

We will now rewrite the indicator so as to obtain independent events from the rest of the equality. This will enable us to distribute the expectation. Suppose $u > 0$ ($u = 0$ is a special case which we will handle afterwards). $\{u^t_{i,l} = u\}$ requires two things:

26. In the case where $u = 0$, one would have to replace the partial gradient with $\alpha^0_i$. We omit this special case here for clarity of exposition.

1. at time $u$, $i$ was picked uniformly at random,
2. (roughly) $i$ was not picked again between $u$ and $t$.

We need to refine both conditions because we have to account for possible collisions due to asynchrony. We know from our definition of $\tau$ that the $t^{\text{th}}$ iteration finishes before at $t+\tau+1$, but it may still be unfinished by time $t+\tau$. This means that we can only be sure that an update selecting $i$ at time $v$ has been written to memory at time $t$ if $v \leq t-\tau-1$. Later updates may not have been written yet at time $t$. Similarly, updates before $v = u+\tau+1$ may be overwritten by the $u^{\text{th}}$ update so we cannot infer that they did not select $i$. From this discussion, we conclude that $u^t_{i,l} = u$ implies that $i_v \neq i$ for all $v$ between $u+\tau+1$ and $t-\tau-1$, though it can still happen that $i_v = i$ for $v$ outside this range.

Using the fact that $i_u$ and $i_v$ are independent for $v \neq u$, we can thus upper bound the indicator function appearing in (69) as follows:

$$\mathbb{1}_{\{u^t_{i,l}=u\}} \leq \mathbb{1}_{\{i_u=i\}} \mathbb{1}_{\{i_v \neq i \ \forall v \text{ s.t. } u+\tau+1 \leq v \leq t-\tau-1\}}. \tag{70}$$

This gives us:

$$\begin{aligned}
&\mathbb{E}\Big[\mathbb{1}_{\{u^t_{i,l}=u\}} \big(f'_i(\hat{x}_u)_l - f'_i(x^*)_l\big)^2\Big] \\
&\quad \leq \mathbb{E}\Big[\mathbb{1}_{\{i_u=i\}} \mathbb{1}_{\{i_v \neq i \ \forall v \text{ s.t. } u+\tau+1 \leq v \leq t-\tau-1\}} \big(f'_i(\hat{x}_u)_l - f'_i(x^*)_l\big)^2\Big] \\
&\quad \leq P\{i_u = i\} P\{i_v \neq i \ \forall v \text{ s.t. } u+\tau+1 \leq v \leq t-\tau-1\} \mathbb{E}\big(f'_i(\hat{x}_u)_l - f'_i(x^*)_l\big)^2 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (i_v \perp\!\!\!\perp \hat{x}_u, \forall v \geq u) \\
&\quad \leq \frac{1}{n}(1-\frac{1}{n})^{(t-2\tau-u-1)_+} \mathbb{E}\big(f'_i(\hat{x}_u)_l - f'_i(x^*)_l\big)^2.
\end{aligned} \tag{71}$$

Note that the third line used the crucial independence assumption $i_v \perp\!\!\!\perp \hat{x}_u, \forall v \geq u$ arising from our "After Read" ordering. Summing over all dimensions $l$, we then get:

$$\mathbb{E}\Big[\mathbb{1}_{\{u^t_{i,l}=u\}} \|f'_i(\hat{x}_u) - f'_i(x^*)\|^2\Big] \leq \frac{1}{n}(1-\frac{1}{n})^{(t-2\tau-u-1)_+} \mathbb{E}\|f'_i(\hat{x}_u) - f'_i(x^*)\|^2. \tag{72}$$

So now:

$$\begin{aligned}
\mathbb{E}\|\hat{\alpha}^t_{i_t} - f'_{i_t}(x^*)\|^2 - \lambda \tilde{e}_0 &\leq \frac{1}{n}\sum_{i=1}^n \sum_{u=1}^{t-1} \frac{1}{n}(1-\frac{1}{n})^{(t-2\tau-u-1)_+} \mathbb{E}\|f'_i(\hat{x}_u) - f'_i(x^*)\|^2 \\
&= \sum_{u=1}^{t-1} \frac{1}{n}(1-\frac{1}{n})^{(t-2\tau-u-1)_+} \frac{1}{n}\sum_{i=1}^n \mathbb{E}\|f'_i(\hat{x}_u) - f'_i(x^*)\|^2 \\
&= \sum_{u=1}^{t-1} \frac{1}{n}(1-\frac{1}{n})^{(t-2\tau-u-1)_+} \mathbb{E}\Big(\mathbf{E}\|f'_{i_u}(\hat{x}_u) - f'_{i_u}(x^*)\|^2\Big) \quad (i_u \perp\!\!\!\perp \hat{x}_u) \\
&\leq \frac{2L}{n} \sum_{u=1}^{t-1} (1-\frac{1}{n})^{(t-2\tau-u-1)_+} e_u \qquad \text{(by Equation 68)} \\
&= \frac{2L}{n} \sum_{u=1}^{(t-2\tau-1)_+} (1-\frac{1}{n})^{t-2\tau-u-1} e_u + \frac{2L}{n} \sum_{u=\max(1,t-2\tau)}^{t-1} e_u \,.
\end{aligned} \tag{73}$$

Note that we have excluded $\tilde{e}_0$ from our formula, using a generic $\lambda$ multiplier. We need to treat the case $u = 0$ differently to bound $\mathbb{1}_{\{u_{i,l}^t = u\}}$. Because all our initial $\alpha_i$ are initialized to a fixed $\alpha_i^0$, $\{u_i^t = 0\}$ just means that $i$ has not been picked between 0 and $t - \tau - 1$, i.e. $\{i_v \neq i \;\forall\; v \text{ s.t. } 0 \leq v \leq t - \tau - 1\}$. This means that the $\mathbb{1}_{\{i_u = i\}}$ term in (70) disappears and thus we lose a $\frac{1}{n}$ factor compared to the case where $u > 1$.

Let us now evaluate $\lambda$. We have:

$$\begin{aligned}\mathbb{E}\Big[\mathbb{1}_{\{u_i^t = 0\}}\|\alpha_i^0 - f_i'(x^*)\|^2\Big] &\leq \mathbb{E}\Big[\mathbb{1}_{\{i_v \neq i \;\forall\; v \text{ s.t. } 0 \leq v \leq t-\tau-1\}}\|\alpha_i^0 - f_i'(x^*)\|^2\Big] \\ &\leq P\{i_v \neq i \;\forall\; v \text{ s.t. } 0 \leq v \leq t - \tau - 1\}\mathbb{E}\|\alpha_i^0 - f_i'(x^*)\|^2 \\ &\leq (1 - \frac{1}{n})^{(t-\tau)_+}\mathbb{E}\|\alpha_i^0 - f_i'(x^*)\|^2. \end{aligned} \tag{74}$$

Plugging (73) and (74) into (67), we get Lemma 13:

$$\mathbb{E}\|g_t\|^2 \leq 4Le_t + \frac{4L}{n}\sum_{u=1}^{t-1}(1 - \frac{1}{n})^{(t-2\tau-u-1)_+}e_u + 4L(1 - \frac{1}{n})^{(t-\tau)_+}\tilde{e}_0\,, \tag{75}$$

where we have introduced $\tilde{e}_0 = \frac{1}{2L}\mathbb{E}\|\alpha_i^0 - f_i'(x^*)\|^2$. Note that in the original Saga algorithm, a batch gradient is computed to set the $\alpha_i^0 = f_i'(x_0)$. In this setting, we can write Lemma 13 using only $\tilde{e}_0 \leq e_0$ thanks to (68). In the more general setting where we initialize all $\alpha_i^0$ to a fixed quantity, we cannot use (68) to bound $\mathbb{E}\|\alpha_i^0 - f_i'(x^*)\|^2$ which means that we have to introduce $\tilde{e}_0$.

## B.4. Lemma 13 for Ahsvrg

In the simpler case of Ahsvrg as described in 4.2, we have a slight variation of (69):

$$\mathbb{E}\|f_i'(x_0^t) - f_i'(x^*)\|^2 = \sum_{u=0}^{t-1}\mathbb{E}\|\mathbb{1}_{\{u_i^t = u\}}\big(f_i'(\hat{x}_u) - f_i'(x^*)\big)\|^2. \tag{76}$$

Note that there is no sum over dimensions in this case because the full gradient computations and writes are synchronized (so the reference gradient is consistent).

As in Section B.3, we can upper bound the indicator $\mathbb{1}_{\{u_i^t = u\}}$. Now, $\{u_{i,l}^t = u\}$ requires two things: first, the next $B$ variable sampled after the $u^{\text{th}}$ update, $\tilde{B}_u$[27], was 1; second, $B$ was 0 for every update between $u$ and $t$ (roughly). Since the batch step is fully synchronized, we do not have to worry about updates from the past overwriting the reference gradient (and the iterates $x_u$ where we compute the gradient contains all past updates because we have waited for every core to finish its current update).

However, updating the state variable $s$ to 1 once a $B = 1$ variable is sampled is not atomic. So it is possible to have iterations with time label bigger than $u$ and that still use an older reference gradient for their update[28]. Fortunately, we can consider the state update as

27. We introduce this quantity because the iterations where full gradients are computed do not receive a time label since they do not correspond to updates to the iterates.

28. Conceivably, another core could start a new iteration, draw $B = 1$ and try to update $s$ to 1 themselves. This is not an issue since the operation of updating $s$ to 1 is idempotent. Only one reference gradient would be computed in this case.

any regular update to shared parameters. As such, Assumption 6 applies to it. This means that we can be certain that the reference gradient has been updated for iterations with time label $v \geq u + \tau + 1$.

This gives us:

$$\begin{aligned}\mathbb{E}\|\mathbb{1}_{\{u_i^t=u\}}\big(f_i'(\hat{x}_u) - f_i'(x^*)\big)\|^2 &\leq \mathbb{E}\Big[\mathbb{1}_{\{\tilde{B}_u=1\}}\mathbb{1}_{\{B_v=0 \ \forall v \text{ s.t. } u+1\leq v\leq t-\tau-1\}}\|f_i'(\hat{x}_u) - f_i'(x^*)\|^2\Big] \\ &\leq \frac{1}{n}\big(1-\frac{1}{n}\big)^{(t-\tau-u-1)_+}\mathbb{E}\|f_i'(\hat{x}_u) - f_i'(x^*)\|^2 \,. \end{aligned} \tag{77}$$

This proves Lemma 13 for Ahsvrg (while we actually have a slightly better exponent, $(t-\tau-u-1)_+$, we can upperbound it by the term in Lemma 13). Armed with this result, we can finish the proof of Theorem 8 for Ahsvrg in exactly the same manner as for Asaga. By remarking that the cost to get to iteration $t$ (including computing reference batch gradients) is the same in the sequential and parallel version, we see that our analysis for Corollary 9 for Asaga also applies for Ahsvrg, so both algorithms obey the same convergence and speedup results.

