PARALLEL STOCHASTIC ASYNCHRONOUS COORDINATE DESCENT: TIGHT BOUNDS ON THE POSSIBLE PARALLELISM*

YUN KUEN CHEUNG†, RICHARD COLE‡, AND YIXIN TAO§

Abstract. Several works have shown linear speedup is achieved by an asynchronous parallel implementation of stochastic coordinate descent so long as there is not too much parallelism. More specifically, it is known that if all updates are of similar duration, then linear speedup is possible with up to $\Theta(L_{\max}\sqrt{n}/L_{\overline{\text{res}}})$ processors, where $L_{\max}$ and $L_{\overline{\text{res}}}$ are suitable Lipschitz parameters. This paper shows the bound is tight for almost all possible values of these parameters.

Key words. stochastic asynchronous coordinate descent, parallelism bound

AMS subject classifications. 90C25, 68W10, 60G50, 68Q99

1. Introduction.

Very large scale optimization problems have arisen in many areas such as machine learning. A natural approach for solving these huge problems is to employ parallel and more specifically asynchronous parallel algorithms. As common wisdom suggests, when a small number of processors are used in these algorithms, (linear) speedup can be achieved; but when too many processors are involved and when they are not properly coordinated, there may be undesirable outcomes. (Recall that speedup is defined as the ratio of the algorithm's execution time on a single core and its parallel execution time. Linear speedup means the speedup is linear in the number of cores.)[1] Typically, the bound on the number of processors is implicit and is expressed in terms of the maximum number q of basic iterations, namely single coordinate updates, that can overlap. In many scenarios q will be a small multiple of the number of processors or cores. In many earlier works the notation τ was used instead of q.[2]

This paper considers asynchronous implementations of stochastic coordinate descent (SCD) applied to smooth convex functions $f : \mathbb{R}^n \to \mathbb{R}$. Several recent works [3, 2, 1] quantify how large q can be in this context while guaranteeing linear speedup.[3] More precisely, they showed: if at any time at most $q \leq \widetilde{q}$ updates can overlap, then linear speedup is guaranteed. The goal in these works was to demonstrate as large a value of $\widetilde{q}$ as possible. Note that these results provide *lower* bounds on the actual value of $\widetilde{q}$.

*Submitted to the editors DATE. Authors are listed in alphabetical order.

Funding: Yun Kuen Cheung would like to acknowledge Singapore NRF 2018 Fellowship NRF-NRFF2018-07 and MOE AcRF Tier 2 Grant 2016-T2-1-170. The work of Richard Cole and Yixin Tao was supported in part by NSF Grants CCF-1527568 and CCF-1909538.

†Royal Holloway University of London (yunkuen.cheung@rhul.ac.uk, http://cs.rhul.ac.uk/~cheung/) Part of the work done while this author was a postdoctoral fellow at Max-Planck Institute for Informatics, Saarland Informatics Campus and Singapore University of Technology and Design, and also during two visits to the Courant Institute, NYU in the summers of 2017 and 2018.

‡Courant Institute, NYU (cole@cs.nyu.edu, https://cs.nyu.edu/cole/)

§Courant Institute, NYU (yt851@nyu.edu, https://tomtao26.github.io/). Yixin Tao is now a postdoc at LSE.

[1] In optimization, we compare the times of two executions that achieve a given level of accuracy.

[2] We chose the notation q to emphasize its likely similarity to p, the number of processors.

[3] Many of these results hold for *composite* functions, i.e., functions of the form $f(\mathbf{x}) = g(\mathbf{x}) + \sum_{k=1}^{n} \Psi_k(x_k)$, where $g : \mathbb{R}^n \to \mathbb{R}$ is a convex function with a continuous gradient, and each $\Psi_k : \mathbb{R} \to \mathbb{R}$ is a univariate convex function, but may be non-smooth.

The best existing lower bound is $\widetilde{q} = \Omega(\sqrt{n}L_{\max}/L_{\overline{\text{res}}})$, where $L_{\max}$ and $L_{\overline{\text{res}}}$ are Lipschitz parameters defined in Section 2.[4] Intuitively, one can view this as a lower bound on the possible parallelism supporting linear speedup; for if there are p processors at hand, and if the durations of the updates vary by at most a factor of d, then $q \leq (d+1)(p-1)$, so achieving linear speedup with up to q updates overlapping implies linear speedup occurs with $p = 1 + q/(d+1)$ processors.

We present the first work concerning the inverse problem: to identify a value $\overline{q}$, such that if $q \geq \overline{q}$, then this can lead to an undesirable outcome.

***Main result*:** *The lower bound of $\Omega(\sqrt{n}L_{\max}/L_{\overline{\text{res}}})$ is asymptotically tight.* (An undesirable outcome can occur if $\overline{q} \geq \bar{c} \cdot \sqrt{n}L_{\max}/L_{\overline{\text{res}}}$ for a sufficiently large constant $\bar{c}$.)

We will present an adversarial family of functions with the following property: if q exceeds $\Theta(\sqrt{n}L_{\max}/L_{\overline{\text{res}}})$ significantly, then there is an asynchronous schedule for which with high probability very rapid divergence occurs for a very long time. The function family uses a dimensionless parameter $\epsilon = \Theta(\frac{L_{\overline{\text{res}}}}{L_{\max}\sqrt{n}})$, which is approximately the inverse of the possible parallelism.

We use the following function family, $f_\epsilon : \mathbb{R}^n \to \mathbb{R}$:

$$f_\epsilon(\mathbf{x}) = \frac{1-\epsilon}{2} \cdot \sum_{i=1}^{n} (x_i)^2 \;+\; \frac{\epsilon}{2} \cdot \left(\sum_{i=1}^{n} x_i \right)^2 , \tag{1.1}$$

for any ϵ satisfying $4/n \leq \epsilon < 1$. As we shall see, $L_{\max} = 1$ and $L_{\overline{\text{res}}} = \Omega(\epsilon\sqrt{n})$ for this function family; thus the existing lower bound on the parallelism achieving linear speedup is $\Omega(\sqrt{n}L_{\max}/L_{\overline{\text{res}}}) = \Omega(1/\epsilon)$. To obtain bounds using arbitrary values of $L_{\max}$ one can simply multiply f_ϵ by $L_{\max}$, which also increases $L_{\overline{\text{res}}}$ by an $L_{\max}$ factor.

Next, we discuss more precisely how we achieve this result. Recall that the performance of a sequential SCD algorithm is expressed in terms of its convergence rate. On strongly convex functions, it has a linear convergence rate, meaning that each update reduces the expected value of the difference $f(\mathbf{x}) - f^*$ by at least an $(1-\alpha/n)$ multiplicative factor, for some constant $\alpha > 0$, where f^* denotes the minimum value of the function. Consequently,

$$\mathbb{E}\left[f(\mathbf{x}^t) - f^*\right] \quad \leq \quad \left(1 - \frac{\alpha}{n}\right)^t \cdot \left(f(\mathbf{x}^0) - f^*\right). \tag{1.2}$$

For our proposed function f_ϵ, which is strongly convex, we will show that for a suitable initial point $\mathbf{x}^0$, for some constant $\alpha' \geq \alpha$,

$$\mathbb{E}\left[f_\epsilon(\mathbf{x}^t) - f_\epsilon^*\right] \quad \geq \quad \left(1 - \frac{\alpha'}{n}\right)^t \cdot \left(f_\epsilon(\mathbf{x}^0) - f_\epsilon^*\right), \tag{1.3}$$

and hence sequential SCD achieves no more than a linear convergence rate in general. For the function f_ϵ, we will show that $\alpha = \frac{1}{3}$ and $\alpha' = 2$.

