Accelerated, Parallel and Proximal Coordinate Descent

Olivier Fercoq * Peter Richtárik †

December 19, 2013

Abstract

We propose a new stochastic coordinate descent method for minimizing the sum of convex functions each of which depends on a small number of coordinates only. Our method (APPROX) is simultaneously Accelerated, Parallel and PROXimal; this is the first time such a method is proposed. In the special case when the number of processors is equal to the number of coordinates, the method converges at the rate $2\bar{\omega}\bar{L}R^2/(k+2)^2$, where k is the iteration counter, $\bar{\omega}$ is an *average* degree of separability of the loss function, $\bar{L}$ is the *average* of Lipschitz constants associated with the coordinates and individual functions in the sum, and R is the distance of the initial point from the minimizer. We show that the method can be implemented without the need to perform full-dimensional vector operations, which is considered to be the major bottleneck of accelerated coordinate descent. The fact that the method depends on the average degree of separability, and not on the maximum degree of separability, can be attributed to the use of new safe large stepsizes, leading to improved expected separable overapproximation (ESO). These are of independent interest and can be utilized in all existing parallel stochastic coordinate descent algorithms based on the concept of ESO.

1 Introduction

Developments in computing technology and ubiquity of digital devices resulted in an increased interest in solving optimization problems of extremely big sizes. Applications can be found in all areas of human endeavor where data is available, including the internet, machine learning, data science and scientific computing. The size of these problems is so large that it is necessary to decompose the problem into smaller, more manageable, pieces. Traditional approaches, where it is possible to rely on full-vector operations in the design of an iterative scheme, must be revisited.

Coordinate descent methods [12, 17] appear as a very popular class of algorithms for such problems as they can break down the problem into smaller pieces, and can take advantage of sparsity patterns in the data. With big data problems it is necessary to design algorithms able to utilize modern parallel computing architectures. This resulted in an interest in parallel [16, 21, 3, 15] and distributed [14] coordinate descent methods.

*School of Mathematics, The University of Edinburgh, United Kingdom (e-mail: olivier.fercoq@ed.ac.uk)

†School of Mathematics, The University of Edinburgh, United Kingdom (e-mail: peter.richtarik@ed.ac.uk)

The work of both authors was supported by the EPSRC grant EP/I017127/1 (Mathematics for Vast Digital Resources) and by the Centre for Numerical Algorithms and Intelligent Software (funded by EPSRC grant EP/G036136/1 and the Scottish Funding Council). The work of P.R. was also supported by EPSRC grant EP/K02325X/1 (Accelerated Coordinate Descent Methods for Big Data Problems) and by the Simons Institute for the Theory of Computing at UC Berkeley.

In this work we focus on the solution of convex optimization problems with a huge number of variables of the form

$$\min_{x \in \mathbf{R}^N} \quad f(x) + \psi(x). \tag{1}$$

Here $x = (x^{(1)}, \dots, x^{(n)}) \in \mathbf{R}^N$ is a decision vector composed of n blocks, with $x^{(i)} \in \mathbf{R}^{N_i}$,

$$f(x) = \textstyle\sum_{j=1}^{m} f_j(x), \tag{2}$$

where f_j are smooth convex functions, and ψ is a block separable regularizer (e.g., $L1$ norm).

In this work we make the following three main contributions:

1. We design and analyze the first stochastic coordinate descent method which is simultaneously *accelerated*, *parallel* and *proximal*. In fact, we are not aware of any published results on accelerated coordinate descent which would either be proximal *or* parallel.

   Our method is *accelerated* in the sense that it achieves an $O(1/k^2)$ convergence rate, where k is the iteration counter. The first *gradient* method with this convergence rate is due to Nesterov [10]; see also [23, 1]. Accelerated stochastic coordinate descent method, for convex minimization without constraints, was originally proposed in 2010 by Nesterov [12].

Paper	**Proximal**	**Parallel**	**Accelerated**	**Notable feature**
Leventhal & Lewis, 2008 [5]	×	×	×	quadratic f
S-Shwartz & Tewari, 2009 [18]	ℓ_1	×	×	1st ℓ_1-regularized
Nesterov, 2010 [12]	×	×	YES	1st block, 1st accelerated
Richtárik & Takáč, 2011 [17]	YES	×	×	1st general proximal
Bradley et al, 2012 [2]	ℓ_1	YES	×	ℓ_1-regularized parallel
Richtárik & Takáč, 2012 [16]	YES	YES	×	1st general parallel
S-Shwartz & Zhang, 2012 [19]	YES	×	×	1st primal-dual
Necoara et al, 2012 [9]	×	×	×	2-coordinate descent
Takáč et al, 2013 [21]	×	YES	×	1st primal-d. & parallel
Tappenden et al, 2013 [22]	YES	×	×	1st inexact
Necoara & Clipici, 2013 [8]	YES	×	×	coupled constraints
Lin & Xiao, 2013 [25]	×	×	YES	improvements
Fercoq & Richtárik, 2013 [3]	YES	YES	×	1st nonsmooth f
Lee & Sidford, 2013 [4]	×	×	YES	1st efficient accelerated
Richtárik & Takáč, 2013 [14]	YES	YES	×	1st distributed
Liu et al, 2013 [6]	×	YES	×	asynchronous
Richtárik & Takáč, 2013 [15]	×	YES	×	1st parallel nonuniform
This paper	**YES**	**YES**	**YES**	**3 × YES**

Table 1: Selected recent papers analyzing the iteration complexity of *stochastic* coordinate descent methods. Our algorithm is simultaneously proximal, parallel and accelerated. In the last column we highlight a single notable feature, necessarily chosen subjectively, of each work.

   Various variants of proximal and parallel (but non-accelerated) stochastic coordinate descent methods were proposed [2, 16, 3, 14]. In Table 1 we provide a list[1] of some recent research papers proposing and analyzing *stochastic* coordinate descent methods. The table substantiates our observation that while the proximal setting is standard in the literature, parallel

[1]This list is necessarily incomplete, it was not our goal to be comprehensive. For a somewhat more substantial review of these and other works we refer the reader to [16, 3].

methods are much less studied, and finally, there is just a handful of papers dealing with accelerated variants.

2. We propose *new stepsizes* for parallel coordinate descent methods, based on a new expected separable overapproximation (ESO). These stepsizes can for some classes of problems (e.g., f_j=quadratics), be much larger than the stepsizes proposed proposed for the (non-accelerated) parallel coordinate descent method (PCDM) in [16]. Let ω_j be the number of of blocks function f_j depends on. The stepsizes, and hence the resulting complexity, of PCDM, depend on the quantity $\omega = \max_j \omega_j$. However, our stepsizes take all the values ω_j into consideration and the result of this is complexity that depends on a data-weighted average $\bar{\omega}$ of the values ω_j. Since $\bar{\omega}$ can be much smaller than ω, our stepsizes result in dramatic acceleration for our method and other methods whose analysis is based on an ESO [16, 3, 14].

3. We identify a large subclass of problems of the form (1) for which the *full-vector operations* inherent in accelerated methods *can be eliminated.* This contrasts with Nesterov's accelerated coordinate descent scheme [12], which is impractical due to this bottleneck. Having established his convergence result, Nesterov remarked [12] that:

   > "However, for some applications [...] the complexity of one iteration of the accelerated scheme is rather high since for computing y_k it needs to operate with full-dimensional vectors."

   Subsequently, in part due to these issues, the work of the community focused on simple methods as opposed to accelerated variants. For instance, Richtárik & Takáč [17] use Nesterov's observation to justify their focus on non-accelerated methods in their work on coordinate descent methods in the proximal/composite setting.

   Recently, Lee & Sidford [4] were able to avoid full dimensional operations in the case of minimizing a convex quadratic without constraints, by a careful modification of Nesterov's method. This was achieved by introducing an extra sequence of iterates and observing that for quadratic functions it is possible to compute partial derivative of f evaluated at a linear combination of full dimensional vectors without ever forming the combination. We extend the ideas of Lee & Sidford [4] to our general setting 1 in the case when $f_j(x) = \phi_j(a_j^T x)$, where ϕ_j are scalar convex functions with Lipschitz derivative and the vectors a_j are block-sparse.

Contents. The rest of the paper is organized as follows. We start by describing new stepsizes for parallel coordinate descent methods, based on novel assumptions, and compare them with existing stepsizes (Section 2). We then describe our algorithm and state and comment on the main complexity result (Section 3). Subsequently, we give a proof of the result (Section 4). We then describe an efficient implementation of our method, one that does not require the computation of full-vector operations (Section 5), and finally comment on our numerical experiments (Section 6).

Notation. It will be convenient to define natural operators acting between the spaces $\mathbf{R}^N$ and $\mathbf{R}^{N_i}$. In particular, we will often wish to lift a block $x^{(i)}$ from $\mathbf{R}^{N_i}$ to $\mathbf{R}^N$, filling the coordinates corresponding to the remaining blocks with zeros. Likewise, we will project $x \in \mathbf{R}^N$ back into $\mathbf{R}^{N_i}$. We will now formalize these operations.