## B.5. Master Inequality Derivation

Now, if we combine the bound on $\mathbb{E}\|g_t\|^2$ which we just derived (i.e. Lemma 13) with Lemma 10, we get:

$$\begin{aligned} a_{t+1} \leq &(1-\frac{\gamma\mu}{2})a_t - 2\gamma e_t \\ &+ 4L\gamma^2 C_1 e_t + \frac{4L\gamma^2 C_1}{n}\sum_{u=1}^{t-1}(1-\frac{1}{n})^{(t-2\tau-u-1)_+}e_u + 4L\gamma^2 C_1(1-\frac{1}{n})^{(t-\tau)_+}\tilde{e}_0 \\ &+ 4L\gamma^2 C_2 \sum_{u=(t-\tau)_+}^{t-1} e_u + 4L\gamma^2 C_2 \sum_{u=(t-\tau)_+}^{t-1}(1-\frac{1}{n})^{(u-\tau)_+}\tilde{e}_0 \\ &+ \frac{4L\gamma^2 C_2}{n}\sum_{u=(t-\tau)_+}^{t-1}\sum_{v=1}^{u-1}(1-\frac{1}{n})^{(u-2\tau-v-1)_+}e_v \,. \end{aligned} \tag{78}$$

If we define $H_t := \sum_{u=1}^{t-1}(1-\frac{1}{n})^{(t-2\tau-u-1)_+}e_u$, then we get:

$$\begin{aligned} a_{t+1} \leq &(1-\frac{\gamma\mu}{2})a_t - 2\gamma e_t \\ &+ 4L\gamma^2 C_1\big(e_t + (1-\frac{1}{n})^{(t-\tau)_+}\tilde{e}_0\big) + \frac{4L\gamma^2 C_1}{n}H_t \\ &+ 4L\gamma^2 C_2 \sum_{u=(t-\tau)_+}^{t-1}\big(e_u + (1-\frac{1}{n})^{(u-\tau)_+}\tilde{e}_0\big) + \frac{4L\gamma^2 C_2}{n}\sum_{u=(t-\tau)_+}^{t-1} H_u \,, \end{aligned} \tag{79}$$

which is the master inequality (27).

### B.6. Lyapunov Function and Associated Recursive Inequality

We define $\mathcal{L}_t := \sum_{u=0}^{t}(1-\rho)^{t-u}a_u$ for some target contraction rate $\rho < 1$ to be defined later. We have:

$$\mathcal{L}_{t+1} = (1-\rho)^{t+1}a_0 + \sum_{u=1}^{t+1}(1-\rho)^{t+1-u}a_u = (1-\rho)^{t+1}a_0 + \sum_{u=0}^{t}(1-\rho)^{t-u}a_{u+1}\,. \tag{80}$$

We now use our new bound on $a_{t+1}$, (79):

$$\begin{aligned}\mathcal{L}_{t+1} \leq (1-\rho)^{t+1}a_0 + \sum_{u=0}^{t}(1-\rho)^{t-u}\Big[&(1-\frac{\gamma\mu}{2})a_u - 2\gamma e_u + 4L\gamma^2 C_1\big(e_u + (1-\frac{1}{n})^{(u-\tau)_+}\tilde{e}_0\big)\\ &+ \frac{4L\gamma^2C_1}{n}H_u + \frac{4L\gamma^2C_2}{n}\sum_{v=(u-\tau)_+}^{u-1}H_v\\ &+ 4L\gamma^2C_2\sum_{v=(u-\tau)_+}^{u-1}\big(e_v + (1-\frac{1}{n})^{(v-\tau)_+}\tilde{e}_0\big)\Big]\\ \leq (1-\rho)^{t+1}a_0 + (1-\frac{\gamma\mu}{2})\mathcal{L}_t&\\ + \sum_{u=0}^{t}(1-\rho)^{t-u}\Big[&-2\gamma e_u + 4L\gamma^2C_1\big(e_u + (1-\frac{1}{n})^{(u-\tau)_+}\tilde{e}_0\big)\\ &+ \frac{4L\gamma^2C_1}{n}H_u + \frac{4L\gamma^2C_2}{n}\sum_{v=(u-\tau)_+}^{u-1}H_v\\ &+ 4L\gamma^2C_2\sum_{v=(u-\tau)_+}^{u-1}\big(e_v + (1-\frac{1}{n})^{(v-\tau)_+}\tilde{e}_0\big)\Big].\end{aligned} \tag{81}$$

We can now rearrange the sums to expose a simple sum of $e_u$ multiplied by factors $r_u^t$:

$$\mathcal{L}_{t+1} \leq (1-\rho)^{t+1}a_0 + (1-\frac{\gamma\mu}{2})\mathcal{L}_t + \sum_{u=1}^{t}r_u^t e_u + r_0^t\tilde{e}_0\,. \tag{82}$$

### B.7. Proof of Lemma 14 (sufficient condition for convergence for Asaga)

We want to make explicit what conditions on $\rho$ and $\gamma$ are necessary to ensure that $r_u^t$ is negative for all $u \geq 1$. Since each $e_u$ is positive, we will then be able to safely drop the sum term from the inequality. The $r_0^t$ term is a bit trickier and is handled separately. Indeed, trying to enforce that $r_0^t$ is negative results in a significantly worse condition on $\gamma$ and eventually a convergence rate smaller by a factor of $n$ than our final result. Instead, we handle this term directly in the Lyapunov function.

***Computation of $r_u^t$.*** Let's now make the multiplying factor explicit. We assume $u \geq 1$. We split $r_u^t$ into five parts coming from (81):

- $r_1$, the part coming from the $-2\gamma e_u$ terms;

- $r_2$, coming from $4L\gamma^2 C_1 e_u$;
- $r_3$, coming from $\frac{4L\gamma^2 C_1}{n} H_u$;
- $r_4$, coming from $4L\gamma^2 C_2 \sum_{v=(u-\tau)_+}^{u-1} e_v$;
- $r_5$, coming from $\frac{4L\gamma^2 C_2}{n} \sum_{v=(u-\tau)_+}^{u-1} H_v$.

$r_1$ is easy to derive. Each of these terms appears only in one inequality. So for $u$ at time $t$, the term is:

$$r_1 = -2\gamma(1-\rho)^{t-u}. \tag{83}$$

For much the same reasons, $r_2$ is also easy to derive and is:

$$r_2 = 4L\gamma^2 C_1 (1-\rho)^{t-u}. \tag{84}$$

$r_3$ is a bit trickier, because for a given $v > 0$ there are several $H_u$ which contain $e_v$. The key insight is that we can rewrite our double sum in the following manner:

$$\begin{aligned}
&\sum_{u=0}^{t}(1-\rho)^{t-u}\sum_{v=1}^{u-1}(1-\frac{1}{n})^{(u-2\tau-v-1)_+}e_v \\
&\qquad = \sum_{v=1}^{t-1} e_v \sum_{u=v+1}^{t}(1-\rho)^{t-u}(1-\frac{1}{n})^{(u-2\tau-v-1)_+} \\
&\qquad \le \sum_{v=1}^{t-1} e_v \Big[\sum_{u=v+1}^{\min(t,v+2\tau)}(1-\rho)^{t-u} + \sum_{u=v+2\tau+1}^{t}(1-\rho)^{t-u}(1-\frac{1}{n})^{u-2\tau-v-1}\Big] \\
&\qquad \le \sum_{v=1}^{t-1} e_v \Big[2\tau(1-\rho)^{t-v-2\tau} + (1-\rho)^{t-v-2\tau-1}\sum_{u=v+2\tau+1}^{t} q^{u-2\tau-v-1}\Big] \\
&\qquad \le \sum_{v=1}^{t-1}(1-\rho)^{t-v}e_v(1-\rho)^{-2\tau-1}\big[2\tau+\frac{1}{1-q}\big],
\end{aligned} \tag{85}$$

where we have defined:

$$q := \frac{1-1/n}{1-\rho}, \quad \text{with the assumption } \rho < \frac{1}{n}\,. \tag{86}$$

Note that we have bounded the $\min(t, v+2\tau)$ term by $v+2\tau$ in the first sub-sum, effectively adding more positive terms.

This gives us that at time $t$, for $u$:

$$r_3 \le \frac{4L\gamma^2 C_1}{n}(1-\rho)^{t-u}(1-\rho)^{-2\tau-1}\big[2\tau+\frac{1}{1-q}\big]. \tag{87}$$

For $r_4$ we use the same trick:

$$\begin{aligned}
\sum_{u=0}^{t}(1-\rho)^{t-u}\sum_{v=(u-\tau)_+}^{u-1} e_v &= \sum_{v=0}^{t-1} e_v \sum_{u=v+1}^{\min(t,v+\tau)}(1-\rho)^{t-u} \\
&\le \sum_{v=0}^{t-1} e_v \sum_{u=v+1}^{v+\tau}(1-\rho)^{t-u} \le \sum_{v=0}^{t-1} e_v \tau (1-\rho)^{t-v-\tau}.
\end{aligned} \tag{88}$$

This gives us that at time $t$, for $u$:

$$r_4 \leq 4L\gamma^2 C_2 (1-\rho)^{t-u} \tau (1-\rho)^{-\tau} \, . \tag{89}$$

Finally we compute $r_5$ which is the most complicated term. Indeed, to find the factor of $e_w$ for a given $w > 0$, one has to compute a triple sum, $\sum_{u=0}^{t} (1-\rho)^{t-u} \sum_{v=(u-\tau)_+}^{u-1} H_v$. We start by computing the factor of $e_w$ in the inner double sum, $\sum_{v=(u-\tau)_+}^{u-1} H_v$.

$$\sum_{v=(u-\tau)_+}^{u-1} \sum_{w=1}^{v-1} (1-\frac{1}{n})^{(v-2\tau-w-1)_+} e_w = \sum_{w=1}^{u-2} e_w \sum_{v=\max(w+1, u-\tau)}^{u-1} (1-\frac{1}{n})^{(v-2\tau-w-1)_+} \, . \tag{90}$$