To achieve linear speedup with a parallel algorithm means that the same convergence rate holds, up to constant factor, i.e., the α might be reduced by a constant factor $c \geq 1$, but no more:

$$\mathbb{E}\left[f(\mathbf{x}^t) - f^*\right] \quad \leq \quad \left(1 - \frac{\alpha}{cn}\right)^t \cdot \left(f(\mathbf{x}^0) - f^*\right), \tag{1.4}$$

[4] Here, we focus on the case where the step-size used in the asychronous SCD algorithm is $1/L_{\max}$; we will discuss the cases with smaller step-sizes later in the introduction.

where t is now the overall number of iterations performed by the various cores.[5] This means that to guarantee a particular accuracy, the total number of iterations for a parallel execution is no more than a constant multiple of the number of iterations needed on a single core (so long as $\alpha \leq n/2$).

Prior work has shown that linear speedup is achieved when $q \leq \tilde{c}\sqrt{n}L_{\max}/L_{\overline{\text{res}}}$ for some constant $\tilde{c} > 0$. To achieve our main result, we show that for the function family f_ϵ, when $q \geq \bar{c}\sqrt{n}L_{\max}/L_{\overline{\text{res}}}$ for some constant $\bar{c} > \tilde{c}$, as an adversary, it is possible to pick asynchronous schedules such that for all $t \leq n^{10}$ (or more generally, for any constant $\hat{c} \geq 1$, for all $t \leq n^{\hat{c}}$),

$$\mathbb{E}\left[f_\epsilon(\mathbf{x}^t) - f_\epsilon^*\right] \;\geq\; \Omega(4^{t/q}) \cdot (f_\epsilon(\mathbf{x}^0) - f_\epsilon^*)$$

for large enough n (in general, when $n = \Omega(c^4)$). This indicates that when q is too large, linear speedup cannot be achieved in worst-case scenarios.

The above upper bound on q holds when the step-size is $1/L_{\max}$. One might wonder what would happen if we reduced the step-size to $1/\Gamma$ for some $\Gamma \geq L_{\max}$. Would the permissible parallelism bound increase significantly, thereby improving the overall speedup? In fact, we show that the upper bound on q increases to at most $\mathcal{O}(\Gamma\sqrt{n}/L_{\text{res}})$, for any $L_{\max} \leq \Gamma \leq \mathcal{O}(L_{\text{res}}\sqrt{n})$. Since this increase is by a factor of at most $\mathcal{O}(\Gamma/L_{\max})$, but the step-size is reduced by a factor of $\Gamma/L_{\max}$, the overall speedup cannot improve by more than a constant factor. This upper bound is also asymptotically tight, as there were matching lower bounds for these choices of step-sizes [1].

Prior Work and Asynchrony Models. First, note that the bound on q is ensuring the asynchrony is bounded, and so we call it *q-bounded asynchrony*. Some requirement of this sort is unavoidable, otherwise there could be updates of arbitrarily long duration, which, when they commit, could undo an arbitrary amount of progress.

In addition to the q-bounded asynchrony assumption, we need to specify how the asynchronous environment affects the read operations. There are two models concerning how coordinates are read in asynchronous environments, namely "consistent" and "inconsistent" reads. Our upper bound applies to both models. Next, we discuss their differences.

Liu et al. [3] gave the first bound on the parallel performance of asynchronous SCD on convex functions, showing linear speedup when $q = O(\sqrt{n}L_{\max}/L_{\text{res}})$ assuming a consistent read model, where L_{res} is another Lipschitz parameter defined in Section 2. We note that $L_{\text{res}} = L_{\overline{\text{res}}}$ for the function family f_ϵ we will be analyzing in this paper. In fact, as we shall see, $L_{\overline{\text{res}}}$ is equal to L_{res} on all quadratic functions f, i.e., f is of the form $\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} + \mathbf{b}^\mathsf{T}\mathbf{x} + \text{constant}$, where $\mathbf{A}$ is an $n \times n$ matrix, and $\mathbf{b}$ is an n-vector. In the consistent read model, all the coordinate values a processor reads when performing a single update on one coordinate may be out of date, but they must have been simultaneously current at some moment. To make this more precise, we view the updates as committing at integer times $t = 1, 2, \ldots$, and we write $\mathbf{x}^t$ to be the value of $\mathbf{x}$ after the update at time t. The consistent reads model requires the vector of $\mathbf{x}$ values used by the time t update to be of the form $\mathbf{x}^{t-\tau}$ for some $\tau \geq 1$.

Consistent reads create a substantial constraint on the asynchrony, and so subsequent works sought to avoid this assumption. To this end, Liu and Wright [2] proposed

[5] Liu et al. [3] and Liu and Wright [2] call this near-linear speedup, reserving the term linear speedup for the case when $c = 1$.

the inconsistent reads model. Allowing inconsistent reads means that the $\tilde{\mathbf{x}}$ values used by the time t update can be any collection of the form $(x_1^{t-\tau_1}, \cdots, x_n^{t-\tau_n})$, where the τ_j's can be distinct; the q-bounded asynchrony assumption implies that $1 \leq \tau_j \leq q$ for each j. Liu and Wright showed that linear speedup (including the more general case of composite functions) can be achieved for $q = O(n^{1/4}\sqrt{L_{\max}}/\sqrt{L_{\text{res}}})$, i.e., the square root of the previous bound. There remained several constraints on the possible asynchrony, in addition to the q-bounded asynchrony, as pointed out by Mania et al. [4] and subsequently by Sun et al. [5]. The latter works also gave analyses removing some or all of these constraints, but at the cost of reducing the bound on q. Finally, Cheung, Cole and Tao [1] gave an analysis achieving linear speedup for $q = O(\sqrt{n}L_{\max}/L_{\overline{\text{res}}})$, again for composite functions, with the only constraint being the q-bounded asynchrony.

In this paper we show the bounds in [3] and [1] are asymptotically tight for almost all possible values of $L_{\max}$, L_{res}, and $L_{\overline{\text{res}}}$ for the function family (1.1).