Let U be the $N \times N$ identity matrix, and let $U = [U_1, U_2, \ldots, U_n]$ be its decomposition into column submatrices $U_i \in \mathbf{R}^{N \times N_i}$. For $x \in \mathbf{R}^N$, let $x^{(i)}$ be the block of variables corresponding to

the columns of U_i, that is, $x^{(i)} = U_i^T x \in \mathbf{R}^{N_i}$, $i = 1, 2, \ldots, n$. Any vector $x \in \mathbf{R}^N$ can be written, uniquely, as $x = \sum_{i=1}^n U_i x^{(i)}$. For $h \in \mathbf{R}^N$ and $\emptyset \neq S \subseteq [n] \stackrel{\text{def}}{=} \{1, 2, \ldots, n\}$, we write

$$h_{[S]} = \sum_{i \in S} U_i h^{(i)}. \tag{3}$$

In words, $h_{[S]}$ is a vector in $\mathbf{R}^N$ obtained from $h \in \mathbf{R}^N$ by zeroing out the blocks that do not belong to S. For convenience, we will also write

$$\nabla_i f(x) \stackrel{\text{def}}{=} (\nabla f(x))^{(i)} = U_i^T \nabla f(x) \in \mathbf{R}^{N_i} \tag{4}$$

for the vector of partial derivatives of f corresponding to coordinates belonging to block i.

With each block $i \in [n]$ we associate a positive definite matrix $B_i \in \mathbf{R}^{N_i \times N_i}$ and a scalar $v_i > 0$, and equip $\mathbf{R}^{N_i}$ and $\mathbf{R}^N$ with the norms

$$\|x^{(i)}\|_{(i)} \stackrel{\text{def}}{=} \langle B_i x^{(i)}, x^{(i)} \rangle^{1/2}, \qquad \|x\|_v \stackrel{\text{def}}{=} \left(\sum_{i=1}^n v_i \|x^{(i)}\|_{(i)}^2 \right)^{1/2}. \tag{5}$$

The corresponding conjugate norms, defined by $\|s\|^* = \max\{\langle s, x \rangle : \|x\| \leq 1\}$, are given by

$$\|x^{(i)}\|_{(i)}^* \stackrel{\text{def}}{=} \langle B_i^{-1} x^{(i)}, x^{(i)} \rangle^{1/2}, \qquad \|x\|_v^* = \left(\sum_{i=1}^n v_i^{-1} \left(\|x^{(i)}\|_{(i)}^* \right)^2 \right)^{1/2}. \tag{6}$$

We also write $\|v\|_1 = \sum_i |v_i|$.

2 Stepsizes for parallel coordinate descent methods

The framework for designing and analyzing (non-accelerated) parallel coordinate descent methods, developed by Richtárik & Takáč [16], is based on the notions of *block sampling* and *expected separable overapproximation (ESO)*. We now briefly review this framework as our accelerated method is cast in it, too. Informally, a block sampling is the random law describing the *selection of blocks* at each iteration. An ESO is an inequality, involving f and $\hat{S}$, which is used to *compute updates* to selected blocks. The complexity analysis in our paper is based on the following generic assumption.

Assumption 1 (Expected Separable Overapproximation [16, 3])**.** *We assume that:*

1. *f is convex and differentiable.*
2. *$\hat{S}$ is a uniform block sampling. That is, $\hat{S}$ is a random subset of $[n] = \{1, 2, \ldots, n\}$ with the property*[2] *that $\mathbf{P}(i \in \hat{S}) = \mathbf{P}(j \in \hat{S})$ for all $i, j \in [n]$. Let $\tau = \mathbf{E}[|\hat{S}|]$.*
3. *There are computable constants $v = (v_1, \ldots, v_n) > 0$ for which the pair $(f, \hat{S})$ admits the Expected Separable Overapproximation (ESO):*

$$\mathbf{E}\left[f(x + h_{[\hat{S}]}) \right] \leq f(x) + \frac{\tau}{n} \left(\langle \nabla f(x), h \rangle + \frac{1}{2} \|h\|_v^2 \right), \qquad x, h \in \mathbf{R}^N. \tag{7}$$

[2]It is easy to see that if $\hat{S}$ is a uniform sampling, then necessarily, $\mathbf{P}(i \in \hat{S}) = \frac{\mathbf{E}|\hat{S}|}{n}$ for all $i \in [n]$.

If the above inequality holds, for simplicity we will write[3] $(f, \hat{S}) \sim \mathrm{ESO}(v)$.

In the context of parallel coordinate descent methods, uniform block samplings and inequalities (7) involving such samplings were introduced and systematically studied by Richtárik & Takáč [16]. An ESO inequality for a uniform *distributed* sampling was developed in [14] and that *nonuniform* samplings and ESO, together with a parallel coordinate descent method based on such samplings, was proposed in [15].

Fercoq & Richtárik [3, Theorem 10] observed that inequality (7) is equivalent to requiring that the gradients of the functions

$$\hat{f}_x : h \mapsto \mathbf{E}\left[f(x + h_{[\hat{S}]})\right], \qquad x \in \mathbf{R}^N,$$

be Lipschitz at $h = 0$, uniformly in x, with constant τ/n, with respect to the norm $\|\cdot\|_v$. Equivalently, the Lipschitz constant is $L^{\hat{f}}$ with respect to the norm $\|\cdot\|_{\tilde{v}}$, where

$$L^{\hat{f}} = \frac{\tau \|v\|_1}{n^2}, \qquad \tilde{v} \stackrel{\text{def}}{=} n\frac{v}{\|v\|_1}.$$

The change of norms is done so as to enforce that the weights in the norm sum to n, which means that different ESOs can be compared using the constants $L^{\hat{f}}$. The above observations are useful in understanding what the ESO inequality encodes: By moving from x to

$$x_+ = x + h_{[\hat{S}]},$$

one is taking a step in a random subspace of $\mathbf{R}^N$ spanned by the blocks belonging to $\hat{S}$. If $\tau \ll n$, which is often the case in big data problems[4], the step is confined to a *low-dimensional* subspace of $\mathbf{R}^N$. It turns out that for many classes of functions arising in applications, for instance for functions exhibiting certain sparsity or partial separability patterns, it is the case that the gradient of f varies much more slowly in such subspaces, on average, than it does in $\mathbf{R}^N$. This in turn would imply that updates h based on minimizing the right hand side of (7) would produce larger steps, and eventually lead to faster convergence.

2.1 New model

Consider f of the form (2), i.e.,

$$f(x) = \sum_{j=1}^{m} f_j(x),$$

where f_j depends on blocks $i \in C_j$ only. Let $\omega_j = |C_j|$, and $\omega = \max_j \omega_j$.

Assumption 2. *The functions $\{f_j\}$ have block-Lipschitz gradient with constants $L_{ji} \geq 0$. That is, for all $j = 1, 2, \ldots, m$ and $i = 1, 2, \ldots, n$,*

$$\|\nabla_i f_j(x + U_i t) - \nabla_i f_j(x)\|^*_{(i)} \leq L_{ji}\|t\|_{(i)}, \qquad x \in \mathbf{R}^N,\ t \in \mathbf{R}^{N_i}. \tag{8}$$

[3] In [16], the authors write $\frac{\beta}{2}\|h\|^2_w$ instead of $\frac{1}{2}\|h\|^2_v$. This is because they study families of samplings $\hat{S}$, parameterized by τ, for which w fixed and all changes can thus be captured in the constant β. Clearly, the two definition as interchangeable as one can choose $v = \beta w$. Here we will need to compare weights which are not linearly dependent, hence the simplified notation.

[4] In fact, one may define a "big data" problem by requiring that the number of parallel processors τ available for optimization is much smaller than the dimension n of the problem.

Note that, necessarily,

$$L_{ji} = 0 \qquad \text{whenever} \qquad i \notin C_j. \tag{9}$$

Assumption 2 is *stronger* than the assumption considered in [16]. Indeed, in [16] the authors only assumed that the *sum* f, as opposed to the individual functions f_j, has a block-Lipschitz gradient, with constants $L_1, \dots, L_n$. That is,

$$\|\nabla_i f(x + U_i t) - \nabla_i f(x)\|^*_{(i)} \leq L_i \|t\|_{(i)}.$$

It is easy to see that if the stronger condition is satisfied, then the weaker one is also satisfied with L_i no worse than $L_i \leq \sum_{j=1}^m L_{ji}$.

2.2 New ESO

We now derive an ESO inequality for functions satisfying Assumption 2 and τ-nice sampling $\hat{S}$. That is, $\hat{S}$ is a random subset of $[n]$ of cardinality τ, chosen uniformly at random. One can derive similar bounds for all uniform samplings considered in [16] using the same approach.