Now there are at most $\tau$ terms for each $e_w$. If $w \leq u - 3\tau - 1$, then the exponent is positive in every term and it is always bigger than $u - 3\tau - 1 - w$, which means we can bound the sum by $\tau(1-\frac{1}{n})^{u-3\tau-1-w}$. Otherwise we can simply bound the sum by $\tau$. We get:

$$\sum_{v=(u-\tau)_+}^{u-1} H_v \leq \sum_{w=1}^{u-2} \big[ \mathbb{1}_{\{u-3\tau \leq w \leq u-2\}} \tau + \mathbb{1}_{\{w \leq u-3\tau-1\}} \tau (1-\frac{1}{n})^{u-3\tau-1-w} \big] e_w \, . \tag{91}$$

This means that for $w$ at time $t$:

$$\begin{aligned}
r_5 &\leq \frac{4L\gamma^2 C_2}{n} \sum_{u=0}^{t} (1-\rho)^{t-u} \big[ \mathbb{1}_{\{u-3\tau \leq w \leq u-2\}} \tau + \mathbb{1}_{\{w \leq u-3\tau-1\}} \tau (1-\frac{1}{n})^{u-3\tau-1-w} \big] \\
&\leq \frac{4L\gamma^2 C_2}{n} \Big[ \sum_{u=w+2}^{\min(t, w+3\tau)} \tau (1-\rho)^{t-u} + \sum_{u=w+3\tau+1}^{t} \tau (1-\frac{1}{n})^{u-3\tau-1-w} (1-\rho)^{t-u} \Big] \\
&\leq \frac{4L\gamma^2 C_2}{n} \tau \Big[ (1-\rho)^{t-w} (1-\rho)^{-3\tau} 3\tau \\
&\qquad + (1-\rho)^{t-w} (1-\rho)^{-1-3\tau} \sum_{u=w+3\tau+1}^{t} (1-\frac{1}{n})^{u-3\tau-1-w} (1-\rho)^{-u+3\tau+1+w} \Big] \\
&\leq \frac{4L\gamma^2 C_2}{n} \tau (1-\rho)^{t-w} (1-\rho)^{-3\tau-1} \big( 3\tau + \frac{1}{(1-q)} \big) \, .
\end{aligned} \tag{92}$$

By combining the five terms together (83, 84, 87, 89 and 92), we get that $\forall u$ s.t. $1 \leq u \leq t$:

$$\begin{aligned}
r_u^t \leq (1-\rho)^{t-u} \Big[ &-2\gamma + 4L\gamma^2 C_1 + \frac{4L\gamma^2 C_1}{n} (1-\rho)^{-2\tau-1} \big( 2\tau + \frac{1}{1-q} \big) \\
&+ 4L\gamma^2 C_2 \tau (1-\rho)^{-\tau} + \frac{4L\gamma^2 C_2}{n} \tau (1-\rho)^{-3\tau-1} \big( 3\tau + \frac{1}{1-q} \big) \Big] .
\end{aligned} \tag{93}$$

***Computation of $r_0^t$.*** Recall that we treat the $\tilde{e}_0$ term separately in Section B.3. The initialization of Saga creates an initial synchronization, which means that the contribution of $\tilde{e}_0$ in our bound on $\mathbb{E}\|g_t\|^2$ (75) is roughly $n$ times bigger than the contribution of any $e_u$ for $1 < u < t$.[29] In order to safely handle this term in our Lyapunov inequality, we only need to prove that it is bounded by a reasonable constant. Here again, we split $r_0^t$ in five contributions coming from (81):

29. This is explained in details right before (74).

- $r_1$, the part coming from the $-2\gamma e_u$ terms;
- $r_2$, coming from $4L\gamma^2 C_1 e_u$;
- $r_3$, coming from $4L\gamma^2 C_1 (1-\frac{1}{n})^{(u-\tau)_+}\tilde{e}_0$;
- $r_4$, coming from $4L\gamma^2 C_2 \sum_{v=(u-\tau)_+}^{u-1} e_v$;
- $r_5$, coming from $4L\gamma^2 C_2 \sum_{v=(u-\tau)_+}^{u-1} (1-\frac{1}{n})^{(v-\tau)_+}\tilde{e}_0$.

Note that there is no $\tilde{e}_0$ in $H_t$, which is why we can safely ignore these terms here.

We have $r_1 = -2\gamma(1-\rho)^t$ and $r_2 = 4L\gamma^2 C_1 (1-\rho)^t$.

Let us compute $r_3$.

$$\begin{aligned}
&\sum_{u=0}^{t} (1-\rho)^{t-u}(1-\frac{1}{n})^{(u-\tau)_+} \\
&\qquad = \sum_{u=0}^{\min(t,\tau)} (1-\rho)^{t-u} + \sum_{u=\tau+1}^{t} (1-\rho)^{t-u}(1-\frac{1}{n})^{u-\tau} \\
&\qquad \leq (\tau+1)(1-\rho)^{t-\tau} + (1-\rho)^{t-\tau} \sum_{u=\tau+1}^{t} (1-\rho)^{\tau-u}(1-\frac{1}{n})^{u-\tau} \\
&\qquad \leq (1-\rho)^t (1-\rho)^{-\tau}\big(\tau + 1 + \frac{1}{1-q}\big) . \qquad (94)
\end{aligned}$$

This gives us:

$$r_3 \leq (1-\rho)^t 4L\gamma^2 C_1 (1-\rho)^{-\tau}\big(\tau + 1 + \frac{1}{1-q}\big) . \qquad (95)$$

We have already computed $r_4$ for $u > 0$ and the computation is exactly the same for $u = 0$. $r_4 \leq (1-\rho)^t 4L\gamma^2 C_2 \frac{\tau}{1-\rho}$ .

Finally we compute $r_5$.

$$\begin{aligned}
&\sum_{u=0}^{t} (1-\rho)^{t-u} \sum_{v=(u-\tau)_+}^{u-1} (1-\frac{1}{n})^{(v-\tau)_+} \\
&\qquad = \sum_{v=1}^{t-1} \sum_{u=v+1}^{\min(t,v+\tau)} (1-\rho)^{t-u}(1-\frac{1}{n})^{(v-\tau)_+} \\
&\qquad \leq \sum_{v=1}^{\min(t-1,\tau)} \sum_{u=v+1}^{v+\tau} (1-\rho)^{t-u} + \sum_{v=\tau+1}^{t-1} \sum_{u=v+1}^{\min(t,v+\tau)} (1-\rho)^{t-u}(1-\frac{1}{n})^{v-\tau} \\
&\qquad \leq \tau^2 (1-\rho)^{t-2\tau} + \sum_{v=\tau+1}^{t-1} (1-\frac{1}{n})^{v-\tau} \tau (1-\rho)^{t-v-\tau} \\
&\qquad \leq \tau^2 (1-\rho)^{t-2\tau} + \tau (1-\rho)^t (1-\rho)^{-2\tau} \sum_{v=\tau+1}^{t-1} (1-\frac{1}{n})^{v-\tau} \tau (1-\rho)^{-v+\tau} \\
&\qquad \leq (1-\rho)^t (1-\rho)^{-2\tau}\big(\tau^2 + \tau \frac{1}{1-q}\big) . \qquad (96)
\end{aligned}$$

Which means:

$$r_5 \leq (1-\rho)^t 4L\gamma^2 C_2 (1-\rho)^{-2\tau}\big(\tau^2 + \tau\frac{1}{1-q}\big). \tag{97}$$

Putting it all together, we get that: $\forall t \geq 0$,

$$\begin{aligned} r_0^t \leq (1-\rho)^t \Big[\Big(-2\gamma + 4L\gamma^2 C_1 + 4L\gamma^2 C_2 \frac{\tau}{1-\rho}\Big)\frac{e_0}{\tilde{e}_0} \\ + 4L\gamma^2 C_1 (1-\rho)^{-\tau}\big(\tau + 1 + \frac{1}{1-q}\big) + 4L\gamma^2 C_2 \tau (1-\rho)^{-2\tau}\big(\tau + \frac{1}{1-q}\big)\Big]. \end{aligned} \tag{98}$$

***Sufficient condition for convergence.*** We need all $r_u^t, u \geq 1$ to be negative so we can safely drop them from (82). Note that for every $u$, this is the same condition. We will reduce that condition to a second-order polynomial sign condition. We also remark that since $\gamma \geq 0$, we can upper bound our terms in $\gamma$ and $\gamma^2$ in this upcoming polynomial, which will give us sufficient conditions for convergence.

Now, recall that $C_2(\gamma)$ (as defined in (17)) depends on $\gamma$. We thus need to expand it once more to find our conditions. We have:

$$C_1 = 1 + \sqrt{\Delta}\tau; \qquad C_2 = \sqrt{\Delta} + \gamma\mu C_1 \,.$$

Dividing the bracket in (93) by $\gamma$ and rearranging as a second degree polynomial, we get the condition:

$$\begin{aligned} 4L\Bigg(C_1 + \frac{C_1}{n}(1-\rho)^{-2\tau-1}\Big[2\tau + \frac{1}{1-q}\Big] + \Big[\frac{\sqrt{\Delta}\tau}{(1-\rho)^\tau} + \frac{\sqrt{\Delta}\tau}{n}(1-\rho)^{-3\tau-1}(3\tau + \frac{1}{1-q})\Big]\Bigg)\gamma \\ + 8\mu C_1 L\tau\Big[(1-\rho)^{-\tau} + \frac{1}{n}(1-\rho)^{-3\tau-1}(3\tau + \frac{1}{1-q})\Big]\gamma^2 + 2 \leq 0 \,. \end{aligned} \tag{99}$$

The discriminant of this polynomial is always positive, so $\gamma$ needs to be between its two roots. The smallest is negative, so the condition is not relevant to our case (where $\gamma > 0$). By solving analytically for the positive root $\phi$, we get an upper bound condition on $\gamma$ that can be used for any overlap $\tau$ and guarantee convergence. Unfortunately, for large $\tau$, the upper bound becomes exponentially small because of the presence of $\tau$ in the exponent in (99). More specifically, by using the bound $1/(1-\rho) \leq \exp(2\rho)$[30] and thus $(1-\rho)^{-\tau} \leq \exp(2\tau\rho)$ in (99), we would obtain factors of the form $\exp(\tau/n)$ in the denominator for the root $\phi$ (recall that $\rho < 1/n$).