2. Notation. Let $\mathbf{e}_j$ denote the n-vector in which the j-th entry is 1 and every other entry is 0.

DEFINITION 2.1. *For any coordinates j, k, the function f is L_{jk}-Lipschitz-smooth if for any $\mathbf{x} \in \mathbb{R}^n$ and $r \in \mathbb{R}$, $|\nabla_k f(\mathbf{x}+r\cdot\mathbf{e}_j) - \nabla_k f(\mathbf{x})| \leq L_{jk}\cdot|r|$; it is L_{res}-Lipschitz-smooth if, for all j, $\|\nabla f(\mathbf{x}+r\cdot\mathbf{e}_j) - \nabla f(\mathbf{x})\| \leq L_{\text{res}}\cdot|r|$. Finally, $L_{\max} := \max_j L_{jj}$ and $L_{\overline{\text{res}}} := \max_k \left(\sum_{j=1}^n (L_{kj})^2\right)^{1/2}$.*

Observe that for the function f_ϵ we are considering,

$$\nabla_j f_\epsilon(\mathbf{x}) \;=\; (1-\epsilon)\cdot x_j + \epsilon \cdot \sum_{i=1}^n x_i. \tag{2.1}$$

Consequently, $L_{\max} = 1$, and $L_{\overline{\text{res}}} = \sqrt{1+(n-1)\epsilon^2} = \Theta(\epsilon\sqrt{n})$, for $\epsilon = \Omega(1/\sqrt{n})$.

The difference between L_{res} and $L_{\overline{\text{res}}}$. In general, $L_{\overline{\text{res}}} \geq L_{\text{res}}$. $L_{\text{res}} = L_{\overline{\text{res}}}$ when the rates of change of the gradient are constant, as for example in quadratic functions such as $\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} + \mathbf{b}^\mathsf{T}\mathbf{x} + c$. We refer the reader to [1] for a discussion of why $L_{\overline{\text{res}}}$ is needed in general for the analysis in [1].

Next, we define strong convexity.

DEFINITION 2.2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. f is strongly convex with parameter $\mu > 0$, if for all x, y, $f(y) - f(x) \geq \langle\, \nabla f(x)\ ,\ y - x\,\rangle + \frac{1}{2}\mu||y-x||^2$.*

A simple calculation shows that the parameter μ for our function f_ϵ has value $(1-\epsilon)$; see Lemma A.1 in Appendix A.

The update rule. Recall that in a standard coordinate descent, be it sequential or parallel and synchronous, the update rule, applied to coordinate j, first computes the accurate gradient $\nabla_j f(\mathbf{x}^{t-1})$, and then performs the update given below.

$$x_j^t \;\leftarrow\; x_j^{t-1} - \frac{\nabla_j f(\mathbf{x}^{t-1})}{\Gamma} \tag{2.2}$$

and for all $k \neq j$, $x_k^t \leftarrow x_k^{t-1}$, where $\Gamma \geq L_{\max}$ is a parameter controlling the step size.

However, in an asynchronous environment, an updating processor might retrieve outdated information $\tilde{\mathbf{x}}$ instead of $\mathbf{x}^{t-1}$, so the gradient the processor computes will be $\nabla_j f(\tilde{\mathbf{x}})$, instead of the accurate value $\nabla_j f(\mathbf{x}^{t-1})$. Hence the update rule is in the asynchronous environment is

$$x_j^t \leftarrow x_j^{t-1} - \frac{\nabla_j f(\tilde{\mathbf{x}})}{\Gamma}. \tag{2.3}$$

We want to show that for any fixed constants $c_1, c_2 \geq 1$, $f_\epsilon(\mathbf{x}^t) - f_\epsilon^*$ is rapidly growing for $t \leq q \cdot n_1^c$ with probability at least $1 - 1/n^{c_2}$.

2.1. The Stochastic Asynchronous Coordinate Descent (SACD) Algorithm. The coordinate descent process starts at an initial point $\mathbf{x}^0 = (x_1^0, x_2^0, \cdots, x_n^0)$. Multiple processors then iteratively update the coordinate values, and for our analysis we assume that at each time, there is exactly one coordinate value being updated, which we can do, as we are choosing the asynchronous schedule.

Algorithm 2.1 SACD Algorithm for Smooth Functions.

Input: The initial point $\mathbf{x}^0 = (x_1^0, x_2^0, \cdots, x_n^0)$.

Multiple processors use a shared memory. Each processor iteratively repeats the following four-step procedure, with no global coordination among them:

Step 1: Choose a coordinate $j \in \{1, 2, \cdots, n\}$ uniformly at random.
Step 2: Retrieve coordinate values $\tilde{x}$ from the shared memory.
Step 3: Compute the gradient $\nabla_j f(\tilde{\mathbf{x}})$.
Step 4: Update coordinate j using rule (2.3) by atomic addition.

2.2. Asynchrony Assumptions, Basic Set-up and Terminology Used in the Construction. In our construction, updates are made in phases. In each phase, q updates are made in parallel by q processors. As in Step 1 of Algorithm 2.1, each processor chooses a coordinate to update. Since these processors are uncoordinated, they choose coordinates randomly and independently, and thus it is possible that some of them choose the same coordinate within a phase. Then, for every coordinate, each processor retrieves the value that was up to date at the end of the previous phase.

Note that $\mathbf{x}^{q\ell}$ denote the up-to-date values immediately after the ℓ-th phase. The starting point $\mathbf{x}^0$ can be viewed as the up-to-date values following the (non-existent) 0-th phase. For each $\ell \geq 0$, in the $(\ell+1)$-st phase, each of the q processors performs the following sequence of steps:

- it picks a coordinate k randomly and independently to update;
- it retrieves $\mathbf{x}^{q\ell}$ and then computes $\nabla_k f(\mathbf{x}^{q\ell})$;
- it increases the value of x_k in the main memory by $-\nabla_k f(\mathbf{x}^{q\ell})/\Gamma$, using an atomic addition operation.

Note that in a single phase a particular coordinate might be chosen two or more times. If coordinate k is chosen b times in the $(\ell+1)$-st phase, the value of x_k after the $(\ell+1)$-st phase is $x_k^{q\ell} - b \cdot \nabla_k f(\mathbf{x}^{q\ell})/\Gamma$.

3. The Result. Cheung, Cole and Tao [1] showed the following upper bound on the performance of asynchronous stochastic coordinate descent on strongly convex

functions, where $\mathbf{x}^*$ is a minimum point for the convex function.

THEOREM 3.1. *i. Suppose the asynchronous updating is run for exactly t updates. Also suppose that $q \leq \min\left\{\frac{\sqrt{n}}{270}, \frac{\Gamma\sqrt{n}}{270L_{\overline{\text{res}}}}\right\}$. If f is strongly convex with parameter μ, then*

$$\mathbb{E}\left[f(\mathbf{x}^t) - f(\mathbf{x}^*)\right] \leq \left(1 - \frac{1}{3n} \cdot \frac{\mu}{\Gamma}\right)^t \left(f(\mathbf{x}^0) - f(\mathbf{x}^*)\right).$$

ii. This result holds for all $q \leq \frac{\Gamma\sqrt{n}}{12L_{\text{res}}}$ if the asynchronous schedule obeys the Strong Common Value assumption.

The Strong Common Value assumption, specified in [1], captures a substantial class of asynchronous schedules including the consistent read constraint from [3], and all the schedules to which the analysis of Liu and Wright [2] apply, and more. The complementary construction in this paper observes the consistent reads condition, which is the most restrictive of these constraints. Consequently, our goal is to show that for $q = \Omega(\frac{\Gamma\sqrt{n}}{L_{\text{res}}})$ convergence with linear speed-up is not guaranteed. In fact, we will show that there is no convergence for most values of q obeying this constraint. To achieve this it suffices to show there is a single asynchronous schedule observing the consistent read condition for which there is no convergence.