Theorem 1. *Let f satisfy Assumption 2.*

(i) If $\hat{S}$ is a τ-nice sampling, then for all $x, h \in \mathbf{R}^N$,

$$\mathbf{E}\left[f(x + h_{[\hat{S}]})\right] \leq f(x) + \frac{\tau}{n}\left(\langle \nabla f(x), h\rangle + \frac{1}{2}\|h\|_v^2\right), \tag{10}$$

where

$$v_i \overset{def}{=} \sum_{j=1}^m \beta_j L_{ji} = \sum_{j: i \in C_j} \beta_j L_{ji}, \qquad i = 1, 2, \dots, n, \tag{11}$$

$$\beta_j \overset{def}{=} 1 + \frac{(\omega_j - 1)(\tau - 1)}{\max\{1, n-1\}}, \qquad j = 1, 2, \dots, m.$$

That is, $(f, \hat{S}) \sim ESO(v)$.

(ii) Moreover, for all $x, h \in \mathbf{R}^N$ we have

$$f(x + h) \leq f(x) + \langle \nabla f(x), h\rangle + \frac{\bar{\omega}\bar{L}}{2}\|h\|_w^2, \tag{12}$$

where

$$\bar{\omega} \overset{def}{=} \sum_j \omega_j \frac{\sum_i L_{ji}}{\sum_{k,i} L_{ki}}, \qquad \bar{L} \overset{def}{=} \frac{\sum_{ji} L_{ji}}{n}, \qquad w_i \overset{def}{=} \frac{n}{\sum_{j,i} \omega_j L_{ji}} \sum_j \omega_j L_{ji}. \tag{13}$$

Note that $\bar{\omega}$ is a data-weighted average *of the values $\{\omega_j\}$ and that $\sum w_i = n$.*

Proof. Statement (ii) is a special case of (i) for $\tau = n$ (notice that $\bar{\omega}\bar{L}w = v$). We hence only need to prove (i). A well known consequence of (8) is

$$f_j(x + U_i t) \leq f_j(x) + \langle \nabla_i f_j(x), t\rangle + \frac{L_{ji}}{2}\|t\|^2_{(i)}, \qquad x \in \mathbf{R}^N,\ t \in \mathbf{R}^{N_i}. \tag{14}$$

We first claim that for all i and j,

$$\mathbf{E}\left[f_j(x+h_{[\hat{S}]})\right] \leq f_j(x) + \frac{\tau}{n}\left(\langle \nabla f_j(x), h\rangle + \frac{\beta_j}{2}\|h\|^2_{L_{j:}}\right), \tag{15}$$

where $L_{j:} = (L_{j1}, \ldots, L_{jn}) \in \mathbf{R}^n$. That is, $(f_j, \hat{S}) \sim ESO(\beta_j L_{j:})$. Equation (10) then follows by adding up[5] the inequalities (15) for all j. Let us now prove the claim.[6] We fix x and define

$$\hat{f}_j(h) \stackrel{\text{def}}{=} f_j(x+h) - f_j(x) - \langle \nabla f_j(x), h\rangle. \tag{16}$$

Since

$$\begin{aligned}\mathbf{E}\left[\hat{f}_j(h_{[\hat{S}]})\right] &\stackrel{(16)}{=} \mathbf{E}\left[f_j(x+h_{[\hat{S}]}) - f_j(x) - \langle \nabla f_j(x), h_{[\hat{S}]}\rangle\right] \\ &\stackrel{(42)}{=} \mathbf{E}\left[f_j(x+h_{[\hat{S}]})\right] - f_j(x) - \tfrac{\tau}{n}\langle \nabla f_j(x), h\rangle,\end{aligned}$$

it now only remains to show that

$$\mathbf{E}\left[\hat{f}_j(h_{[\hat{S}]})\right] \leq \tfrac{\tau\beta_j}{2n}\|h\|^2_{L_{j:}}. \tag{17}$$

We now adopt the convention that expectation conditional on an event which happens with probability 0 is equal to 0. Let $\eta_j \stackrel{\text{def}}{=} |C_j \cap \hat{S}|$, and using this convention, we can write

$$\mathbf{E}\left[\hat{f}_j(h_{[\hat{S}]})\right] = \sum_{k=0}^{n} \mathbf{P}(\eta_j = k)\mathbf{E}\left[\hat{f}_j(h_{[\hat{S}]}) \mid \eta_j = k\right]. \tag{18}$$

For any $k \geq 1$ for which $\mathbf{P}(\eta_j = k) > 0$, we now use use convexity of $\hat{f}_j$ to write

$$\begin{aligned}\mathbf{E}\left[\hat{f}_j(h_{[\hat{S}]}) \mid \eta_j = k\right] &= \mathbf{E}\left[\hat{f}_j\left(\tfrac{1}{k}\sum_{i\in C_j\cap\hat{S}} kU_i h^{(i)}\right) \mid \eta_j = k\right] \\ &\leq \mathbf{E}\left[\tfrac{1}{k}\sum_{i\in C_j\cap\hat{S}} \hat{f}_j\left(kU_i h^{(i)}\right) \mid \eta_j = k\right] \\ &= \tfrac{1}{\omega_j}\sum_{i\in C_j} \hat{f}_j\left(kU_i h^{(i)}\right) \\ &\stackrel{(14)+(16)}{\leq} \tfrac{1}{\omega_j}\sum_{i\in C_j} \tfrac{L_{ji}}{2}\|kh^{(i)}\|^2_{(i)} = \tfrac{k^2}{2\omega_j}\|h\|^2_{L_{j:}}\end{aligned} \tag{19}$$

where the second equality follows from Equation (41) in [16]. Finally,

$$\mathbf{E}\left[\hat{f}_j(h_{[\hat{S}]})\right] \stackrel{(19)+(18)}{\leq} \sum_k \mathbf{P}(\eta_j = k)\tfrac{k^2}{2\omega_j}\|h\|^2_{L_{j:}} = \tfrac{1}{2\omega_j}\|h\|^2_{L_{j:}}\mathbf{E}[|C_j\cap\hat{S}|^2] = \tfrac{\tau\beta_j}{2n}\|h\|^2_{L_{j:}}, \tag{20}$$

where the last identity is Equation (40) in [16], and hence (17) is established. □

[5] At this step we could have also simply applied Theorem 10 from [16], which give the formula for an ESO for a conic combination of functions given ESOs for the individual functions. The proof, however, also amounts to simply adding up the inequalities.

[6] This claim is a special case of Theorem 14 in [16] which gives an ESO bound for a *sum* of functions f_j (here we only have a single function). We include the proof as in this special case it more straightforward.

2.3 Computation of L_{ji}

We now give a formula for the constants L_{ji} in the case when f_j arises as a composition of a scalar function ϕ_j whose derivative has a known Lipschitz constant (this is often easy to compute), and a linear functional. Let A be an $m \times N$ real matrix and for $j \in \{1, 2, \dots, m\}$ and $i \in [n]$ define

$$A_{ji} \stackrel{\text{def}}{=} e_j^T A U_i \in \mathbf{R}^{1 \times N_i}. \tag{21}$$

That is, A_{ji} is a row vector composed of the elements of row j of A corresponding to block i.

Theorem 2. *Let $f_j(x) = \phi_j(e_j^T A x)$, where $\phi_j : \mathbf{R} \to \mathbf{R}$ is a function with L_{ϕ_j}-Lipschitz derivative:*

$$|\phi_j(s) - \phi_j(s')| \leq L_{\phi_j}|s - s'|, \qquad s, s' \in \mathbf{R}. \tag{22}$$

Then f_j has a block Lipshitz gradient with constants

$$L_{ji} = L_{\phi_j}\left(\|A_{ji}^T\|_{(i)}^*\right)^2, \qquad i = 1, 2, \dots, n. \tag{23}$$

In other words, f_j satisfies (8) with constants L_{ji} given above.

Proof. For any $x \in \mathbf{R}^N$, $t \in \mathbf{R}^{N_i}$ and i we have

$$\begin{aligned}
\|\nabla_i f_j(x + U_i t) - \nabla_i f_j(x)\|_{(i)}^* &\stackrel{(4)}{=} \|U_i^T (e_j^T A)^T \phi_j'(e_j^T A(x + U_i t)) - U_i^T (e_j^T A)^T \phi_j'(e_j^T A x)\|_{(i)}^* \\
&\stackrel{(21)}{=} \|A_{ji}^T \phi_j'(e_j^T A(x + U_i t)) - A_{ji}^T \phi_j'(e_j^T A x)\|_{(i)}^* \\
&\leq \|A_{ji}^T\|_{(i)}^* |\phi_j'(e_j^T A(x + U_i t)) - \phi_j'(e_j^T A x)| \\
&\stackrel{(22)+(21)}{\leq} \|A_{ji}^T\|_{(i)}^* L_{\phi_j} |A_{ji} t| \;\leq\; \|A_{ji}^T\|_{(i)}^* L_{\phi_j} \|A_{ji}^T\|_{(i)}^* \|t\|_{(i)},
\end{aligned}$$

where the last step follows by applying the Cauchy-Schwartz inequality. □

Example 1 (Quadratics)**.** Consider the quadratic function

$$f(x) = \tfrac{1}{2}\|Ax - b\|^2 = \tfrac{1}{2}\sum_{j=1}^m (e_j^T A x - b_j)^2.$$

Then $f_j(x) = \phi_j(e_j^T A x)$, where $\phi_j(s) = \frac{1}{2}(s - b_j)^2$ and $L_{\phi_j} = 1$.