Our Lemma 14 is derived instead under the assumption that $\tau \leq \mathcal{O}(n)$, with the constants chosen in order to make the condition (99) more interpretable and to relate our convergence result with the standard SAGA convergence (see Theorem 2). As explained in Section 6.7, the assumption that $\tau \leq \mathcal{O}(n)$ appears reasonable in practice. First, by using Bernoulli's inequality, we have:

$$(1-\rho)^{k\tau} \geq 1 - k\tau\rho \qquad \text{for integers} \quad k\tau \geq 0 \,. \tag{100}$$

30. This bound can be derived from the inequality $(1 - x/2) \geq \exp(-x)$ which is valid for $0 \leq x \leq 1.59$.

To get manageable constants, we make the following slightly more restrictive assumptions on the target rate $\rho$[31] and overlap $\tau$:[32]

$$\rho \leq \frac{1}{4n} \tag{101}$$
$$\tau \leq \frac{n}{10}\,. \tag{102}$$

We then have:

$$\frac{1}{1-q} \leq \frac{4n}{3} \tag{103}$$
$$\frac{1}{1-\rho} \leq \frac{4}{3} \tag{104}$$
$$k\tau\rho \leq \frac{3}{40} \qquad \text{for } 1 \leq k \leq 3 \tag{105}$$
$$(1-\rho)^{-k\tau} \leq \frac{1}{1-k\tau\rho} \leq \frac{40}{37} \qquad \text{for } 1 \leq k \leq 3 \text{ and by using (100).} \tag{106}$$

We can now upper bound loosely the three terms in brackets appearing in (99) as follows:

$$(1-\rho)^{-2\tau-1}\big[2\tau + \frac{1}{1-q}\big] \leq 3n \tag{107}$$
$$\sqrt{\Delta}\tau(1-\rho)^{-\tau} + \frac{\sqrt{\Delta}\tau}{n}(1-\rho)^{-3\tau-1}(3\tau + \frac{1}{1-q}) \leq 4\sqrt{\Delta}\tau \leq 4C_1 \tag{108}$$
$$(1-\rho)^{-\tau} + \frac{1}{n}(1-\rho)^{-3\tau-1}(3\tau + \frac{1}{1-q}) \leq 4\,. \tag{109}$$

By plugging (107)–(109) into (99), we get the simpler sufficient condition on $\gamma$:

$$-1 + 16LC_1\gamma + 16LC_1\mu\tau\gamma^2 \leq 0\,. \tag{110}$$

The positive root $\phi$ is:

$$\phi = \frac{16LC_1(\sqrt{1+\frac{\mu\tau}{4LC_1}} - 1)}{32LC_1\mu\tau} = \frac{\sqrt{1+\frac{\mu\tau}{4LC_1}} - 1}{2\mu\tau}\,. \tag{111}$$

We simplify it further by using the inequality:[33]

$$\sqrt{x} - 1 \geq \frac{x-1}{2\sqrt{x}} \qquad \forall x > 0\,. \tag{112}$$

Using (112) in (111), and recalling that $\kappa := L/\mu$, we get:

$$\phi \geq \frac{1}{16LC_1\sqrt{1+\frac{\tau}{4\kappa C_1}}}\,. \tag{113}$$

31. Note that we already expected $\rho < 1/n$.
32. This bound on $\tau$ is reasonable in practice, see Appendix 6.7.
33. This inequality can be derived by using the concavity property $f(y) \leq f(x) + (y-x)f'(x)$ on the differentiable concave function $f(x) = \sqrt{x}$ with $y = 1$.

Since $\frac{\tau}{C_1} = \frac{\tau}{1+\sqrt{\Delta}\tau} \leq \min(\tau, \frac{1}{\sqrt{\Delta}})$, we get that a sufficient condition on our step size is:

$$\gamma \leq \frac{1}{16L(1+\sqrt{\Delta}\tau)\sqrt{1+\frac{1}{4\kappa}\min(\tau, \frac{1}{\sqrt{\Delta}})}}\,. \tag{114}$$

Subject to our conditions on $\gamma$, $\rho$ and $\tau$, we then have that: $r_u^t \leq 0$ for all $u$ s.t. $1 \leq u \leq t$. This means we can rewrite (82) as:

$$\mathcal{L}_{t+1} \leq (1-\rho)^{t+1}a_0 + (1-\frac{\gamma\mu}{2})\mathcal{L}_t + r_0^t\tilde{e}_0\,. \tag{115}$$

Now, we could finish the proof from this inequality, but it would only give us a convergence result in terms of $a_t = \mathbb{E}\|x_t - x^*\|^2$. A better result would be in terms of the suboptimality at $\hat{x}_t$ (because $\hat{x}_t$ is a real quantity in the algorithm whereas $x_t$ is virtual). Fortunately, to get such a result, we can easily adapt (115).

We make $e_t$ appear on the left side of (115), by adding $\gamma$ to $r_t^t$ in (82):[34]

$$\gamma e_t + \mathcal{L}_{t+1} \leq (1-\rho)^{t+1}a_0 + (1-\frac{\gamma\mu}{2})\mathcal{L}_t + \sum_{u=1}^{t-1} r_u^t e_u + r_0^t\tilde{e}_0 + (r_t^t + \gamma)e_t. \tag{116}$$

We now require the stronger property that $\gamma + r_t^t \leq 0$, which translates to replacing $-2\gamma$ with $-\gamma$ in (93):

$$\begin{aligned}0 \geq \Big[ &-\gamma + 4L\gamma^2C_1 + \frac{4L\gamma^2C_1}{n}(1-\rho)^{-2\tau-1}\big(2\tau + \frac{1}{1-q}\big) \\ &+ 4L\gamma^2C_2\tau(1-\rho)^{-\tau} + \frac{4L\gamma^2C_2}{n}\tau(1-\rho)^{-3\tau}\big(3\tau + \frac{1}{1-q}\big)\Big].\end{aligned} \tag{117}$$

We can easily derive a new stronger condition on $\gamma$ under which we can drop all the $e_u, u > 0$ terms in (116):

$$\gamma \leq \gamma^* = \frac{1}{32L(1+\sqrt{\Delta}\tau)\sqrt{1+\frac{1}{8\kappa}\min(\tau, \frac{1}{\sqrt{\Delta}})}}, \tag{118}$$

and thus under which we get:

$$\gamma e_t + \mathcal{L}_{t+1} \leq (1-\rho)^{t+1}a_0 + (1-\frac{\gamma\mu}{2})\mathcal{L}_t + r_0^t\tilde{e}_0. \tag{119}$$

This finishes the proof of Lemma 14. ■

34. We could use any multiplier from 0 to $2\gamma$, but choose $\gamma$ for simplicity. For this reason and because our analysis of the $r_t^t$ term was loose, we could derive a tighter bound, but it does not change the leading terms.

## B.8. Proof of Theorem 8 (convergence guarantee and rate of Asaga)

***End of Lyapunov convergence.*** We continue with the assumptions of Lemma 14 which gave us (119). Thanks to (98), we can also rewrite $r_0^t \leq (1-\rho)^{t+1}A$ where $A$ is a constant which depends on $n$, $\Delta$, $\gamma$ and $L$ but is finite and crucially does not depend on $t$. In fact, by reusing similar arguments as in B.7, we can show the loose bound $A \leq \gamma n$ under the assumptions of Lemma 14 (including $\gamma \leq \gamma^*$).[35] We then have:

$$\begin{aligned}\mathcal{L}_{t+1} \leq \gamma e_t + \mathcal{L}_{t+1} &\leq (1-\frac{\gamma\mu}{2})\mathcal{L}_t + (1-\rho)^{t+1}(a_0 + A\tilde{e}_0) \\ &\leq (1-\frac{\gamma\mu}{2})^{t+1}\mathcal{L}_0 + (a_0 + A\tilde{e}_0)\sum_{k=0}^{t+1}(1-\rho)^{t+1-k}(1-\frac{\gamma\mu}{2})^k. \qquad (120)\end{aligned}$$

We have two linearly contracting terms. The sum contracts linearly with the minimum geometric rate factor between $\gamma\mu/2$ and $\rho$. If we define $m := \min(\rho, \gamma\mu/2)$, $M := \max(\rho, \gamma\mu/2)$ and $\rho^* := \nu m$ with $0 < \nu < 1$,[36] we then get:[37]

$$\begin{aligned}\gamma e_t \leq \gamma e_t + \mathcal{L}_{t+1} &\leq (1-\frac{\gamma\mu}{2})^{t+1}\mathcal{L}_0 + (a_0 + A\tilde{e}_0)\sum_{k=0}^{t+1}(1-m)^{t+1-k}(1-M)^k \\ &\leq (1-\frac{\gamma\mu}{2})^{t+1}\mathcal{L}_0 + (a_0 + A\tilde{e}_0)\sum_{k=0}^{t+1}(1-\rho^*)^{t+1-k}(1-M)^k \\ &\leq (1-\frac{\gamma\mu}{2})^{t+1}\mathcal{L}_0 + (a_0 + A\tilde{e}_0)(1-\rho^*)^{t+1}\sum_{k=0}^{t+1}(1-\rho^*)^{-k}(1-M)^k \\ &\leq (1-\frac{\gamma\mu}{2})^{t+1}\mathcal{L}_0 + (1-\rho^*)^{t+1}\frac{1}{1-\eta}(a_0 + A\tilde{e}_0) \\ &\leq (1-\rho^*)^{t+1}\big(a_0 + \frac{1}{1-\eta}(a_0 + A\tilde{e}_0)\big), \qquad (121)\end{aligned}$$

where $\eta := \frac{1-M}{1-\rho^*}$. We have $\frac{1}{1-\eta} = \frac{1-\rho^*}{M-\rho^*}$.

By taking $\nu = \frac{4}{5}$ and setting $\rho = \frac{1}{4n}$ – its maximal value allowed by the assumptions of Lemma 14 – we get $M \geq \frac{1}{4n}$ and $\rho^* \leq \frac{1}{5n}$, which means $\frac{1}{1-\eta} \leq 20n$. All told, using $A \leq \gamma n$, we get:

$$e_t \leq (1-\rho^*)^{t+1}\tilde{C}_0, \qquad (122)$$

where:

$$\tilde{C}_0 := \frac{21n}{\gamma}\Big(\|x_0 - x^*\|^2 + \gamma\frac{n}{2L}\mathbb{E}\|\alpha_i^0 - f_i'(x^*)\|^2\Big). \qquad (123)$$

Since we set $\rho = \frac{1}{4n}, \nu = \frac{4}{5}$, we have $\nu\rho = \frac{1}{5n}$. Using a step size $\gamma = \frac{a}{L}$ as in Theorem 8, we get $\nu\frac{\gamma\mu}{2} = \frac{2a}{5\kappa}$. We thus obtain a geometric rate of $\rho^* = \min\{\frac{1}{5n}, a\frac{2}{5\kappa}\}$, which we simplified

35. In particular, note that $e_0$ does not appear in the definition of $A$ because it turns out that the parenthesis group multiplying $e_0$ in (98) is negative. Indeed, it contains less positive terms than (93) which we showed to be negative under the assumptions from Lemma 14.