First, in Appendix A, we will prove the following theorem, which shows that the sequential stochastic coordinate descent on our function f_ϵ when starting at the point $\mathbf{x}^0 = (-1, +1, -1, +1, \ldots)$ achieves a convergence rate that is at most a constant factor faster than the convergence rate given in Theorem 3.1. Recall that $\mu = 1 - \epsilon$ for f_ϵ.

PROPOSITION 3.2. *Let $\mathbf{x}^0$ be the point with even-indexed coordinates equal to $+1$ and odd-indexed coordinates equal to -1. Suppose the sequential stochastic coordinate descent is run for t steps on function f_ϵ starting at point x^0. Then*

$$\mathbb{E}\left[f_\epsilon(\mathbf{x}^t) - f_\epsilon(\mathbf{x}^*)\right] \geq \left(1 - \frac{2}{n} \cdot \frac{1-\epsilon}{\Gamma}\right)^t \left(f_\epsilon(\mathbf{x}^0) - f_\epsilon(\mathbf{x}^*)\right).$$

This result shows that for the function f_ϵ, the speedup guaranteed by Theorem 3.1 is indeed a linear speedup of the performance of the sequential SCD.

Now, we come to our main results.

Our analyses look at phases of q successive updates when the coordinate descent in Algorithm 4 is applied to the function $f \equiv L_{\max} \cdot f_\epsilon$ in (1.1) with $\epsilon = \sqrt{\frac{\left(\frac{L_{\text{res}}}{L_{\max}}\right)^2 - 1}{n-1}}$. The first result states that the expected value of $f(\mathbf{x}) - f^*$ grows by at least a factor of 4 from phase to phase.

THEOREM 3.3. *Suppose that $L_{\text{res}} \geq L_{\max}(1 + 8/n)$ and $q \geq \frac{4\Gamma\sqrt{n}}{\sqrt{L_{\text{res}}^2 - L_{\max}^2}}$. Then there is an asynchronous schedule for which the coordinate descent diverges when applied to the function f. Specifically, for every $t = rq$, with $r \geq 1$ an integer,*

$$\mathbb{E}\left[f(\mathbf{x}^t) - f^*\right] \;\geq\; 4^{(t/q)-1} \cdot (f(\mathbf{x}^0) - f^*).$$

If $L_{\text{res}} \geq 2L_{\max}$ (i.e. $\epsilon \geq \sqrt{3/(n-1)}$), the constraint on q becomes $q \geq \frac{8\Gamma\sqrt{n}}{\sqrt{3}L_{\text{res}}}$.

When $L_{\text{res}} \geq 2L_{\max}$, Theorem 3.3 states that once q exceeds the bound in Theorem 3.1(ii) by a constant factor, in order to have any possibility of convergence, let alone linear convergence, Γ has to increase at the same rate as q. Note that the upper bound on the rate of convergence decreases linearly with Γ, so this says that increasing q, the number of parallel updates, beyond this bound cannot increase the parallel runtime by more than a constant factor at best.

However, one might wonder if this divergence in expectation is a low probability event. Our next result shows that for most values of $q < n$ it is in fact a high probability event.

THEOREM 3.4. *Let $c_1, c_2 \geq 1$ be constants, let $c = c_1 + c_2$, and suppose that*

$$\frac{n}{e} \geq q \geq \max\left\{\frac{8\Gamma}{\epsilon L_{\max}}, \frac{10(c+1)}{\epsilon}\frac{\log n}{\log \frac{n}{q}}\right\}.$$

For each $c_1, c_2 \geq 1$, with probability at least $1 - 1/n^{c_2}$, there is an asynchronous schedule for which the coordinate descent diverges for at least $q \cdot n^{c_1}$ updates when applied to the function f. Specifically, for every $t = rq$ with $1 \leq r \leq n^{c_1}$ an integer,

$$f(\mathbf{x}^t) - f^* \;\geq\; 4^{(t/q)-1} \cdot (f(\mathbf{x}^0) - f^*).$$

If $L_{\text{res}} \geq 2L_{\max}$, the conditions $\frac{n}{e} \geq q \geq \max\left\{\frac{16\Gamma\sqrt{n}}{\sqrt{3}L_{\text{res}}}, \frac{200(c+1)L_{\max}\sqrt{n}}{\sqrt{3}L_{\text{res}}}\right\}$ and $n \geq \left(\frac{544(c+1)}{\sqrt{3}} \cdot \frac{L_{\max}}{L_{\text{res}}}\right)^{5/2}$ suffice.

Thus the comments in the paragraph after Theorem 3.3 apply here too, so long as the stated bounds on n hold.

Finally, one might wonder what happens to the value $f(\mathbf{x}) - f^*$ during a phase. Could it be small at any time? As it happens, with our schedule the value oscillates a lot, but the next and final result shows that with high probability it remains large at all times.

THEOREM 3.5. *Let $c_1, c_2 \geq 1$ be constants, let $c = c_1 + c_2$, and suppose that $L_{\text{res}} \geq 2L_{\max}$, $n \geq 400c^2(c+2)^2$, and*

$$\frac{n}{e} \geq q \geq \max\left\{\frac{8\Gamma\sqrt{n}}{\sqrt{3}L_{\text{res}}} + \frac{4c\Gamma}{L_{\max}}\;,\; 20c(c+2)\right\}.$$

For each $c_1, c_2 \geq 1$, with probability at least $1 - 1/n^{c_2}$, there is an asynchronous schedule for which the coordinate descent diverges for at least $q \cdot n^{c_1}$ updates when applied to the function f. Specifically, for every $1 \cdot n^{c_1}$,

$$f(\mathbf{x}^t) - f^* \;\geq\; \frac{1}{n^2} \cdot 4^{\lceil (t/q) \rceil - 1} \cdot (f(\mathbf{x}^0) - f^*).$$

Once more, the comments in the paragraph after Theorem 3.3 apply.

The three theorems incorporate general choices of Γ, c_1 and c_2. In Theorems 3.4 and 3.5, if we pick c_1, c_2 to be large enough constants (for example, both are 10 — which corresponds to a $1/n^{10}$ failure probability over the course of $q \cdot n^{10}$ updates), the conditions on n and q reduce to $n \geq \Theta(1)$ and $n/e \geq q \geq \Theta(\Gamma\sqrt{n}/L_{\text{res}})$, respectively.

4. Analysis. We consider running SACD with q processors on the function

$$f(\mathbf{x}) = L_{\max} \cdot \left[\frac{1-\epsilon}{2} \cdot \sum_{i=1}^{n} (x_i)^2 \; + \; \frac{\epsilon}{2} \cdot \left(\sum_{i=1}^{n} x_i \right)^2 \right],$$

for any ϵ satisfying $4/n \le \epsilon < 1$. We prove each theorem in turn.