(i) Consider the block setup with $N_i = 1$ (all blocks are of size 1) and $B_i = 1$ for all $i \in [n]$. Then $L_{ji} = A_{ji}^2$. In Table 3 we list stepsizes for coordinate descent methods proposed in the literature. It can be seen that our stepsizes are better than those proposed by Richtárik & Takáč [16] and those proposed by Necoara & Clipici [7]. Indeed, $v_i^{\text{rt}} \geq v_i^{\text{fr}}$ for all i. The difference grows as τ grows; and there is equality for $\tau = 1$. We also have $\|v^{\text{nc}}\|_1 \geq \|v^{\text{fr}}\|_1$, but here the difference decreases with τ; and there is equality for $\tau = n$.

(ii) Choose nontrivial block sizes and define data-driven block norms with $B_i = A_i^T A_i$, where $A_i = AU_i$, assuming that the matrices $A_i^T A_i$ are positive definite. Then

$$L_{ji} = L_{\phi_j}(\|A_{ji}^T\|_{(i)}^*)^2 \stackrel{(6)}{=} \langle (A_i^T A_i)^{-1} A_{ji}^T, A_{ji}^T \rangle \stackrel{(21)}{=} e_j^T A_i (A_i^T A_i)^{-1} A_i^T e_j.$$

Table 2 lists constants L_ϕ for selected scalar loss functions ϕ popular in machine learning.

Loss	$\phi(s)$	L_ϕ
Square Loss	$\frac{1}{2}s^2$	1
Logistic Loss	$\log(1+e^s)$	1

Table 2: Lipschitz constants of the derivative of selected scalar loss functions.

Paper	v_i
Richtárik & Takáč [16]	$v_i^{\rm rt} = \sum_{j=1}^m \left(1 + \frac{(\omega-1)(\tau-1)}{\max\{1,n-1\}}\right) A_{ji}^2$
Necoara & Clipici [7]	$v_i^{\rm nc} = \sum_{j:i\in C_j} \sum_{k=1}^n A_{jk}^2$
This paper	$v_i^{\rm fr} = \sum_{j=1}^m \left(1 + \frac{(\omega_j-1)(\tau-1)}{\max\{1,n-1\}}\right) A_{ji}^2$

Table 3: ESO stepsizes for coordinate descent methods suggested in the literature in the case of a quadratic $f(x) = \frac{1}{2}\|Ax-b\|^2$. We consider setup with elementary block sizes ($N_i = 1$) and $B_i = 1$.

3 Accelerated parallel coordinate descent

We are interested in solving the regularized optimization problem

$$\begin{array}{rl} \text{minimize} & F(x) \stackrel{\rm def}{=} f(x) + \psi(x), \\ \text{subject to} & x = (x^{(1)}, \dots, x^{(n)}) \in \mathbf{R}^{N_1} \times \cdots \times \mathbf{R}^{N_n} = \mathbf{R}^N, \end{array} \tag{24}$$

where $\psi : \mathbf{R}^N \to \mathbf{R} \cup \{+\infty\}$ is a (possibly nonsmooth) convex regularizer that is separable in the blocks $x^{(i)}$:

$$\psi(x) = \sum_{i=1}^n \psi_i(x^{(i)}). \tag{25}$$

3.1 The algorithm

We now describe our method (Algorithm 1). It is presented here in a form that facilitates analysis and comparison with existing methods. In Section 5 we rewrite the method into a different (equivalent) form – one that is geared towards practical efficiency.

The method starts from $x_0 \in \mathbf{R}^N$ and generates three vector sequences, $\{x_k, y_k, z_k\}_{k\geq 0}$. In Step 3, y_k is defined as a convex combination of x_k and z_k, which may in general be full dimensional vectors. This is not efficient; but we will ignore this issue for now. In Section 5 we show that it is possible to implement the method in such a way that it not necessary to ever form y_k. In Step 4 we generate a random block sampling S_k and then perform steps 5–9 in parallel. The assignment

Algorithm 1 APPROX: **A**ccelerated **P**arallel **Prox**imal Coordinate Descent Method

1: Choose $x_0 \in \mathbf{R}^N$ and set $z_0 = x_0$ and $\theta_0 = \frac{\tau}{n}$
2: **for** $k \geq 0$ **do**
3: $y_k = (1-\theta_k)x_k + \theta_k z_k$
4: Generate a random set of coordinates $S_k \sim \hat{S}$
5: $z_{k+1} = z_k$
6: **for** $i \in S_k$ **do**
7: $z_{k+1}^{(i)} = \arg\min_{z \in \mathbf{R}^{N_i}} \left\{ \langle \nabla_i f(y_k), z - y_k^{(i)} \rangle + \frac{n\theta_k v_i}{2\tau} \|z - z_k^{(i)}\|_{(i)}^2 + \psi_i(z) \right\}$
8: **end for**
9: $x_{k+1} = y_k + \frac{n}{\tau}\theta_k(z_{k+1} - z_k)$
10: $\theta_{k+1} = \frac{\sqrt{\theta_k^4 + 4\theta_k^2} - \theta_k^2}{2}$
11: **end for**

$z_{k+1} \leftarrow z_k$ is not necessary in practice; the vector z_k should be overwritten in place. Instead, Steps 5–8 should be seen as saying that we update blocks $i \in S_k$ of z_k, by solving $|S_k|$ proximal problems in parallel, and call the resulting vector z_{k+1}. Note in Step 9, x_{k+1} should also be computed in parallel. Indeed, x_{k+1} is obtained from y_k by changing the blocks of y_k that belong to S_k - this is because z_{k+1} and z_k differ in those blocks only. Note that gradients are evaluated only at y_k. We show in Section 5 how this can be done efficiently, for some problems, without the need to form y_k.

We now formulate the main result of this paper.

Theorem 3. *Let Assumption 1 hold, with $(f, \hat{S}) \sim \mathrm{ESO}(v)$, where $\tau = \mathbf{E}[|\hat{S}|] > 0$. Let $x_0 \in \operatorname{dom}\psi$, and assume that the random sets S_k in Algorithm 1 are chosen independently, following the distribution of $\hat{S}$. Then for any optimal point x_* of problem (24), the iterates $\{x_k\}_{k\geq 0}$ of Algorithm 1 satisfy*

$$\mathbf{E}[F(x_k) - F(x_*)] \quad \leq \quad \frac{4n^2}{(k\tau + 2n)^2} C, \tag{26}$$

where

$$C \quad \stackrel{def}{=} \quad \left(1 - \frac{\tau}{n}\right)(F(x_0) - F(x_*)) + \frac{1}{2}\|x_0 - x_*\|_v^2. \tag{27}$$

In other words, for any $0 < \epsilon \leq C$, the number of iterations for obtaining an ϵ-solution in expectation does not exceed

$$k \quad = \quad \left\lceil \frac{2n}{\tau}\left(\sqrt{\frac{C}{\epsilon}} - 1\right) \right\rceil. \tag{28}$$

The proof of Theorem 3 can be found in Section 4. We now comment the result:

1. Note that we do not assume that f be of the form (1); all that is needed is Assumption 1.
2. If $n = 1$, we recover Tseng's proximal gradient descent algorithm [23]. If $n > 1$, $\tau = 1$ and $\psi \equiv 0$, we obtain a new version of (serial) accelerated coordinate descent [12, 4] for minimizing smooth functions. Note that no existing accelerated coordinate descent methods are either proximal, *or* parallel. Our method is both proximal and parallel.

3. In the case when we update all blocks in one iteration ($\tau = n$), the bound (26) simplifies to

$$F(x_k) - F(x_*) \leq \frac{2\frac{\|v\|_1}{n}}{(k+2)^2}\|x_0 - x_*\|^2_{\tilde{v}}, \tag{29}$$

where as before, $\tilde{v} = nv/\|v\|_1$. There is no expectation here as the method is deterministic in this case.

If we use stepsize v proposed in Theorem 1, then in view of part (ii) of that theorem, bound (29) takes the form

$$F(x_k) - F(x_*) \leq \frac{2\bar{\omega}\bar{L}}{(k+2)^2}\|x_0 - x_*\|^2_w, \tag{30}$$

as advertised in the abstract. Recall that $\bar{\omega}$ is a data-weighted *average* of the values $\{\omega_j\}$.