36. $\nu$ is introduced to circumvent the problematic case where $\rho$ and $\gamma\mu/2$ are too close together, which does not prevent the geometric convergence, but makes the constant $\frac{1}{1-\eta}$ potentially very big (in the case both terms are equal, the sum even becomes an annoying linear term in t).

37. Note that if $m \neq \rho$, we can perform the index change $t+1-k \rightarrow k$ to get the sum.

to $\frac{1}{5}\min\{\frac{1}{n}, a\frac{1}{\kappa}\}$ in Theorem 8, finishing the proof. We also observe that $\tilde{C}_0 \leq \frac{60n}{\gamma}C_0$, with $C_0$ defined in Theorem 2. ■

### B.9. Proof of Corollary 9 (speedup regimes for Asaga)

Referring to Hofmann et al. (2015) and our own Theorem 2, the geometric rate factor of Saga is $\frac{1}{5}\min\{\frac{1}{n}, \frac{a}{\kappa}\}$ for a step size of $\gamma = \frac{a}{5L}$. We start by proving the first part of the corollary which considers the step size $\gamma = \frac{a}{L}$ with $a = a^*(\tau)$. We distinguish between two regimes to study the parallel speedup our algorithm obtains and to derive a condition on $\tau$ for which we have a linear speedup.

***Well-conditioned regime.*** In this regime, $n > \kappa$ and the geometric rate factor of sequential Saga is $\frac{1}{5n}$. To get a linear speedup (up to a constant factor), we need to enforce $\rho^* = \Omega(\frac{1}{n})$. We recall that $\rho^* = \min\{\frac{1}{5n}, a\frac{1}{5\kappa}\}$.

We already have $\frac{1}{5n} = \Omega(\frac{1}{n})$. This means that we need $\tau$ to verify $\frac{a^*(\tau)}{5\kappa} = \Omega(\frac{1}{n})$, where $a^*(\tau) = \frac{1}{32\left(1+\tau\sqrt{\Delta}\right)\xi(\kappa,\Delta,\tau)}$ according to Theorem 8. Recall that $\xi(\kappa, \Delta, \tau) := \sqrt{1 + \frac{1}{8\kappa}\min\{\frac{1}{\sqrt{\Delta}}, \tau\}}$. Up to a constant factor, this means we can give the following sufficient condition:

$$\frac{1}{\kappa\left(1+\tau\sqrt{\Delta}\right)\xi(\kappa,\Delta,\tau)} = \Omega\left(\frac{1}{n}\right) \tag{124}$$

i.e.

$$\left(1+\tau\sqrt{\Delta}\right)\xi(\kappa,\Delta,\tau) = \mathcal{O}\left(\frac{n}{\kappa}\right). \tag{125}$$

We now consider two alternatives, depending on whether $\kappa$ is bigger than $\frac{1}{\sqrt{\Delta}}$ or not. If $\kappa \geq \frac{1}{\sqrt{\Delta}}$, then $\xi(\kappa, \Delta, \tau) < 2$ and we can rewrite the sufficient condition (125) as:

$$\tau = \mathcal{O}(1)\frac{n}{\kappa\sqrt{\Delta}}. \tag{126}$$

In the alternative case, $\kappa \leq \frac{1}{\sqrt{\Delta}}$. Since $a^*(\tau)$ is decreasing in $\tau$, we can suppose $\tau \geq \frac{1}{\sqrt{\Delta}}$ without loss of generality and thus $\xi(\kappa, \Delta, \tau) = \sqrt{1 + \frac{1}{8\kappa\sqrt{\Delta}}}$. We can then rewrite the sufficient condition (125) as:

$$\frac{\tau\sqrt{\Delta}}{\sqrt{\kappa}\sqrt[4]{\Delta}} = \mathcal{O}(\frac{n}{\kappa})\,;$$
$$\tau = \mathcal{O}(1)\frac{n}{\sqrt{\kappa}\sqrt[4]{\Delta}}\,. \tag{127}$$

We observe that since we have supposed that $\kappa \leq \frac{1}{\sqrt{\Delta}}$, we have $\sqrt{\kappa\sqrt{\Delta}} \leq \kappa\sqrt{\Delta} \leq 1$, which means that our initial assumption that $\tau < \frac{n}{10}$ is stronger than condition (127).

We can now combine both cases to get the following sufficient condition for the geometric rate factor of Asaga to be the same order as sequential Saga when $n > \kappa$:

$$\tau = \mathcal{O}(1)\frac{n}{\kappa\sqrt{\Delta}};\quad \tau = \mathcal{O}(n)\,. \tag{128}$$

***Ill-conditioned regime.*** In this regime, $\kappa > n$ and the geometric rate factor of sequential Saga is $a\frac{1}{\kappa}$. Here, to obtain a linear speedup, we need $\rho^* = \mathcal{O}(\frac{1}{\kappa})$. Since $\frac{1}{n} > \frac{1}{\kappa}$, all we require is that $\frac{a^*(\tau)}{\kappa} = \Omega(\frac{1}{\kappa})$ where $a^*(\tau) = \frac{1}{32\left(1+\tau\sqrt{\Delta}\right)\xi(\kappa,\Delta,\tau)}$, which reduces to $a^*(\tau) = \Omega(1)$.

We can give the following sufficient condition:

$$\frac{1}{\left(1+\tau\sqrt{\Delta}\right)\xi(\kappa,\Delta,\tau)} = \Omega(1) \tag{129}$$

Using that $\frac{1}{n} \leq \Delta \leq 1$ and that $\kappa > n$, we get that $\xi(\kappa,\Delta,\tau) \leq 2$, which means our sufficient condition becomes:

$$\begin{aligned} \tau\sqrt{\Delta} &= \mathcal{O}(1) \\ \tau &= \frac{\mathcal{O}(1)}{\sqrt{\Delta}}. \end{aligned} \tag{130}$$

This finishes the proof for the first part of Corollary 9.

***Universal step size.*** If $\tau = \mathcal{O}(\frac{1}{\sqrt{\Delta}})$, then $\xi(\kappa,\Delta,\tau) = \mathcal{O}(1)$ and $(1+\tau\sqrt{\Delta}) = \mathcal{O}(1)$, and thus $a^*(\tau) = \Omega(1)$ (for any $n$ and $\kappa$). This means that the universal step size $\gamma = \Theta(1/L)$ satisfies $\gamma \leq a^*(\tau)$ for any $\kappa$, giving the same rate factor $\Omega(\min\{\frac{1}{n}, \frac{1}{\kappa}\})$ that sequential Saga has, completing the proof for the second part of Corollary 9. ■

## Appendix C. Kromagnon – Proof of Theorem 15 and Corollary 18

### C.1. Proof of Lemma 19 (suboptimality bound on $\mathbb{E}\|g_t\|^2$)

Mania et al. (2017, Lemma 9), tells us that for serial sparse Svrg we have for all $km \leq t \leq (k+1)m - 1$:

$$\mathbb{E}\|g_t\|^2 \leq 2\mathbb{E}\|f'_{i_t}(\hat{x}_t) - f'_{i_t}(x^*)\|^2 + 2\mathbb{E}\|f'_{i_t}(\hat{x}_k) - f'_{i_t}(x^*)\|^2. \tag{131}$$

This remains true in the case of Kromagnon. We can use Hofmann et al. (2015, Equations 7 and 8) to bound both terms in the following manner:

$$\mathbb{E}\|g_t\|^2 \leq 4L(\mathbb{E}f(\hat{x}_t) - f(x^*)) + 4L(\mathbb{E}f(\tilde{x}_k) - f(x^*)) \leq 4Le_t + 4L\tilde{e}_k\,. \tag{132}$$

■

### C.2. Proof of Theorem 15 (convergence guarantee and rate of Kromagnon)

***Master inequality derivation.*** As in our Asaga analysis, we plug Lemma 19 in Lemma 10, which gives us that for all $k \geq 0, km \leq t \leq (k+1)m - 1$:

$$a_{t+1} \leq (1 - \frac{\gamma\mu}{2})a_t + \gamma^2 C_1(4Le_t + 4L\tilde{e}_k) + \gamma^2 C_2 \sum_{u=\max(km,t-\tau)}^{t-1} (4Le_t + 4L\tilde{e}_k) - 2\gamma e_t\,. \tag{133}$$

By grouping the $\tilde{e}_k$ and the $e_t$ terms we get our master inequality (42):

$$a_{t+1} \leq (1 - \frac{\gamma\mu}{2})a_t + (4L\gamma^2 C_1 - 2\gamma)e_t + 4L\gamma^2 C_2 \sum_{u=\max(km, t-\tau)}^{t-1} e_u + (4L\gamma^2 C_1 + 4L\gamma^2 \tau C_2)\tilde{e}_k \,.$$

***Contraction inequality derivation.*** We now adopt the same method as in the original Svrg paper (Johnson and Zhang, 2013); we sum the master inequality over a whole epoch and then we use the randomization trick:

$$\tilde{e}_k = \mathbb{E} f(\tilde{x}_k) - f(x^*) = \frac{1}{m} \sum_{t=(k-1)m}^{km-1} e_t \tag{134}$$

This gives us:

$$\begin{aligned} \sum_{t=km+1}^{(k+1)m} a_t \leq &(1 - \frac{\gamma\mu}{2}) \sum_{t=km}^{(k+1)m-1} a_t + (4L\gamma^2 C_1 - 2\gamma) \sum_{t=km}^{(k+1)m-1} e_t \\ &+ 4L\gamma^2 C_2 \sum_{t=km}^{(k+1)m-1} \sum_{u=\max(km, t-\tau)}^{t-1} e_u + m(4L\gamma^2 C_1 + 4L\gamma^2\tau C_2)\tilde{e}_k \,. \end{aligned} \tag{135}$$

Since $1 - \frac{\gamma\mu}{2} < 1$, we can upper bound it by 1, and then remove all the telescoping terms from (135). We also have:

$$\begin{aligned} \sum_{t=km}^{(k+1)m-1} \sum_{u=\max(km, t-\tau)}^{t-1} e_u = \sum_{u=km}^{(k+1)m-2} \sum_{t=u+1}^{\min((k+1)m-1, u+\tau)} e_u &\leq \tau \sum_{u=km}^{(k+1)m-2} e_u \\ &\leq \tau \sum_{u=km}^{(k+1)m-1} e_u \,. \end{aligned} \tag{136}$$