We choose the initial point to be the all ones point, $\mathbf{x}^0 = (+1, +1, \ldots, +1)$. For $\ell \ge 0$, let $\mathbf{y}^\ell$ denote the up-to-date values of the coordinates immediately after the ℓ-th phase, i.e. $\mathbf{y}^\ell = \mathbf{x}^{q\ell}$. Also, let

$$G^\ell \triangleq \left| \sum_{k=1}^{n} y_k^\ell \right| \qquad \text{and} \qquad M^\ell \triangleq \max_{k=1,2,\cdots,n} |y_k^\ell|.$$

4.1. Proof of Theorem 3.3. The key claim, implying exponential growth in the value of $f(\mathbf{x})$ at the end of each successive phase of q updates, is given by the following lemma.

LEMMA 4.1. *If $\epsilon \ge \frac{4}{n}$ and $q \ge \frac{4\Gamma}{\epsilon L_{\max}}$ then for all ℓ, $\mathbb{E}\left[G^\ell\right] \ge 2^\ell G^0$.*

Proof. The proof is by induction. The claim clearly holds for $\ell = 0$.

Now suppose the result holds for some $\ell \ge 0$. Without loss of generality, we assume $\sum_{j=1}^{n} y_j^\ell > 0$; the other case will be symmetric. In the $(\ell+1)$-st phase, there are q updates. Each update picks a coordinate k; by Equation (2.1), the update reduces its value by

$$\frac{L_{\max}}{\Gamma} \cdot \left[(1-\epsilon) y_k^\ell + \epsilon \sum_{j=1}^{n} y_j^\ell \right] \; = \; \frac{L_{\max}}{\Gamma} \cdot \left[(1-\epsilon) y_k^\ell + \epsilon G^\ell \right].$$

The expected reduction due to q updates is at least

$$\begin{aligned} \frac{qL_{\max}}{\Gamma} \cdot \left[\epsilon G^\ell - \frac{1}{n}(1-\epsilon) G^\ell \right] &\ge \frac{3q\epsilon L_{\max}}{4\Gamma} \cdot G^\ell \quad (\text{as } n\epsilon \ge 4) \\ &\ge 3G^\ell \qquad\qquad (\text{as } q \ge \tfrac{4\Gamma}{\epsilon L_{\max}}). \end{aligned}$$

Thus, $\mathbb{E}\left[G^{\ell+1} | G^\ell\right] \ge \left| \mathbb{E}\left[\sum_{k=1}^{n} y_k^{\ell+1} | G^\ell \right] \right| \ge (3-1) G^\ell = 2G^\ell$; therefore $\mathbb{E}\left[G^{\ell+1}\right] \ge 2 \cdot \mathbb{E}\left[G^\ell\right]$, demonstrating the inductive claim. ∎

Proof of Theorem 3.3. We begin by showing that $f(\mathbf{y}^0) - f^* \le \epsilon \cdot (G^0)^2$, assuming that $\epsilon \ge 4/n$, which we justify in the next paragraph. To see this, note that $G^0 = n$ and $f(\mathbf{y}^0) - f^* = \frac{1}{2}(1-\epsilon)n + \frac{1}{2}\epsilon n^2$. As $\epsilon \ge 4/n$, we see that $f(\mathbf{y}^0) - f^* \le \frac{1}{8} \cdot \epsilon n^2 + \frac{1}{2}\epsilon n^2 \le \epsilon (G^0)^2$. Next, immediately after Phase ℓ, $f(\mathbf{y}^\ell) - f^* \ge \frac{\epsilon}{2} \cdot (G^\ell)^2$. Then, by the Cauchy-Schwarz inequality and Lemma 4.1, $\mathbb{E}\left[f(\mathbf{y}^\ell) - f^*\right] \ge \mathbb{E}\left[\frac{\epsilon}{2} \cdot (G^\ell)^2\right] \ge \frac{\epsilon}{2} \cdot \mathbb{E}\left[G^\ell\right]^2 \ge 4^{\ell-1}\epsilon \cdot (G^0)^2 \ge 4^{\ell-1}(f(\mathbf{y}^0) - f^*)$.

It remains to show that $\epsilon \ge 4/n$. Recall that $\epsilon^2 L_{\max}^2 = (L_{\text{res}}^2 - L_{\max}^2)/(n-1)$. Together, these imply $(L_{\text{res}}^2 - L_{\max}^2) \ge 16(n-1)/n^2 \cdot L_{\max}^2$. It suffices to have $L_{\text{res}} \ge L_{\max}(1 + 8/n)$, proving the theorem. ∎

4.2. Proof of Theorem 3.4. The idea of the construction is to show that if we can bound M^ℓ by $\frac{1}{4}\epsilon G^\ell$, then we can show the bound $G^\ell \ge 2^\ell G_0$ holds absolutely and not just in expectation. We will show that $M^\ell \le \frac{1}{4}\epsilon G^\ell$ with high probability.

Our construction uses a parameter b which we will specify later. We will need that in each phase, each coordinate is chosen at most b times. In Lemma 4.2 and Corollary 4.3 below, we show that this occurs with high probability when b is suitably large.

LEMMA 4.2. *In one phase, the probability that each coordinate is chosen at most b times by the q processors is at least $1 - n(eq/(b+1))^{b+1}/n^{b+1}$, where $e = 2.71828\ldots$.*

Proof. Let k be one of the coordinates, and let $\mathcal{E}(k)$ denote the event that coordinate k is chosen more than b times. Note that

$$\mathcal{E}(k) \;=\; \bigcup_{\substack{S:S\subset\{1,2,\cdots,q\}\\ |S|=b+1}} \{\text{coordinate } k \text{ is chosen by all the processors with labels in } S\}.$$

Thus, by the union bound, the probability that $\mathcal{E}(k)$ holds is at most $\binom{q}{b+1}\cdot\left(\frac{1}{n}\right)^{b+1}$, which is at most $(eq/(b+1))^{b+1}/n^{b+1}$ by a well-known formula for bounding binomial coefficients.

By the union bound again, the probability that $\cup_{k=1}^{n}\mathcal{E}(k)$ holds is at most $n(eq/(b+1))^{b+1}/n^{b+1}$. The lemma follows. ∎

COROLLARY 4.3. [Events $\mathcal{E}_1$ and $\mathcal{E}_1'$] *Suppose that $q \le \frac{n}{e}$, $c_1, c_2 \ge 1$, and $b \ge e-1$; also, let $\Lambda = \frac{\log n}{\log(n/q)}$. Then the probability that in each of the first n^{c_1} phases each coordinate is chosen at most b times by the q processors is at least $1 - n^{c_1+1-(b+1)/\Lambda}$. Furthermore,*
a. [Event $\mathcal{E}_1$] *if $b \ge (c_1+c_2+1)/\Lambda$, the probability is at least $1 - 1/n^{c_2}$; and*
b. [Event $\mathcal{E}_1'$] *if $n \ge 2$ and $b \ge (c_1+c_2+2)/\Lambda$, the probability is at least $1 - 1/n^{c_2+1} \ge 1 - 1/2n^{c_2}$.*

Next, we show the inductive bounds on M^ℓ and G^ℓ. Recall that $c = c_1 + c_2$.