In contrast, using the stepsizes proposed by Richtárik & Takáč (see Table 3), we get

$$F(x_k) - F(x_*) \leq \frac{2\omega\frac{\sum_i L_i}{n}}{(k+2)^2}\|x_0 - x_*\|^2_{\tilde{v}}. \tag{31}$$

Note that in the case when the functions f_j are convex quadratics ($f_j(x) = \frac{1}{2}(a_j^T x - b_j)^2$), for instance, we have $L_i = \sum_j L_{ji}$, and hence the new ESO leads to a vast improvement in the complexity in cases when $\bar{\omega} \ll \omega$. On he other hand, in cases where $L_i \ll \sum_j L_{ji}$ (which can happen with logistic regression, for instance), then the result based on the classical ESO may be better.

4. Consider the smooth case ($\psi \equiv 0$): $F = f$ and $f'(x_*) = 0$. By part (ii) of Theorem 1, ∇f is Lipschitz with constant 1 wrt $\|\cdot\|_w$. Choosing $x = x_*$ and $h = x_0 - x_*$, we get

$$f(x_0) - f(x_*) \leq \frac{1}{2}\|x_0 - x_*\|^2_w. \tag{32}$$

Now, consider running Algorithm 1 with a τ-nice sampling and stepsize parameter v as in Theorem 1. Letting $d = (d_1, \ldots, d_n)$, where d_i is defined by

$$\left(1 - \frac{\tau}{n}\right) w_i + v_i = \left(1 - \frac{\tau}{n}\right)\sum_j \omega_j L_{ji} + \sum_j \beta_j L_{ji} \leq \sum_j (\omega_j + 1) L_{ji} \stackrel{\text{def}}{=} d_i, \tag{33}$$

we get

$$\begin{aligned}
\mathbf{E}[f(x_k) - f(x_*)] &\overset{(26)+(32)}{\leq} \frac{2n^2}{(k\tau + 2n)^2}\|x_0 - x_*\|^2_{(1-\frac{\tau}{n})w+v} \\
&\overset{(33)}{\leq} \frac{2n^2}{(k\tau + 2n)^2}\|x_0 - x_*\|^2_d \overset{(11)+(13)}{\leq} \frac{2n^2(\bar{\omega}+1)\bar{L}}{(k\tau + 2n)^2}\|x_0 - x_*\|^2_{\tilde{d}},
\end{aligned}$$

where in the last step we have used the estimate $\omega_j + \beta_j - \frac{\tau\omega_j}{n} \in [\omega_j, \omega_j + 1]$, and $\tilde{d}$ is a scalar multiple of d for which $\|\tilde{d}\|_1 = 1$. Similarly as in (28), this means that

$$k \geq k(\tau) \stackrel{\text{def}}{=} \frac{n}{\tau}\sqrt{\frac{2(\bar{\omega}+1)\bar{L}}{\epsilon}}\|x_0 - x_*\|_{\tilde{d}}$$

iterations suffice to produce an ϵ-solution in expectation. Hence, we get *linear speedup* in the number of parallel updates / processors. This is *different* from the situation in simple (non-accelerated) parallel coordinate descent methods where parallelization speedup depends on the degree of separability (speedup is better if ω is small). In APPROX, the average degree of separability $\bar{\omega}$ is decoupled from τ, and hence one benefits from separability even for large τ. This means that *accelerated methods are better suitable for parallelization.*

5. We focused on the case of *uniform* samplings, but with a proper change in the definition of ESO, one can also handle *non-uniform* samplings [15].

4 Complexity analysis

We first establish four lemmas and then prove Theorem 3.

4.1 Lemmas

In the first lemma we summarize well-known properties of the sequence θ_k used in Algorithm 1.

Lemma 1 (Tseng [23])**.** *The sequence $\{\theta_k\}_{k\geq 0}$ defined in Algorithm 1 is decreasing and satisfies $0 < \theta_k \leq \frac{2}{k+2n/\tau} \leq \frac{\tau}{n} \leq 1$ and*

$$\frac{1-\theta_{k+1}}{\theta_{k+1}^2} = \frac{1}{\theta_k^2}. \tag{34}$$

We now give an explicit characterization of x_k as a convex combination of the vectors $z_0, \ldots, z_k$.

Lemma 2. *Let $\{x_k, z_k\}_{k\geq 0}$ be the iterates of Algorithm 1. Then for all $k \geq 0$ we have*

$$x_k = \sum_{l=0}^{k} \gamma_k^l z_l, \tag{35}$$

where the constants $\gamma_k^0, \gamma_k^1, \ldots, \gamma_k^k$ are non-negative and sum to 1. That is, x_k is a convex combination of the vectors $z_1, \ldots, z_k$. In particular, the constants are defined recursively in k by setting $\gamma_0^0 = 1$, $\gamma_1^0 = 0$, $\gamma_1^1 = 1$ and for $k \geq 1$,

$$\gamma_{k+1}^l = \begin{cases} (1-\theta_k)\gamma_k^l, & l = 0, \ldots, k-1, \\ \theta_k(1-\frac{n}{\tau}\theta_{k-1}) + \frac{n}{\tau}(\theta_{k-1}-\theta_k), & l = k, \\ \frac{n}{\tau}\theta_k, & l = k+1. \end{cases} \tag{36}$$

Moreover, for all $k \geq 0$, the following identity holds

$$\gamma_{k+1}^k + \frac{n-\tau}{\tau}\theta_k = (1-\theta_k)\gamma_k^k. \tag{37}$$

Proof. We proceed by induction. First, notice that $x_0 = z_0 = \gamma_0^0 z_0$, This implies that $y_0 = z_0$, which in turn together with $\theta_0 = \frac{\tau}{n}$ gives $x_1 = y_0 + \frac{n}{\tau}\theta_0(z_1 - x_0) = z_1 = \gamma_1^0 z_0 + \gamma_1^1 z_1$. Assuming

now that the relation holds for some $k \geq 1$, we obtain

$$\begin{aligned}
x_{k+1} &\overset{\text{(Alg 1, step 9)}}{=} y_k + \frac{n}{\tau}\theta_k(z_{k+1} - z_k) \\
&\overset{\text{(Alg 1, step 3)}}{=} (1-\theta_k)x_k + \theta_k z_k - \frac{n}{\tau}\theta_k z_k + \frac{n}{\tau}\theta_k z_{k+1} \\
&= \sum_{l=1}^{k-1}(1-\theta_k)\gamma_k^l z_l + \left((1-\theta_k)\gamma_k^k + \theta_k - \frac{n}{\tau}\theta_k\right) z_k + \frac{n}{\tau}\theta_k z_{k+1} \\
&= \sum_{l=1}^{k-1} \underbrace{(1-\theta_k)\gamma_k^l}_{\gamma_{k+1}^l} z_l + \underbrace{\left(\theta_k(1-\frac{n}{\tau}\theta_{k-1}) + \frac{n}{\tau}(\theta_{k-1}-\theta_k)\right)}_{\gamma_{k+1}^k} z_k + \underbrace{\left(\frac{n}{\tau}\theta_k\right)}_{\gamma_{k+1}^{k+1}} z_{k+1}.
\end{aligned}$$

By applying Lemma 1, together with the inductive assumption that $\gamma_k^l \geq 0$ for all l, we observe that $\gamma_{k+1}^l \geq 0$ for all l. It remains to show that the constants sum to 1. This is true since x_k is a convex combination of $z_1, \ldots, z_k$, and by (38), x_{k+1} is an affine combination of x_k, z_k and z_{k+1}. Identity (37) is verified by direct substitution. □

Define

$$\begin{aligned}
\tilde{z}_{k+1} &\overset{\text{def}}{=} \arg\min_{z\in\mathbf{R}^N}\left\{\psi(z) + \langle \nabla f(y_k), z - y_k\rangle + \frac{n\theta_k}{2\tau}\|z - z_k\|_v^2\right\} \\
&\overset{(5)+(25)}{=} \arg\min_{z=(z^{(1)},\ldots,z^{(n)})\in\mathbf{R}^N}\sum_{i=1}^n\left\{\psi_i(z^{(i)}) + \langle \nabla_i f(y_k), z^{(i)} - y_k^{(i)}\rangle + \frac{n\theta_k v_i}{2\tau}\|z^{(i)} - z_k^{(i)}\|_{(i)}^2\right\}.
\end{aligned}$$

From this and the definition of z_{k+1} we see that

$$z_{k+1}^{(i)} = \begin{cases} \tilde{z}_{k+1}^{(i)}, & i \in S_k \\ z_k^{(i)}, & i \notin S_k. \end{cases} \tag{38}$$

The next lemma is an application to a specific function of a well-known result that can be found, for instance, in [23]. The result was used by Tseng to construct a simplified complexity proof for a proximal gradient descent method. This lemma requires the norms $\|\cdot\|_{(i)}$ to be Euclidean – and this is the only place in our analysis where this is required.