All told:

$$a_{(k+1)m} \leq a_{km} + (4L\gamma^2 C_1 + 4L\gamma^2\tau C_2 - 2\gamma) \sum_{t=km}^{(k+1)m-1} e_t + m(4L\gamma^2 C_1 + 4L\gamma^2\tau C_2)\tilde{e}_k \,. \tag{137}$$

Now we use the randomization trick (134):

$$a_{(k+1)m} \leq a_{km} + (4L\gamma^2 C_1 + 4L\gamma^2\tau C_2 - 2\gamma)m\tilde{e}_{k+1} + m(4L\gamma^2 C_1 + 4L\gamma^2\tau C_2)\tilde{e}_k \,. \tag{138}$$

Finally, in order to get a recursive inequality in $\tilde{e}_k$, we can remove the positive $a_{(k+1)m}$ term from the left-hand side of (138) and bound the $a_{km}$ term on the right-hand side by $2e_{km}/\mu$ using a standard strong convexity inequality. We get our final contraction inequality (45):

$$(2\gamma m - 4mL\gamma^2 C_1 - 4mL\gamma^2\tau C_2)\tilde{e}_{k+1} \leq \big(\frac{2}{\mu} + 4mL\gamma^2 C_1 + 4mL\gamma^2\tau C_2\big)\tilde{e}_k \,.$$

■

## C.3. Proof of Corollary 16, Corollary 17 and Corollary 18 (speedup regimes)

***A simpler result for* Svrg.** The standard convergence rate for serial Svrg is given by:

$$\theta := \frac{\frac{1}{\mu\gamma m} + 2L\gamma}{1 - 2L\gamma}\,. \tag{139}$$

If we define $a$ such that $\gamma = a/4L$, we obtain:

$$\theta = \frac{\frac{4\kappa}{am} + \frac{a}{2}}{1 - \frac{a}{2}}\,. \tag{140}$$

Now, since we need $\theta \leq 1$, we see that we require $a \leq 1$. The optimal value of the denominator is then 1 (when $a = 0$), whereas the worst case value is $1/2$ ($a = 1$). We can thus upper bound $\theta$ by replacing the denominator with $1/2$, while satisfied that we do not lose more than a factor of 2. This gives us:

$$\theta \leq \frac{8\kappa}{am} + a\,. \tag{141}$$

Enforcing $\theta \leq 1/2$ can be done easily by choosing $a \leq 1/4$ and $m = 32\kappa/a$. Now, to be able to compare algorithms easily, we want to frame our result in terms of rate factor per gradient computation $\rho$, such that (38) is verified:

$$\mathbb{E}f(\tilde{x}_k) - f(x^*) \leq (1-\rho)^{k(2m+n)}\left(\mathbb{E}f(x_0) - f(x^*)\right) \qquad \forall k \geq 0\,.$$

We define $\rho_b := 1 - \theta$. We want to estimate $\rho$ such that $(1-\rho)^{2m+n} = 1 - \rho_b$. We get that $\rho = 1 - (1-\rho_b)^{\frac{1}{2m+n}}$. Using Bernoulli's inequality, we get:

$$\rho \geq \frac{\rho_b}{2m+n} \geq \frac{1}{2(2m+n)} \geq \frac{1}{4}\min\left\{\frac{1}{2m}, \frac{1}{n}\right\} \geq \frac{1}{4}\min\left\{\frac{a}{64\kappa}, \frac{1}{n}\right\}. \tag{142}$$

This finishes the proof for Corollary 16. ■

***A simpler result for* Kromagnon.** We also define $a$ such that $\gamma = a/4L$. Theorem 15 tells us that:

$$\theta = \frac{\frac{4\kappa}{am} + \frac{a}{2}C_3(1 + \frac{\tau}{16\kappa})}{1 - \frac{a}{2}C_3(1 + \frac{\tau}{16\kappa})}\,. \tag{143}$$

We can once again upper bound $\theta$ by removing its denominator at a reasonable worst-case cost of a factor of 2:

$$\theta \leq \frac{8\kappa}{am} + aC_3(1 + \frac{\tau}{16\kappa})\,. \tag{144}$$

Now, to enforce $\theta \leq 1/2$, we can choose $a \leq \frac{1}{4C_3(1+\frac{\tau}{16\kappa})}$ and $m = \frac{32\kappa}{a}$. We also obtain a rate factor per gradient computation of: $\rho \geq \frac{1}{4}\min\{\frac{a}{64\kappa}, \frac{1}{n}\}$. This finishes the proof of Corollary 17. ■

***Speedup conditions.*** All we have to do now is to compare the rate factors of Svrg and Kromagnon in different regimes. Note that while our convergence result hold for any $a \leq 1/4$ Svrg (or the slightly more complex expression in the case of Kromagnon), the best step size (in terms of number of gradient computations) ensuring $\theta \leq \frac{1}{2}$ is the biggest allowable one – thus this is the one we use for our comparison.

Suppose we are in the "well-conditioned" regime where $n \geq \kappa$. The rate factor of Svrg is $\Omega(1/n)$. To make sure we have a linear speedup, we need the rate factor of Kromagnon to also be $\Omega(1/n)$, which means that:

$$\frac{1}{256\kappa C_3 + 16\tau C_3} = \Omega(\frac{1}{n}) \tag{145}$$

Recalling that $C_3 = 1 + 2\tau\sqrt{\Delta}$, we can rewrite (145) as:

$$\kappa = \mathcal{O}(n)\,; \quad \kappa\tau\sqrt{\Delta} = \mathcal{O}(n)\,; \quad \tau = \mathcal{O}(n)\,; \quad \tau^2\sqrt{\Delta} = \mathcal{O}(n)\,. \tag{146}$$

We can condense these conditions down to:

$$\tau = \mathcal{O}(\frac{n}{\kappa\sqrt{\Delta}})\,; \qquad \tau = \mathcal{O}(\sqrt{n\Delta^{-1/2}})\,. \tag{147}$$

Suppose now we are in the "ill-conditioned" regime, where $\kappa \geq n$. The rate factor of Svrg is now $\Omega(1/\kappa)$. To make sure we have a linear speedup, we need the rate factor of Kromagnon to also be $\Omega(1/\kappa)$, which means that:

$$\frac{1}{256\kappa C_3 + 16\tau C_3} = \Omega(\frac{1}{\kappa}) \tag{148}$$

We can derive the following sufficient conditions:

$$\tau = \mathcal{O}(\frac{1}{\sqrt{\Delta}})\,; \qquad \tau = \mathcal{O}(\kappa)\,. \tag{149}$$

Since $\kappa \geq n$, we obtain the conditions of Corollary 18 and thus finish its proof. ∎

## Appendix D. Hogwild – Proof of Theorem 22 and Corollary 23

### D.1. Proof of Lemma 24 (suboptimality bound on $\mathbb{E}\|g_t\|^2$)

As was the case for proving Lemma 13 and Lemma 19, we simply introduce $f_i'(x^*)$ in $g_t$ to derive our bound.

$$\begin{aligned}\mathbb{E}\|g_t\|^2 = \mathbb{E}\|f_i'(\hat{x}_t)\|^2 &= \mathbb{E}\|f_i'(\hat{x}_t) - f_i'(x^*) + f_i'(x^*)\|^2 \\ &\leq 2\mathbb{E}\|f_i'(\hat{x}_t) - f_i'(x^*)\|^2 + 2\mathbb{E}\|f_i'(x^*)\|^2 \quad (\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2) \\ &\leq 4Le_t + 2\sigma^2\,. \qquad \text{(Hofmann et al. (2015), Eq (7) \& (8))}\end{aligned}$$

∎

## D.2. Proof of Theorem 22 (convergence guarantee and rate of Hogwild)

***Master inequality derivation.*** Once again, we plug Lemma 24 into Lemma 10 which gives us:

$$a_{t+1} \leq (1 - \frac{\gamma\mu}{2})a_t + \gamma^2 C_1(4Le_t + 2\sigma^2) + \gamma^2 C_2 \sum_{u=(t-\tau)_+}^{t-1} (4Le_u + 2\sigma^2) - 2\gamma e_t \,. \tag{150}$$

By grouping the $e_t$ and the $\sigma^2$ terms we get our master inequality (48):

$$a_{t+1} \leq (1 - \frac{\gamma\mu}{2})a_t + (4L\gamma^2 C_1 - 2\gamma)e_t + 4L\gamma^2 C_2 \sum_{u=(t-\tau)_+}^{t-1} e_u + 2\gamma^2\sigma^2(C_1 + \tau C_2) \,.$$

***Contraction inequality derivation ($x_t$).*** We now unroll (48) all the way to $t = 0$ to get:

$$\begin{aligned} a_{t+1} \leq (1 - \frac{\gamma\mu}{2})^{t+1} a_0 &+ \sum_{u=0}^{t} (1 - \frac{\gamma\mu}{2})^{t-u} (4L\gamma^2 C_1 - 2\gamma) e_u \\ &+ \sum_{u=0}^{t} (1 - \frac{\gamma\mu}{2})^{t-u} 4L\gamma^2 C_2 \sum_{v=(u-\tau)_+}^{u-1} e_v \\ &+ \sum_{u=0}^{t} (1 - \frac{\gamma\mu}{2})^{t-u} 2\gamma^2\sigma^2 (C_1 + \tau C_2) \,. \end{aligned} \tag{151}$$

Now we can simplify these terms as follows:

$$\begin{aligned} \sum_{u=0}^{t} (1 - \frac{\gamma\mu}{2})^{t-u} \sum_{v=(u-\tau)_+}^{u-1} e_v &= \sum_{v=0}^{t-1} \sum_{u=v+1}^{\min(t, v+\tau)} (1 - \frac{\gamma\mu}{2})^{t-u} e_v \\ &= \sum_{v=0}^{t-1} (1 - \frac{\gamma\mu}{2})^{t-v} e_v \sum_{u=v+1}^{\min(t, v+\tau)} (1 - \frac{\gamma\mu}{2})^{v-u} \\ &\leq \sum_{v=0}^{t-1} (1 - \frac{\gamma\mu}{2})^{t-v} e_v \tau (1 - \frac{\gamma\mu}{2})^{-\tau} \\ &\leq \tau (1 - \frac{\gamma\mu}{2})^{-\tau} \sum_{v=0}^{t} (1 - \frac{\gamma\mu}{2})^{t-v} e_v \,. \end{aligned} \tag{152}$$

This $(1 - \frac{\gamma\mu}{2})^{-\tau}$ term is easily bounded, as we did in (107) for Asaga. Using Bernoulli's inequality (100), we get that if we assume $\tau \leq \frac{1}{\gamma\mu}$:[38]

$$(1 - \frac{\gamma\mu}{2})^{-\tau} \leq 2 \,. \tag{153}$$

38. While this assumption on $\tau$ may appear restrictive, it is in fact weaker than the condition for a linear speed-up obtained by our analysis in Corollary 23.