LEMMA 4.4. *For any fixed $c_1, c_2 \ge 1$, conditioned on $\mathcal{E}_1$, if $\epsilon \ge \frac{4}{n}$ and $\frac{n}{e} \ge q \ge \max\left\{\frac{8\Gamma}{\epsilon L_{\max}}, \frac{10(c+1)}{\epsilon}\cdot\frac{\log n}{\log\frac{n}{q}}\right\}$, then for $\ell = 0, 1, 2, \ldots, n^{c_1}$, $G^\ell \ge 2^\ell G^0$ and $M^\ell \le \epsilon G^\ell/4$.*

Proof. Suppose the lemma holds for some $\ell \ge 0$, and without loss of generality, assume that $\sum_{j=1}^{n} y_j^\ell$ is positive; the other case will be symmetric. As in Lemma 4.1, consider an update in the $(\ell+1)$-st phase that picks coordinate k; the update reduces its value by

$$\frac{L_{\max}}{\Gamma}\cdot\left[(1-\epsilon)y_k^\ell + \epsilon G^\ell\right] \;\ge\; \frac{L_{\max}}{\Gamma}\cdot\left[-(1-\epsilon)M^\ell + \epsilon G^\ell\right] \;\ge\; \frac{3\epsilon L_{\max}}{4\Gamma}G^\ell.$$

Thus, immediately after the $(\ell+1)$-st phase, $\sum_{j=1}^{n} y_j^{\ell+1} \le \sum_{j=1}^{n} y_j^\ell - q\cdot\frac{3\epsilon L_{\max}}{4\Gamma}G^\ell = \left(1 - \frac{3\epsilon q L_{\max}}{4\Gamma}\right)G^\ell$. If $q \ge \frac{4\Gamma}{3\epsilon L_{\max}}$, then

$$G^{\ell+1} = \left|\sum_{j=1}^{n} y_j^{\ell+1}\right| \;\ge\; \left(\frac{3\epsilon q L_{\max}}{4\Gamma} - 1\right)G^\ell. \tag{4.1}$$

On the other hand, for each coordinate k, if it is chosen by b_k processors in the $(\ell+1)$-st phase, then the value of $y_k^{\ell+1}$ is

$$y_k^\ell - b_k\cdot\frac{L_{\max}}{\Gamma}\cdot\left[(1-\epsilon)y_k^\ell + \epsilon G^\ell\right].$$

For each q we consider, we apply Corollary 4.3(a). Note that $\frac{1}{\Lambda} = \frac{\log n}{\log \frac{n}{q}}$. Conditioned on $\mathcal{E}_1$, $b_k \le \frac{1}{\Lambda}(c+1)$. Also, since $|y_k^\ell| \le M^\ell \le \epsilon G^\ell/4$ and $L_{\max} \le \Gamma$,

$$|y_k^{\ell+1}| \le \frac{\epsilon}{4}\cdot G^\ell + \frac{1}{\Lambda}(c+1)\cdot\frac{L_{\max}}{\Gamma}\cdot\left(\frac{\epsilon}{4}\cdot G^\ell + \epsilon G^\ell\right)$$

$$= \left(\frac{1}{4} + \frac{5\cdot\frac{1}{\Lambda}(c+1)L_{\max}}{4\Gamma}\right)\epsilon G^\ell. \tag{4.2}$$

Given Equations (4.1) and (4.2), to guarantee that $G^{\ell+1} \ge 2G^\ell$, having $\frac{3\epsilon q L_{\max}}{4\Gamma} - 1 \ge 2$ suffices. We satisfy this by imposing $q \ge \frac{8\Gamma}{\epsilon L_{\max}}$, or equivalently $\frac{\epsilon q L_{\max}}{4\Gamma} \ge 2$, which is slightly stronger than needed, but will help improve the next constraint on q. To guarantee that $M^{\ell+1} \le \epsilon G^{\ell+1}/4$, the following condition suffices:

$$\frac{\epsilon}{4}\left[\frac{3\epsilon q L_{\max}}{4\Gamma} - 1\right] G^\ell \ge \left(\frac{1}{4} + \frac{5\cdot\frac{1}{\Lambda}(c+1)L_{\max}}{4\Gamma}\right)\epsilon G^\ell. \tag{4.3}$$

As $\frac{\epsilon q L_{\max}}{4\Gamma} \ge 2$, we see that $\frac{2\epsilon q L_{\max}}{4\Gamma} \ge \frac{5\cdot(c+1)L_{\max}}{\Lambda\Gamma}$ suffices; i.e. $q \ge \frac{10(c+1)}{\epsilon}\frac{\log n}{\log\frac{n}{q}}$ suffices. ∎

Proof of Theorem 3.4. This theorem follows from Lemma 4.4, which requires that Event $\mathcal{E}_1$ hold. So the following conditions suffice:

$$\epsilon \ge \frac{4}{n} \quad \text{and} \quad \frac{n}{e} \ge q \ge \max\left\{\frac{8\Gamma}{\epsilon L_{\max}}, \frac{10(c+1)}{\epsilon}\cdot\frac{\log n}{\log\frac{n}{q}}\right\}.$$

If $L_{\text{res}} \ge 2L_{\max}$ (implying $\frac{1}{\epsilon} \le \frac{2L_{\max}\sqrt{n}}{\sqrt{3}L_{\text{res}}}$), the final condition becomes

$$q \ge \max\left\{\frac{16\Gamma\sqrt{n}}{\sqrt{3}L_{\text{res}}}, \frac{20(c+1)\sqrt{n}L_{\max}}{\sqrt{3}L_{\text{res}}}\cdot\frac{\log n}{\log\frac{n}{q}}\right\}.$$

We focus on the second constraint, namely $q \ge \frac{20(c+1)\sqrt{n}L_{\max}}{\sqrt{3}L_{\text{res}}}\cdot\frac{\log n}{\log\frac{n}{q}}$. Let $\mathcal{C} = \frac{20(c+1)\sqrt{n}L_{\max}}{\sqrt{3}L_{\text{res}}}$. The constraint becomes $q \ge \mathcal{C}\frac{\log n}{\log\frac{n}{q}}$, or $q\log\frac{n}{q} \ge \mathcal{C}\log n$, where the RHS is independent of q. Note that $q\log\frac{n}{q}$ is an increasing function for $1 \le q \le \frac{n}{e}$. Therefore, if suffices to seek a $\hat{q} \ge 1$ such that $\hat{q}\log(n/\hat{q}) \ge \mathcal{C}\log n$; then $q \ge \hat{q}$ implies that the second constraint holds for $q \le n/e$.