Lemma 3 (Property 1 in [23])**.** *Let* $\xi(u) \overset{def}{=} f(y_k) + \langle \nabla f(y_k), u - y_k\rangle + \frac{n\theta_k}{2\tau}\|u - z_k\|_v^2$. *Then*

$$\psi(\tilde{z}_{k+1}) + \xi(\tilde{z}_{k+1}) \leq \psi(x_*) + \xi(x_*) - \frac{n\theta_k}{2\tau}\|x_* - \tilde{z}_{k+1}\|_v^2. \tag{39}$$

Our next lemma is a technical result connecting the gradient mapping (producing $\tilde{z}_{k+1}$) and the stochastic block gradient mapping (producing the random vector z_{k+1}). The lemma reduces to a trivial identity in the case when all $n = 1$. From now on, by $\mathbf{E}_k$ we denote the expectation with respect to S_k, keeping everything else fixed.

Lemma 4. *For any* $x \in \mathbf{R}^N$ *and* $k \geq 0$,

$$\mathbf{E}_k\left[\|z_{k+1} - x\|_v^2 - \|z_k - x\|_v^2\right] = \frac{\tau}{n}\left(\|\tilde{z}_{k+1} - x\|_v^2 - \|z_k - x\|_v^2\right). \tag{40}$$

Moreover,

$$\mathbf{E}_k\left[\psi(z_{k+1})\right] = \left(1 - \frac{\tau}{n}\right)\psi(z_k) + \frac{\tau}{n}\psi(\tilde{z}_{k+1}). \tag{41}$$

Proof. Let $\hat{S}$ be any uniform sampling and $a, h \in \mathbf{R}^N$. Theorem 4 in [16] implies that

$$\mathbf{E}[\|h_{[\hat{S}]}\|_v^2] = \tfrac{\tau}{n}\|h\|_v^2, \quad \mathbf{E}[\langle a, h_{[\hat{S}]}\rangle_v] = \tfrac{\tau}{n}\langle a, h\rangle_v, \quad \mathbf{E}[\psi(a + h_{[\hat{S}]})] = \left(1 - \tfrac{\tau}{n}\right)\psi(a) + \tfrac{\tau}{n}\psi(a+h), \quad (42)$$

where for $\langle a, h\rangle_v \overset{\text{def}}{=} \sum_{i=1}^n v_i \langle a^{(i)}, h^{(i)}\rangle$. Let $h = \tilde{z}_{k+1} - z_k$. In view of (3) and (38), we can write $z_{k+1} - z_k = h_{[S_k]}$. Applying the first two identities in (42) with $a = z_k - x$ and $\hat{S} = S_k$, we get

$$\begin{aligned}\mathbf{E}_k\left[\|z_{k+1} - x\|_v^2 - \|z_k - x\|_v^2\right] &= \mathbf{E}_k\left[\|h_{[S_k]}\|_v^2 + 2\langle z_k - x, h_{[S_k]}\rangle_v\right] \\ &\overset{(42)}{=} \frac{\tau}{n}\left(\|h\|_v^2 + 2\langle z_k - x, h\rangle_v\right) = \frac{\tau}{n}\left(\|\tilde{z}_{k+1} - x\|_v^2 - \|z_k - x\|_v^2\right).\end{aligned}$$

The remaining statement follows from the last identity in (42) used with $a = z_k$. □

4.2 Proof of Theorem 3

Using Lemma 2 and convexity of ψ, for all $k \geq 0$ we have

$$\psi(x_k) \overset{(35)}{=} \psi\left(\sum_{l=0}^k \gamma_k^l z_l\right) \overset{\text{(convexity)}}{\leq} \sum_{l=0}^k \gamma_k^l \psi(z_l) \overset{\text{def}}{=} \hat{\psi}_k. \quad (43)$$

From this we get

$$\begin{aligned}\mathbf{E}_k[\hat{\psi}_{k+1}] &\overset{(43)+(36)}{=} \sum_{l=0}^k \gamma_{k+1}^l \psi(z_l) + \frac{n}{\tau}\theta_k \mathbf{E}_k\left[\psi(z_{k+1})\right] \\ &\overset{(41)}{=} \sum_{l=0}^k \gamma_{k+1}^l \psi(z_l) + \frac{n}{\tau}\theta_k\left(\left(1 - \frac{\tau}{n}\right)\psi(z_k) + \frac{\tau}{n}\psi(\tilde{z}_{k+1})\right) \\ &= \sum_{l=0}^k \gamma_{k+1}^l \psi(z_l) + \left(\frac{n}{\tau} - 1\right)\theta_k\psi(z_k) + \theta_k\psi(\tilde{z}_{k+1}). \quad (44)\end{aligned}$$

Since $x_{k+1} = y_k + h_{[S_k]}$ with $h = \frac{n}{\tau}\theta_k(\tilde{z}_{k+1} - z_k)$, we can use ESO to bound

$$\begin{aligned}\mathbf{E}_k[f(x_{k+1})] &\overset{(7)}{\leq} f(y_k) + \theta_k\langle \nabla f(y_k), \tilde{z}_{k+1} - z_k\rangle + \frac{n\theta_k^2}{2\tau}\|\tilde{z}_{k+1} - z_k\|_v^2 \\ &= (1 - \theta_k)f(y_k) - \theta_k\langle \nabla f(y_k), z_k - y_k\rangle \\ &\quad + \theta_k\Big(f(y_k) + \langle \nabla f(y_k), \tilde{z}_{k+1} - y_k\rangle + \frac{n\theta_k}{2\tau}\|\tilde{z}_{k+1} - z_k\|_v^2\Big). \quad (45)\end{aligned}$$

Note that from the definition of y_k in the algorithm, we have

$$\theta_k(y_k - z_k) = ((1 - \theta_k)x_k - y_k) + \theta_k y_k = (1 - \theta_k)(x_k - y_k). \quad (46)$$

For all $k \geq 0$ we define an upper bound on $F(x_k)$,

$$\hat{F}_k \overset{\text{def}}{=} \hat{\psi}_k + f(x_k) \overset{(43)}{\geq} F(x_k), \quad (47)$$

and bound the expectation of $\hat{F}_{k+1}$ in S_k as follows:

$$
\begin{aligned}
\mathbf{E}_k[\hat{F}_{k+1}] &= \mathbf{E}_k[\hat{\psi}_{k+1}] + \mathbf{E}_k[f(x_{k+1})] \\
&\overset{(44)+(45)}{\leq} \sum_{l=0}^{k} \gamma_{k+1}^l \psi(z_l) + \frac{n-\tau}{\tau}\theta_k \psi(z_k) + (1-\theta_k) f(y_k) - \theta_k \langle \nabla f(y_k), z_k - y_k \rangle \\
&\quad + \theta_k \Big(\psi(\tilde{z}_{k+1}) + f(y_k) + \langle \nabla f(y_k), \tilde{z}_{k+1} - y_k \rangle + \frac{n\theta_k}{2\tau} \|\tilde{z}_{k+1} - z_k\|_v^2 \Big) \\
&\overset{(39)}{\leq} \sum_{l=0}^{k} \gamma_{k+1}^l \psi(z_l) + \frac{n-\tau}{\tau}\theta_k \psi(z_k) + (1-\theta_k) f(y_k) - \theta_k \langle \nabla f(y_k), z_k - y_k \rangle \\
&\quad + \theta_k \Big(\psi(x_*) + f(y_k) + \langle \nabla f(y_k), x_* - y_k \rangle + \frac{n\theta_k}{2\tau} \|x_* - z_k\|_v^2 - \frac{n\theta_k}{2\tau} \|x_* - \tilde{z}_{k+1}\|_v^2 \Big) \\
&\overset{(46)}{=} \sum_{l=0}^{k-1} \underbrace{\gamma_{k+1}^l}_{\overset{(36)}{=}(1-\theta_k)\gamma_k^l} \psi(z_l) + \underbrace{\left(\gamma_{k+1}^k + \frac{n-\tau}{\tau}\theta_k \right)}_{\overset{(37)}{=}(1-\theta_k)\gamma_k^k} \psi(z_k) \\
&\quad + \underbrace{(1-\theta_k) f(y_k) + (1-\theta_k) \langle \nabla f(y_k), x_k - y_k \rangle}_{\leq (1-\theta_k) f(x_k)} \\
&\quad + \theta_k \Big(\underbrace{\psi(x_*) + f(y_k) + \langle \nabla f(y_k), x_* - y_k \rangle}_{\leq F(x_*)} + \frac{n\theta_k}{2\tau} \|x_* - z_k\|_v^2 - \frac{n\theta_k}{2\tau} \|x_* - \tilde{z}_{k+1}\|_v^2 \Big) \\
&\overset{(43)+(47)}{\leq} (1-\theta_k)\hat{F}_k + \theta_k F(x_*) + \frac{n\theta_k^2}{2\tau} \left(\|x_* - z_k\|_v^2 - \|x_* - \tilde{z}_{k+1}\|_v^2 \right) \\
&\overset{(40)}{=} (1-\theta_k)\hat{F}_k + \theta_k F(x_*) + \frac{n^2\theta_k^2}{2\tau^2} \left(\|x_* - z_k\|_v^2 - \mathbf{E}_k \left[\|x_* - z_{k+1}\|_v^2 \right] \right). \qquad (48)
\end{aligned}
$$