We note that the last term in (151) is a geometric sum:

$$\sum_{u=0}^{t}(1-\frac{\gamma\mu}{2})^{t-u}\sigma^2 = \frac{2}{\gamma\mu}\sigma^2\,. \tag{154}$$

We plug (152)–(154) in (151) to obtain (49):

$$a_{t+1} \leq (1-\frac{\gamma\mu}{2})^{t+1}a_0 + (4L\gamma^2C_1 + 8L\gamma^2\tau C_2 - 2\gamma)\sum_{u=0}^{t}(1-\frac{\gamma\mu}{2})^{t-u}e_u + \frac{4\gamma\sigma^2}{\mu}(C_1+\tau C_2)\,.$$

***Contraction inequality derivation ($\hat{x}_t$).*** We now have an contraction inequality for the convergence of $x_t$ to $x^*$. However, since this quantity does not exist (except if we fix the number of iterations prior to running the algorithm and then wait for all iterations to be finished – an unwieldy solution), we rather want to prove that $\hat{x}_t$ converges to $x^*$. In order to do this, we use the simple following inequality:

$$\|\hat{x}_t - x^*\|^2 \leq 2a_t + 2\|\hat{x}_t - x_t\|^2\,. \tag{155}$$

We already have a contraction bound on the first term (49). For the second term, we combine (65) with Lemma 24 to get:

$$\mathbb{E}\|\hat{x}_t - x_t\|^2 \leq 4L\gamma^2C_1\sum_{u=(t-\tau)_+}^{t-1} e_u + 2\gamma^2\tau\sigma^2\,. \tag{156}$$

To make it easier to combine with (49), we rewrite (156) as:

$$\begin{aligned}\mathbb{E}\|\hat{x}_t - x_t\|^2 &\leq 4L\gamma^2C_1(1-\frac{\gamma\mu}{2})^{-\tau}\sum_{u=(t-\tau)_+}^{t-1}(1-\frac{\gamma\mu}{2})^{t-1-u}e_u + 2\gamma^2\tau\sigma^2 \\ &\leq 8L\gamma^2C_1\sum_{u=(t-\tau)_+}^{t-1}(1-\frac{\gamma\mu}{2})^{t-1-u}e_u + 2\gamma^2\tau\sigma^2 \\ &\leq 8L\gamma^2C_1\sum_{u=0}^{t-1}(1-\frac{\gamma\mu}{2})^{t-1-u}e_u + 2\gamma^2\tau\sigma^2\,.\end{aligned} \tag{157}$$

Combining (49) and (157) gives us (50):

$$\begin{aligned}\mathbb{E}\|\hat{x}_t - x^*\|^2 \leq (1-\frac{\gamma\mu}{2})^{t+1}2a_0 &+ (\frac{8\gamma(C_1+\tau C_2)}{\mu} + 4\gamma^2C_1\tau)\sigma^2 \\ &+ (24L\gamma^2C_1 + 16L\gamma^2\tau C_2 - 4\gamma)\sum_{u=0}^{t}(1-\frac{\gamma\mu}{2})^{t-u}e_u\,.\end{aligned}$$

***Maximum step size condition on $\gamma$.*** To prove Theorem 22, we need an inequality of the following type: $\mathbb{E}\|\hat{x}_t - x_t\|^2 \leq (1-\rho)^t a + b$. To give this form to Equation (50), we need to remove all the $(e_u, u < t)$ terms from its right-hand side. To safely do so, we need to enforce that all these terms are negative, hence that:

$$24L\gamma^2 C_1 + 16L\gamma^2 \tau C_2 - 4\gamma \leq 0\,. \tag{158}$$

Plugging the values of $C_1$ and $C_2$ we get:

$$4L\mu\tau(1 + \sqrt{\Delta}\tau)\gamma^2 + 6L(1 + 2\sqrt{\Delta}\tau)\gamma - 1 \leq 0\,. \tag{159}$$

As in our Asaga analysis, this reduces to a second-order polynomial sign condition. We remark again that since $\gamma \geq 0$, we can upper bound our terms in $\gamma$ and $\gamma^2$ in this polynomial, which will still give us sufficient conditions for convergence. This means if we define $C_3 := 1 + 2\sqrt{\Delta}\tau$, a sufficient condition is:

$$4L\mu\tau C_3\gamma^2 + 6LC_3\gamma - 1 \leq 0\,. \tag{160}$$

The discriminant of this polynomial is always positive, so $\gamma$ needs to be between its two roots. The smallest is negative, so the condition is not relevant to our case (where $\gamma > 0$). By solving analytically for the positive root $\phi$, we get an upper bound condition on $\gamma$ that can be used for any overlap $\tau$ and guarantee convergence. This positive root is:

$$\phi = \frac{3}{4}\frac{\sqrt{1 + \frac{\mu\tau}{2LC_3}} - 1)}{LC_3}\,. \tag{161}$$

We simplify it further by using (112):

$$\phi \geq \frac{3}{16LC_3\sqrt{1 + \frac{\tau}{2\kappa C_3}}}\,. \tag{162}$$

This finishes the proof for Theorem 22. ∎

## D.3. Proof of Theorem 21 (convergence result for serial Sgd)

In order to analyze Corollary 23, we need to derive the maximum allowable step size for serial Sgd. Note that Sgd verifies a simpler contraction inequality than Lemma 10. For all $t \geq 0$:

$$a_{t+1} \leq (1 - \gamma\mu)a_t + \gamma^2\mathbb{E}\|g_t\|^2 - 2\gamma e_t\,, \tag{163}$$

Here, the contraction factor is $(1 - \gamma\mu)$ instead of $(1 - \frac{\gamma\mu}{2})$ because $\hat{x}_t = x_t$ so there is no need for a triangle inequality to get back $\|x_t - x^*\|^2$ from $\|\hat{x}_t - x^*\|^2$ after we apply our strong convexity bound in our initial recursive inequality (see Section B.1). Lemma 24 also holds for serial Sgd. By plugging it into (163), we get:

$$a_{t+1} \leq (1 - \gamma\mu)a_t + (4L\gamma^2 - 2\gamma)e_t + 2\gamma^2\sigma^2\,. \tag{164}$$

We then unroll (164) until $t = 0$ to get:

$$a_{t+1} \leq (1 - \gamma\mu)^{t+1}a_0 + (4L\gamma^2 - 2\gamma)\sum_{u=0}^{t}(1 - \gamma\mu)^{t-u}e_u + 2\frac{\gamma\sigma^2}{\mu}\,. \tag{165}$$

To get linear convergence up to a ball around the optimum, we need to remove the $(e_u, 0 \leq u \leq t)$ terms from the right-hand side of the equation. To safely do this, we need these terms to be negative, i.e. $4L\gamma^2 - 2\gamma \leq 0$. We can then trivially derive the condition on $\gamma$ to achieve linear convergence: $\gamma \leq \frac{1}{2L}$.

We see that if $\gamma = a/L$ with $a \leq 1/2$, Sgd converges at a geometric rate of at least: $\rho(a) = a/\kappa$, up to a ball of radius $2\frac{\gamma\sigma^2}{\mu}$ around the optimum. Now, to make sure we reach $\epsilon$-accuracy, we need $\frac{2\gamma\sigma^2}{\mu} \leq \epsilon$, i.e. $\gamma \leq \frac{\epsilon\mu}{2\sigma^2}$. All told, in order to get linear convergence to $\epsilon$-accuracy, serial Sgd requires $\gamma \leq \min\left\{\frac{1}{2L}, \frac{\epsilon\mu}{2\sigma^2}\right\}$.

### D.4. Proof of Corollary 23 (speedup regimes for Hogwild)

The convergence rate of both Sgd and Hogwild is directly proportional to the step size. Thus, in order to make sure Hogwild is linearly faster than Sgd for any reasonable step size, we need to show that the maximum allowable step size ensuring linear convergence for Hogwild – given in Theorem 22 – is of the same order as the one for Sgd, $\mathcal{O}(1/L)$. Recalling that $\gamma = \frac{a}{L}$, we get the following sufficient condition: $a^*(\tau) = \mathcal{O}(1)$.

Given (46), the definition of $a^*(\tau)$, we require both:

$$\tau\sqrt{\Delta} = \mathcal{O}(1)\,; \qquad \sqrt{1 + \frac{1}{2\kappa}\min\{\frac{1}{\sqrt{\Delta}}, \tau\}} = \mathcal{O}(1)\,. \tag{166}$$

This gives us the final condition on $\tau$ for a linear speedup: $\tau = \mathcal{O}(\min\{\frac{1}{\sqrt{\Delta}}, \kappa\})$.

To finish the proof of Corollary 23, we only have to show that under this condition, the size of the ball is of the same order regardless of the algorithm used.

Using $\gamma\mu\tau \leq 1$ and $\tau \leq \frac{1}{\sqrt{\Delta}}$, we get that $(\frac{8\gamma(C_1+\tau C_2)}{\mu} + 4\gamma^2 C_1\tau)\sigma^2 = \mathcal{O}(\frac{\gamma\sigma^2}{\mu})$, which finishes the proof of Corollary 23. Note that these two conditions are weaker than $\tau = \mathcal{O}(\min\{\frac{1}{\sqrt{\Delta}}, \kappa\})$, which allows us to get better bounds in case we want to reach $\epsilon$-accuracy with $\frac{\epsilon\mu}{\sigma^2} \ll \frac{1}{2L}$. ■

## Appendix E. On the Difficulty of Parallel Lagged Updates

In the implementation presented in Schmidt et al. (2016), the dense part $(\bar{\alpha})$ of the updates is deferred. Instead of writing dense updates, counters $c_d$ are kept for each coordinate of the parameter vector – which represent the last time these variables were updated – as well as the average gradient $\bar{\alpha}$ for each coordinate. Then, whenever a component $[\hat{x}]_d$ is needed (in order to compute a new gradient), we subtract $\gamma(t - c_d)[\bar{\alpha}]_d$ from it and $c_d$ is set to $t$. It is possible to do this without modifying the algorithm because $[\bar{\alpha}]_d$ only changes when $[\hat{x}]_d$ also does.

In the sequential setting, this results in the same iterations as performing the updates in a dense fashion, since the coordinates are only stale when they are not used. Note that at the end of an execution all counters have to be subtracted at once to get the true final parameter vector (and to bring every $c_d$ counter to the final $t$).