Consider $\hat{q} = 10\mathcal{C}$. Then $\hat{q}\log(n/\hat{q}) \ge \mathcal{C}\log n$ is equivalent to the inequality $9\mathcal{C}\log n \ge 10\mathcal{C}\log(10\mathcal{C})$, and hence to $n^9 \ge (10\mathcal{C})^{10}$. Substituting for $\mathcal{C}$, we obtain

$$n^9 \ge \left(\frac{200(c+1)}{\sqrt{3}}\cdot\frac{L_{\max}}{L_{\text{res}}}\cdot\sqrt{n}\right)^{10}, \text{ or equivalently, } n \ge \left(\frac{200(c+1)}{\sqrt{3}}\cdot\frac{L_{\max}}{L_{\text{res}}}\right)^{5/2}.$$

We also need $\hat{q} = 10\mathcal{C} \le n/e$; $n \ge \left(\frac{544(c+1)}{\sqrt{3}}\cdot\frac{L_{\max}}{L_{\text{res}}}\right)^2$ suffices. ∎

4.3. Proof of Theorem 3.5.

LEMMA 4.5. *Conditioned on Event $\mathcal{E}_1'$ defined in Corollary 4.3, if $L_{\text{res}} \ge 2L_{\max}$, $n \ge 400c^2(c+2)^2$, and*

$$q \ge \max\left\{\frac{8\Gamma\sqrt{n}}{\sqrt{3}L_{\text{res}}} + \frac{4c\Gamma}{L_{\max}},\ 20c(c+2)\right\}$$

then $M^\ell \le \frac{1}{4c} G^\ell$ *and* $G^\ell \ge 2^\ell G^0$, *for* $1 \le \ell \le n^{c_1}$.

Proof. We follow the inductive argument in the proof of Lemma 4.4 closely. In the spirit of deriving (4.3), we apply Corollary 4.3(b) instead of Corollary 4.3(a), causing the $c+1$ term to be replaced by a $c+2$ term. Then, it suffices that

$$\frac{1}{4c}\left[\frac{3\epsilon q L_{\max}}{4\Gamma} - 1\right] G^\ell \ge \left(\frac{1}{4} + \frac{5\cdot(c+2)L_{\max}}{4\Gamma}\cdot\frac{\log n}{\log\frac{n}{q}}\right)\epsilon G^\ell.$$

The above constraint is equivalent to $q \ge \frac{4\Gamma}{3\epsilon L_{\max}} + \frac{4\Gamma c}{3L_{\max}} + \frac{20c(c+2)}{3}\cdot\frac{\log n}{\log\frac{n}{q}}$, so it suffices that $q \ge \max\left\{\frac{4\Gamma}{L_{\max}}\left(\frac{1}{\epsilon}+c\right), 10c(c+2)\cdot\frac{\log n}{\log\frac{n}{q}}\right\}$.

We focus on the second constraint. As argued in the proof of Theorem 3.4, it suffices to find a lower bound $\hat{q}$ on q, such that $\hat{q}\log\frac{n}{\hat{q}} \ge \mathcal{C}\log n$, where $\mathcal{C} \triangleq 10c(c+2)$. $\hat{q} = 2\mathcal{C}$ suffices if $n \ge (2\mathcal{C})^2$.

Since $q \ge \frac{4\Gamma}{\epsilon L_{\max}}$, following the derivation of (4.1) yields $G^{\ell+1} \ge 2G^\ell$. ∎

LEMMA 4.6. *[Event $\mathcal{E}_2$] Let* $\mathcal{I} = [-\frac{1}{2n}G^\ell, \frac{1}{2n}G^\ell]$. *Suppose all the conditions imposed in Lemma 4.5 hold. Then, with probability at least* $1 - 1/n^{c_2}$, *for* $0 \le \ell < n^{c_1}$, *at every time during phase* $\ell+1$, *at least one coordinate has a value outside* $\mathcal{I}$.

Proof. We condition on Event $\mathcal{E}_1'$, which by Corollary 4.3, occurs with probability at least $1 - 1/2n^{c_2}$.

As before, without loss of generality, we assume that $\sum_i y_i^\ell > 0$.

We start by showing that at the start of the $(\ell+1)$-st phase at least $2c$ coordinates have values larger than $\frac{1}{2n}G^\ell$. Note that $\sum_{i:y_i^\ell\in\mathcal{I}} y_i^\ell \le \frac{1}{2}G^\ell$. By Lemma 4.5, $M^\ell \le \frac{1}{4c}G^\ell$. Thus, at the start of Phase $\ell+1$, at least $\frac{1}{2}G^\ell / \frac{1}{4c}G^\ell = 2c$ coordinates have value greater than $\frac{1}{2n}G^\ell$, proving this claim.

Next, we observe that an update to a coordinate with value in I changes its value to be smaller than $-\frac{1}{2n}G^\ell$. For the largest value resulting from such an update is $\left[1-(1-\epsilon)\frac{L_{\max}}{\Gamma}\right]\frac{1}{2n}G^\ell - \frac{L_{\max}}{\Gamma}\epsilon G^\ell$. By assumption, $\frac{\Gamma}{L_{\max}} \le \frac{1}{4}\epsilon q \le \frac{1}{4}\epsilon n$; therefore $\frac{\epsilon L_{\max}}{\Gamma} \ge \frac{4}{n}$. We deduce the update value is at most $-\left(\frac{4}{n} - \frac{1}{2n}\right)G^\ell$, which demonstrates the claim.

In order for all the coordinates to end up in $\mathcal{I}$ during Phase $\ell+1$, we need that there be no coordinates less than $-\frac{1}{2n}G^\ell$ initially, and that there be no updates to the coordinates in $\mathcal{I}$ until all the other coordinates enter $\mathcal{I}$, assuming this is possible. This requires some $k \ge 2c$ updates to these at least $2c$ coordinates.

The probability that these updates happen first is at most

$$\frac{k!}{n^k} \le \frac{(2c)!}{n^{2c}} \le \left(\frac{2c}{n}\right)^{2c},$$

and this is bounded by $1/(2n)^c$ if $n \ge 8c^2$. Summed over all n^{c_1} phases, this gives a total failure probability of (significantly) less than $1/2n^{c_2}$.

Taking into account the $1/2n^{c_2}$ probability that Event $\mathcal{E}_1'$ does not occur, we see that the overall probability of Event $\mathcal{E}_2$ is at least $1 - 1/n^{c_2}$, as claimed. ∎

Proof of Theorem 3.5. Conditioned on Event $\mathcal{E}_2$, throughout phase $\ell+1$, at least one coordinate has a value outside the interval $\mathcal{I}$ defined in Lemma 4.6. Thus $f(\mathbf{x}) \ge \left[\frac{1}{2n}G^\ell\right]^2 \ge \frac{1}{4n^2}\cdot 4^\ell\left(G^0\right)^2 \ge \frac{1}{4}\cdot 4^\ell$.

Note that $f(\mathbf{x}^0) = \frac{1-\epsilon}{2} \cdot n + \frac{\epsilon}{2} n^2 \leq \epsilon n^2$ as, by assumption, $\epsilon > \frac{1}{n}$. We deduce that $f(\mathbf{x}) \geq \frac{1}{4n^2} \cdot 4^\ell \cdot f(\mathbf{x}^0)$ throughout this phase. ∎

5. Discussion. We have shown a tight asymptotic upper bound on the possible parallelism for achieving linear speedup when using asynchronous coordinate descent for almost the whole range of $\frac{L_{\overline{\text{res}}}}{L_{\max}}$. This upper bound holds even for composite functions. Furthermore, it holds even in the somewhat restrictive consistent read model, and thus it holds for the inconsistent read model too.

Acknowledgments. We thank the referees for their thoughtful and incisive comments.

Appendix A. Missing Proofs.