After dividing both sides of (48) by θ_k^2, using (34), and rearranging the terms, we obtain

$$
\frac{1-\theta_{k+1}}{\theta_{k+1}^2} \mathbf{E}_k[\hat{F}_{k+1} - F(x_*)] + \frac{n^2}{2\tau^2} \mathbf{E}_k[\|x_* - z_{k+1}\|_v^2] \;\leq\; \frac{1-\theta_k}{\theta_k^2}(\hat{F}_k - F(x_*)) + \frac{n^2}{2\tau^2} \|x_* - z_k\|_v^2.
$$

We now apply total expectation to the above inequality and unroll the recurrence for l between 0 and k, obtaining

$$
\frac{1-\theta_k}{\theta_k^2} \mathbf{E}[\hat{F}_k - F(x_*)] + \frac{n^2}{2\tau^2} \mathbf{E}[\|x_* - z_{k+1}\|_v^2] \;\leq\; \frac{1-\theta_0}{\theta_0^2}(\hat{F}_0 - F(x_*)) + \frac{n^2}{2\tau^2} \|x_* - z_0\|_v^2, \quad (49)
$$

from which we finally get

$$
\begin{aligned}
\mathbf{E}[F(x_k) - F(x_*)] &\overset{(47)}{\leq} \mathbf{E}[\hat{F}_k - F(x_*)] \\
&\overset{(49)}{\leq} \frac{\theta_k^2}{\theta_0^2}(1-\theta_0)(\hat{F}_0 - F(x_*)) + \frac{n^2\theta_k^2}{2\tau^2} \|x_* - z_0\|_v^2 \\
&\leq \frac{4n^2}{(k\tau + 2n)^2} \Big(\left(1 - \frac{\tau}{n}\right) (F(x_0) - F(x_*)) + \frac{1}{2} \|x_0 - x_*\|_v^2 \Big),
\end{aligned}
$$

where in the last step we have used the facts that $\hat{F}_0 = F(x_0)$, $x_0 = z_0$ $\theta_0 = \frac{\tau}{n}$ and the estimate $\theta_k^2 \leq \frac{2}{k + 2n/\tau}$ from Lemma 1.

5 Implementation without full-dimensional vector operations

Algorithm 1, as presented, performs full-dimensional vector operations. Indeed, y_k is defined as a convex combination of x_k and z_k. Also, x_{k+1} is obtained from y_k by changing $|S_k|$ coordinates; however, if $|S_k|$ is small, the latter operation is not costly. In any case, vectors x_k and z_k will in general be dense, and hence computation of y_k may cost $O(N)$ arithmetic operations. However, simple (i.e., non-accelerated) coordinate descent methods are successful and popular precisely because they can avoid such operations.

Borrowing ideas from Lee & Sidford [4], we rewrite[7] Algorithm 1 into a new form, incarnated as Algorithm 2.

Algorithm 2 APPROX (written in a form facilitating efficient implementation)

Pick $\tilde{z}_0 \in \mathbf{R}^N$ and set $\theta_0 = \frac{\tau}{n}$, $u_0 = 0$
for $k \geq 0$ **do**
  Generate a random set of coordinates $S_k \sim \hat{S}$
  $u_{k+1} \leftarrow u_k$, $\tilde{z}_{k+1} \leftarrow \tilde{z}_k$
  for $i \in S_k$ **do**
    $t_k^{(i)} = \arg\min_{t \in \mathbf{R}^{N_i}} \left\{ \langle \nabla_i f(\theta_k^2 u_k + \tilde{z}_k), t \rangle + \frac{n\theta_k v_i}{2\tau} \|t\|_{(i)}^2 + \psi_i(\tilde{z}_k^{(i)} + t) \right\}$
    $\tilde{z}_{k+1}^{(i)} \leftarrow \tilde{z}_k^{(i)} + t_k^{(i)}$
    $u_{k+1}^{(i)} \leftarrow u_k^{(i)} - \frac{1 - \frac{n}{\tau}\theta_k}{\theta_k^2} t_k^{(i)}$
  end for
  $\theta_{k+1} = \frac{\sqrt{\theta_k^4 + 4\theta_k^2} - \theta_k^2}{2}$
end for
OUTPUT: $\theta_k^2 u_{k+1} + \tilde{z}_{k+1}$

As it is not immediately obvious that the two methods are equivalent, we include the following result.

Proposition 1 (Equivalence)**.** *Run Algorithm 2 with $\tilde{z}_0 = x_0$, where $x_0 \in \operatorname{dom} \psi$ is the starting point of Algorithm 1. If we define*

$$\tilde{x}_k = \begin{cases} \tilde{z}_0, & k = 0, \\ \theta_{k-1}^2 u_k + \tilde{z}_k, & k \geq 1, \end{cases} \tag{50}$$

and

$$\tilde{y}_k = \theta_k^2 u_k + \tilde{z}_k, \quad k \geq 0, \tag{51}$$

then $x_k = \tilde{x}_k$, $y_k = \tilde{y}_k$ and $z_k = \tilde{z}_k$ for all $k \geq 0$. That is, Algorithms 1 and 2 are equivalent.

Note that in Algorithm 2 we never need to form x_k throughout the iterations. The only time this is needed is when producing the output: $x_{k+1} = \theta_k^2 u_{k+1} + z_{k+1}$. More importantly, note that the method does need to explicitly compute y_k. Instead, we introduce a new vector, u_k, and express y_k as $y_k = \theta_k^2 u_k + \tilde{z}_k$. Note that the method accesses y_k only via the block-gradients $\nabla_i f(y_k)$ for $i \in S_k$. Hence, if it is possible to cheaply compute these gradients *without* actually forming y_k, we can avoid full-dimensional operations.

[7] Note that we override the notation $\tilde{z}_k$ here – it now has a different meaning from that in Section 4.

We now show that this can be done for functions f of the form (1), where f_j is as in Theorem 2. That is,

$$f(x) = \sum_{j=1}^{m} \phi_j(e_j^T Ax). \tag{52}$$

Let D_i be the set of such j for which $A_{ji} \neq 0$. If we write $r_{u_k} = Au_k$ and $r_{\tilde{z}_k} = A\tilde{z}_k$, then using (52) we can write

$$\nabla_i f(\theta_k^2 u_k + \tilde{z}_k) = \sum_{j \in D_i} A_{ji}^T \phi_j'(\theta_k^2 r_{u_k}^j + r_{\tilde{z}_k}^j). \tag{53}$$

Assuming we store and maintain the residuals r_{u_k} and $r_{\tilde{z}_k}$, the computation of the product $A_{ji}^T \phi_j'(\cdot)$ costs $\mathcal{O}(N_i)$, and hence the computation of the block derivative (53) requires $\mathcal{O}(|D_i|N_i)$ arithmetic operations. Hence on average, computing all block gradients for $i \in S_k$ will cost

$$C = \mathbf{E}\left[\sum_{i \in \hat{S}} \mathcal{O}(|D_i|N_i)\right] = \frac{\tau}{n} \sum_{i=1}^{n} \mathcal{O}(|D_i|N_i).$$

This will be small if $|D_i|$ are small and τ is small. For simplicity, assume all blocks are of equal size, $N_i = b = N/n$. Then

$$C = \frac{b\tau}{n} \times \mathcal{O}\left(\sum_{i=1}^{n} |D_i|\right) = \frac{b\tau}{n} \times \mathcal{O}\left(\sum_{j=1}^{m} \omega_j\right) = \frac{b\tau m}{n} \mathcal{O}(\bar{\omega}) = \tau \times \mathcal{O}\left(\frac{bm\bar{\omega}}{n}\right).$$

In many practical situations, $m \leq n$,and often $m \ll n$ (we focus on this case in the paper since usually this corresponds to f not being strongly convex) and $\bar{\omega} = O(1)$. This then means that $C = \tau \times \mathcal{O}(b)$. That is, each of the τ processors do work proportional to the size of a single block per iteration.