In the parallel setting, several issues arise:

- two cores might be attempting to correct the lag at the same time. In which case since updates are done as additions and not replacements (which is necessary to ensure that there are no overwrites), the lag might be corrected multiple times, i.e. overly corrected.

- we would have to read and write atomically to each $[\hat{x}_d], c_d, [\bar{\alpha}]_d$ triplet, which is highly impractical.
- we would need to have an explicit global counter, which we do not in Asaga (our global counter $t$ being used solely for the proof).
- in the dense setting, updates happen coordinate by coordinate. So at time $t$ the number of $\bar{\alpha}$ updates a coordinate has received from a fixed past time $c_d$ is a random variable, which may differs from coordinate to coordinate. Whereas in the lagged implementation, the multiplier is always $(t - c_d)$ which is a constant (conditional to $c_d$), which means a potentially different $\hat{x}_t$.
- the trick used in Reddi et al. (2015) for asynchronous parallel Svrg does not apply here because it relies on the fact that the "reference" gradient term in Svrg is constant throughout a whole epoch, which is not the case for Saga.

All these points mean both that the implementation of such a scheme in the parallel setting would be impractical, and that it would actually yields a different algorithm than the dense version, which would be even harder to analyze. This is confirmed by Pan et al. (2016), where the authors tried to implement a parallel version of the lagged updates scheme and had to alter the algorithm to succeed, obtaining an algorithm with suboptimal performance as a result.

## Appendix F. Additional Empirical Details

### F.1. Detailed Description of Data Sets

We run our experiments on four data sets. In every case, we run logistic regression for the purpose of binary classification.

***RCV1* ($n = 697,641$, $d = 47,236$).** The first is the Reuters Corpus Volume I (RCV1) data set (Lewis et al., 2004), an archive of over 800,000 manually categorized newswire stories made available by Reuters, Ltd. for research purposes. The associated task is a binary text categorization.

***URL* ($n = 2,396,130$, $d = 3,231,961$).** Our second data set was first introduced in Ma et al. (2009). Its associated task is a binary malicious URL detection. This data set contains more than 2 million URLs obtained at random from Yahoo's directory listing (for the "benign" URLs) and from a large Web mail provider (for the "malicious" URLs). The benign to malicious ratio is 2. Features include lexical information as well as metadata. This data set was obtained from the libsvmtools project.[39]

39. `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`

***Covertype* (*$n = 581,012$*, *$d = 54$*).** On our third data set, the associated task is a binary classification problem (down from 7 classes originally, following the pre-treatment of Collobert et al., 2002). The features are cartographic variables. Contrarily to the first two, this is a dense data set.

***Realsim* (*$n = 73,218$*, *$d = 20,958$*).** We only use our fourth data set for non-parallel experiments and a specific compare-and-swap test. It constitutes of UseNet articles taken from four discussion groups (simulated auto racing, simulated aviation, real autos, real aviation).

### F.2. Implementation Details

***Hardware.*** All experiments were run on a Dell PowerEdge 920 machine with 4 Intel Xeon E7-4830v2 processors with 10 2.2GHz cores each and 384GB 1600 MHz RAM.

***Software.*** All algorithms were implemented in the Scala language and the software stack consisted of a Linux operating system running Scala 2.11.7 and Java 1.6.

We chose this expressive, high level language for our experimentation despite its typical 20x slower performance compared to C because our primary concern was that the code may easily be reused and extended for research purposes (which is harder to achieve with low level, heavily optimized C code; especially for error prone parallel computing).

As a result our timed experiments exhibit sub-optimal running times, e.g. compared to Konečný and Richtárik (2013). This is as we expected. The observed slowdown is both consistent across data sets (roughly 20x) and with other papers that use Scala code (e.g. Mania et al. 2017, Ma et al. 2015, Fig. 2).

Despite this slowdown, our experiments show state-of-the-art results in convergence per number of iterations. Furthermore, the speed-up patterns that we observe for our implementation of Hogwild and Kromagnon are similar to the ones given in Mania et al. (2017); Niu et al. (2011); Reddi et al. (2015) (in various languages).

The code we used to run all the experiments is available at `http://www.di.ens.fr/sierra/research/asaga/`.

***Necessity of compare-and-swap operations.*** Interestingly, we have found necessary to use compare-and-swap instructions in the implementation of Asaga. In Figure 7, we display suboptimality plots using non-thread safe operations and compare-and-swap (CAS) operations. The non-thread safe version starts faster but then fails to converge beyond a specific level of suboptimality, while the compare-and-swap version does converges linearly up to machine precision.

For *compare-and-swap* instructions we used the `AtomicDoubleArray` class from the Google library `Guava`. This class uses an `AtomicLongArray` under the hood (from package `java.util.concurrent.atomic` in the standard Java library), which does indeed benefit from lower-level CPU-optimized instructions.

***Efficient storage of the $\alpha_i$.*** Storing $n$ gradient may seem like an expensive proposition, but for linear predictor models, one can actually store a single scalar per gradient (as proposed in Schmidt et al., 2016), which is what we do in our implementation of Asaga.

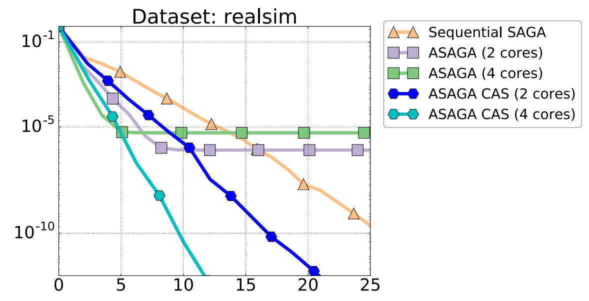


Figure 7: **Compare and swap in the implementation of Asaga.** Suboptimality as a function of time for Asaga, both using compare-and-swap (CAS) operations and using standard operations. The graph reveals that CAS is indeed needed in a practical implementation to ensure convergence to a high precision.

### F.3. Biased Update in the Implementation

In the implementation detailed in Algorithm 2, $\bar{\alpha}$ is maintained in memory instead of being recomputed for every iteration. This saves both the cost of reading every data point for each iteration and of computing $\bar{\alpha}$ for each iteration.

However, this removes the unbiasedness guarantee. The problem here is the definition of the expectation of $\hat{\alpha}_i$. Since we are sampling uniformly at random, the average of the $\hat{\alpha}_i$ is taken at the precise moment when we read the $\alpha_i^t$ components. Without synchronization, between two reads to a single coordinate in $\alpha_i$ and in $\bar{\alpha}$, new updates might arrive in $\bar{\alpha}$ that are not yet taken into account in $\alpha_i$. Conversely, writes to a component of $\alpha_i$ might precede the corresponding write in $\bar{\alpha}$ and induce another source of bias.

In order to alleviate this issue, we can use coordinate-level locks on $\alpha_i$ and $\bar{\alpha}$ to make sure they are always synchronized. Such low-level locks are quite inexpensive when $d$ is large, especially when compared to vector-wide locks.

However, as previously noted, experimental results indicate that this fix is not necessary.

## References

Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, 1989.

R. Collobert, S. Bengio, and Y. Bengio. A parallel mixture of SVMs for very large scale problems. *Neural Computation*, 2002.

Christopher De Sa, Ce Zhang, Kunle Olukotun, and Christopher Ré. Taming the wild: A unified analysis of Hogwild!-style algorithms. In *Advances in Neural Information Processing Systems 28 (NIPS)*, 2015.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems 27 (NIPS)*, 2014.

John C. Duchi, Sorathan Chaturapruek, and Christopher Ré. Asynchronous stochastic convex optimization. In *Advances in Neural Information Processing Systems 28 (NIPS)*, 2015.

Thomas Hofmann, Aurelien Lucchi, Simon Lacoste-Julien, and Brian McWilliams. Variance reduced stochastic gradient descent with neighbors. In *Advances in Neural Information Processing Systems 28 (NIPS)*, 2015.

Cho-Jui Hsieh, Hsiang-Fu Yu, and Inderjit Dhillon. PASSCoDe: Parallel asynchronous stochastic dual co-ordinate descent. In *Proceedings of the* 32nd *International Conference on Machine Learning (ICML)*, 2015.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26 (NIPS)*, 2013.

Jakub Konečný and Peter Richtárik. Semi-stochastic gradient descent methods. *arXiv:1312.1666*, 2013.

Nicolas Le Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems 25 (NIPS)*, 2012.

Rémi Leblond, Fabian Pedregosa, and Simon Lacoste-Julien. ASAGA: Asynchronous parallel SAGA. In *Proceedings of the* 20th *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 2004.

Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems 28 (NIPS)*, 2015.

Ji Liu, Stephen J. Wright, Christopher Ré, Victor Bittorf, and Srikrishna Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. *Journal of Machine Learning Research*, 2015.

Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I. Jordan, Peter Richtarik, and Martin Takac. Adding vs. averaging in distributed primal-dual optimization. In *Proceedings of the* 32nd *International Conference on Machine Learning (ICML)*, 2015.

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 *International Conference on Machine Learning (ICML)*, 2009.

Horia Mania, Xinghao Pan, Dimitris Papailiopoulos, Benjamin Recht, Kannan Ramchandran, and Michael I. Jordan. Perturbed iterate analysis for asynchronous stochastic optimization. *SIAM Journal on Optimization*, 2017.

Eric Moulines and Francis R. Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems 24 (NIPS)*, 2011.

Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm. In *Advances in Neural Information Processing Systems 27 (NIPS)*, 2014.

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 24 (NIPS)*, 2011.

Xinghao Pan, Maximilian Lam, Stephen Tu, Dimitris Papailiopoulos, Ce Zhang, Michael I. Jordan, Kannan Ramchandran, Chris Re, and Benjamin Recht. Cyclades: Conflict-free asynchronous machine learning. In *Advances in Neural Information Processing Systems 29 (NIPS)*, 2016.

Fabian Pedregosa, Rémi Leblond, and Simon Lacoste-Julien. Breaking the nonsmooth barrier: A scalable parallel method for composite optimization. In *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017.

Sashank J. Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Póczos, and Alex Smola. On variance reduction in stochastic gradient descent and its asynchronous variants. In *Advances in Neural Information Processing Systems 28 (NIPS)*, 2015.

Sashank J. Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Póczós, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *Proceedings of the* 33rd *International Conference on Machine Learning (ICML)*, 2016.

M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 2016.

Mark Schmidt. Convergence rate of stochastic gradient with constant step size. *UBC Technical Report*, 2014.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *Journal of Machine Learning Research*, 2013.

Shen-Yi Zhao and Wu-Jun Li. Fast asynchronous parallel stochastic gradient descent. In *Proceedings of the* 30th *AAAI Conference on Artificial Intelligence*, 2016.