LEMMA A.1. *The strong convexity parameter of f_ϵ is $(1-\epsilon)$.*

Proof.

$$
\begin{aligned}
& f_\epsilon(y) - f_\epsilon(x) - \langle\, \nabla f_\epsilon(x)\ ,\ y - x\, \rangle \\
& \qquad \geq \frac{1-\epsilon}{2} \sum_i y_i^2 - x_i^2 + \frac{\epsilon}{2}\left[\big(\sum_i y_i\big)^2 - \big(\sum_i x_i\big)^2 \right] \\
& \qquad\qquad - (1-\epsilon) \sum_i (x_i y_i - x_i^2) - \epsilon \left[\big(\sum_i x_i\big)\big(\sum_i y_i - \sum_i x_i\big) \right] \\
& \qquad \geq \frac{1-\epsilon}{2} \sum_i (y_i^2 + x_i^2 - 2x_i y_i) + \frac{\epsilon}{2}\left[\big(\sum_i y_i\big)^2 + \big(\sum_i x_i\big)^2 - 2 \sum_i x_i \sum_i y_i \right] \\
& \qquad \geq \frac{1-\epsilon}{2} \sum_i \big(y_i - x_i\big)^2 + \frac{\epsilon}{2}\left[\sum_i y_i - \sum_i x_i \right]^2 \\
& \qquad \geq \frac{1-\epsilon}{2} ||y - x||^2.
\end{aligned}
$$

∎

Proof of Proposition 3.2. Recall that $f_\epsilon(\mathbf{x}^*) = 0$. Also, recall that the SCD process starts at the all ones point. Let C_1 denotes the even index coordinates, and C_{-1} those of odd index. Consider the following random variables:

$$
S_1(t) \ := \ \sum_{j \in C_1} x_j^t \qquad S_{-1}(t) \ := \ \sum_{j \in C_{-1}} (-x_j^t) \qquad S(t) \ := \ S_1(t) + S_{-1}(t).
$$

Note that $S_1(0) = S_{-1}(0) = n/2$ and $S(0) = n$.

Next, we derive a recurrence which, conditioned on $S(t)$, computes the expected value of $S(t+1) - S(t)$. Recall that if coordinate j is chosen to be updated at time $t+1$, then

$$
x_j^{t+1} - x_j^t \ = \ - \frac{\nabla_j f(\mathbf{x}^t)}{\Gamma} \ = \ - \frac{1-\epsilon}{\Gamma} \cdot x_j^t - \frac{\epsilon}{\Gamma} \cdot \left[S_1(t) - S_{-1}(t) \right].
$$

Thus,

$$\begin{aligned}\mathbb{E}\left[S(t+1)-S(t)\;\middle|\;S(t)\right] &= \frac{1}{n}\left[\sum_{j\in C_1}\left(-\frac{1-\epsilon}{\Gamma}\cdot x_j^t-\frac{\epsilon}{\Gamma}\cdot[S_1(t)-S_{-1}(t)]\right)\right.\\ &\qquad\left.+\sum_{j\in C_{-1}}\left(\frac{1-\epsilon}{\Gamma}\cdot x_j^t+\frac{\epsilon}{\Gamma}\cdot[S_1(t)-S_{-1}(t)]\right)\right]\\ &= \frac{1}{n}\left(-\frac{1-\epsilon}{\Gamma}\cdot S_1(t)-\frac{1-\epsilon}{\Gamma}\cdot S_{-1}(t)\right)\\ &= -\frac{1-\epsilon}{n\Gamma}S(t).\end{aligned}$$

The second equality above holds because $|C_1|=|C_{-1}|$, leading to cancellation of the terms $\pm\frac{\epsilon}{\Gamma}\cdot[S_1(t)-S_{-1}(t)]$. Thus, $\mathbb{E}\left[S(t+1)\mid S(t)\right]=S(t)\cdot\left(1-\frac{1-\epsilon}{n\Gamma}\right)$. Iterating this recurrence yields

$$\mathbb{E}\left[S(t)\right]=n\cdot\left(1-\frac{1-\epsilon}{n\Gamma}\right)^t. \tag{A.1}$$

Next, observe that for any fixed $S_1(t), S_{-1}(t)$, by the Power-Mean Inequality, $\sum_{j\in C_1}(x_j^t)^2\geq\frac{2|S_1(t)|^2}{n}$ and $\sum_{j\in C_{-1}}(x_j^t)^2\geq\frac{2|S_{-1}(t)|^2}{n}$. On the other hand, for any fixed $S(t)$, which equals ~~to~~ $S_1(t)+S_{-1}(t)$, the sum $\frac{|S_1(t)|^2}{n}+\frac{|S_{-1}(t)|^2}{n}$ is minimized when $S_1(t)=S_{-1}(t)=S(t)/2$. Thus,

$$\begin{aligned}\mathbb{E}\left[f_\epsilon(\mathbf{x}^t)\right] &\geq \mathbb{E}\left[\frac{1-\epsilon}{2}\sum_{j=1}^n(x_j^t)^2\right] \geq (1-\epsilon)\cdot\mathbb{E}\left[\frac{|S_1(t)|^2}{n}+\frac{|S_{-1}(t)|^2}{n}\right]\\ &\geq \frac{1-\epsilon}{2n}\cdot\mathbb{E}\left[S(t)^2\right].\end{aligned}$$

Finally, we complete the proof by using the Cauchy-Schwarz inequality and Equation (A.1):

$$\begin{aligned}\frac{1-\epsilon}{2n}\cdot\mathbb{E}\left[S(t)^2\right] \geq \frac{1-\epsilon}{2n}\cdot\mathbb{E}\left[S(t)\right]^2 &= \frac{(1-\epsilon)n}{2}\cdot\left(1-\frac{1-\epsilon}{n\Gamma}\right)^{2t}\\ &\geq \frac{(1-\epsilon)n}{2}\cdot\left(1-\frac{2(1-\epsilon)}{n\Gamma}\right)^{t}\\ &= f_\epsilon(\mathbf{x}^0)\cdot\left(1-\frac{2(1-\epsilon)}{n\Gamma}\right)^{t}.\end{aligned}$$

REFERENCES

[1] Y. K. Cheung, R. Cole and Y. Tao, *Fully asynchronous stochastic coordinate descent: A tight lower bound on the parallelism achieving linear speedup*, Math. Program. Series A, (to appear).

[2] J. Liu and S. J. Wright, *Asynchronous stochastic coordinate descent: Parallelism and convergence properties*, SIAM J. Optim., 25 (2015), pp. 351–376.

[3] J. Liu, S. J. Wright, C. Ré, V. Bittorf and S. Sridhar, *An asynchronous parallel stochastic coordinate descent algorithm*, J. Mach. Learn. Res., 16 (2015), pp. 285–322.

[4] H. Mania, X. Pan, D. S. Papailiopoulos, B. Recht, K. Ramchandran and M. I. Jordan, *Perturbed iterate analysis for asynchronous stochastic optimization*, SIAM J. Optim., 27 (2017), pp. 2202–2229.
[5] T. Sun, R. Hannah and W. Yin, *Asynchronous coordinate descent under more realistic assumptions*, Advances in Neural Information Processing Systems, (2017), pp. 6182–6190.