The favorable situation described above is the consequence of the block sparsity of the data matrix A and does not depend on ϕ_j insofar as the evaluation of its derivative takes $\mathcal{O}(1)$ work. Hence, it applies to convex quadratics ($\phi_j(s) = s^2$), logistic regression ($\phi_j(r) = \log(1 + \exp(s))$) and also to the smooth approximation $f_\mu(x)$ of $f(x) = \|Ax - b\|_1$, defined by

$$f_\mu(x) = \sum_{j=1}^{m} \|e_j^T A\|_{w^*}^* \psi_\mu\left(\frac{|e_j^T Ax - b_j|}{\|e_j^T A\|_v^*}\right), \qquad \psi_\mu(t) = \begin{cases} \frac{t^2}{2\mu}, & 0 \leq t \leq \mu, \\ t - \frac{\mu}{2}, & \mu \leq t, \end{cases}$$

with smoothing parameter $\mu > 0$, as considered in [11, 3]. Vector w^* is as defined in [3]; $\|\cdot\|_v$ is a weighted norm in $\mathbf{R}^m$.

6 Numerical experiments

In all tests we used a shared-memory workstation with 32 Intel Xeon processors at 2.6 GHz and 128 GB RAM. In the experiments, we have departed from the theory in two ways: i) our implementation of APPROX is *asynchronous* in order to limit communication costs, and ii) we approximated the τ-nice sampling by a τ-independent sampling as in [16] (the latter is very easy to generate in parallel; please note that our analysis can be very easily extended to cover the τ-independent sampling). For simplicity, in all tests we assume all blocks are of size 1 ($N_i = 1$ for all i). However, further speedups can be obtained by working with larger block sizes as then each processor is better utilized.

6.1 The effect of new stepsizes

In this experiment, we compare the performance of the new stepsizes (introduced in Section 2.2) with those proposed in [16] (see Table 3). We generated random instances of the L_1-regularized least squares problem (LASSO),

$$f(x) = \frac{1}{2}\|Ax - b\|^2, \qquad \psi(x) = \lambda\|x\|_1,$$

with various distributions of the separability degrees ω_j (= number of nonzero elements on the jth row of A) and studied the weighted distance to the optimum $\|x_* - x_0\|_v$ for the initial point $x_0 = 0$. This quantity appears in the complexity estimate (28) and depends on τ (the number of processors). We chose a random matrix of small size: $N = m = 1000$ as this is sufficient to make our point, and consider $\tau \in \{10, 100, 1000\}$.

In particular, we consider three different distributions of $\{\omega_j\}$: uniform, intermediate and extreme. The results are summarized in Table 4. First, we generated a *uniformly* sparse matrix with $\omega_j = 30$ for all j. In this case, $v^{\rm fr} = v^{\rm rt}$, and hence the results are the same. We then generated an *intermediate* instance, with $\omega_j = 1 + \lfloor 30j^2/m^2 \rfloor$. The matrix has many rows with a few nonzero elements and some rows with up to 30 nonzero elements. Looking at the table, clearly, the new stepsizes are better. The improvement is moderate when there are a few processors, but for $\tau = 1000$, the complexity is 25% better. Finally, we generated a rather *extreme* matrix with $\omega_1 = 500$ and $\omega_j = 3$ for $j > 1$. We can see that the new stepsizes are much better, even with few processors,and can lead to $5\times$ speedup.

	Uniform		Intermediate		Extreme	
τ	$\|x^*\|_{v^{\rm fr}}$	$\|x^*\|_{v^{\rm rt}}$	$\|x^*\|_{v^{\rm fr}}$	$\|x^*\|_{v^{\rm rt}}$	$\|x^*\|_{v^{\rm fr}}$	$\|x^*\|_{v^{\rm rt}}$
10	10.82	10.82	6.12	6.43	2.78	5.43
100	19.00	19.00	9.30	11.38	4.31	16.08
1000	52.49	52.49	24.00	31.78	11.32	50.52

Table 4: Comparison of ESOs in the uniform case

In the experiments above, we have first fixed a sparsity pattern and then generated a *random* matrix A based on it. However, much larger differences can be seen for special matrices A. We shall now comment on this.

Consider the case $\tau = n$. In view of (29), the complexity of APPROX is proportional to $\|v\|_1$. Fix ω and $\omega_1, \ldots, \omega_j$ and let us ask the question: for what data matrix A will the ratio $\theta = \|v^{\rm rt}\|_1/\|v^{\rm fr}\|_1$ be maximized? Since $\|v^{\rm rt}\|_1 = \omega \sum_j \|A_{j:}\|^2$ and $\|v^{\rm fr}\|_1 = \sum_j \omega_j \|A_{j:}\|^2$, we the maximal ratio is given by

$$\max_A \theta \stackrel{\rm def}{=} \max_{\alpha \geq 0} \left\{ \omega \sum_{j=1}^m \alpha_j \ : \ \sum_{j=1}^m \omega_j \alpha_j \leq 1 \right\} = \max_j \frac{\omega}{\omega_j}.$$

The extreme case is attained for some matrix with at least one dense row (ω_j) and one maximally sparse row ($\omega_j = 1$), leading to $\theta = n$. So, there are instances for which the new stesizes can lead to an up to $n\times$ speedup for APPROX when compared to the stepsizes $v^{\rm rt}$. Needless to say, these extreme instances are artificially constructed.

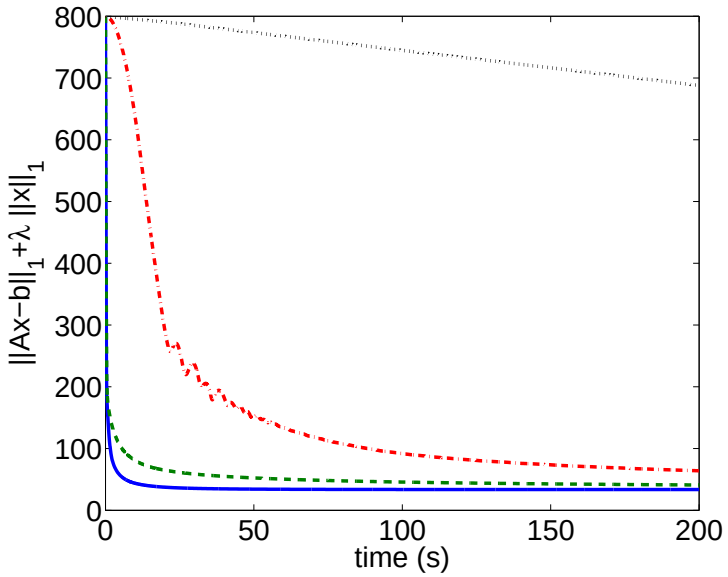

Figure 1: Comparison of four algorithms for L_1 regularized L_1 regression on the `dorothea` dataset: gradient method (dotted black line), accelerated gradient method ([11], dash-dotted red line), smoothed parallel coordinate descent method (SPCDM [3], dashed green line) and APPROX with stepsizes v^{fr} (solid blue line).

6.2 L1-regularized L1 regression

We consider the data given in the dorothea dataset [13]. It is a sparse moderate-sized feature matrix A with m=800, N=100,000, ω=6,061 and a vector $b \in \mathbf{R}^m$. We wish to find $x \in \mathbb{R}^N$ that minimizes

$$\|Ax - b\|_1 + \lambda \|x\|_1$$

with $\lambda = 1$. Because the objective is nonsmooth and non-separable, we apply the smoothing technique presented in [11] for the first part of the objective and use the smoothed parallel coordinate descent method proposed in [3] (this methods needs special stepsizes which are studied in that paper). The level of smoothing depends on the expected accuracy: we chose $\epsilon = 0.1$, which corresponds to 0.0125% of the initial value.

We compared 4 algorithms (see Figure 1), all run with 4 processors. As one can see, the coordinate descent method is very efficient on this problem. However, the accelerated coordinate descent is still able to outperform it. As the problem is of small size (which is sufficient for the sake of comparison), we could compute the optimal solution using an interior point method for linear programming and compare the value at each iteration to the optimal value (Table 5). Each line of the table gives the time needed by APPROX and PCDM to reach a given accuracy target. In the beginning (until $F(x_k) - F(x^*) < 6.4$), the algorithms are in a transitional phase. Then, when one runs the algorithm twice as long, $F(x_k) - F(x^*)$ is divided by 2 for SPCDM and by 4 for APPROX. This highlights the difference in the convergence speeds: $O(1/k)$ compared to $O(1/k^2)$. As a result, APPROX gives an ϵ-solution in 156.5 seconds while SPCDM has not finished yet after 2000 seconds.

$F(x_k) - F(x_*)$	APPROX	SPCDM
409.6	0.2 s	0.2 s
204.8	0.3 s	0.4 s
102.4	1.0 s	2.3 s
51.2	2.2 s	8.8 s
25.6	4.5 s	29.2 s
12.8	8.3 s	93.4 s
6.4	14.4 s	246.6 s
3.2	22.8 s	562.3 s
1.6	34.4 s	1082.1 s
0.8	50.1 s	1895.3 s
0.4	71.8 s	>2000 s
0.2	103.4 s	>2000 s
0.1	156.5 s	>2000 s

Table 5: Comparison of objective decreases for APPROX and smoothed parallel coordinate descent (SPCDM) on a problem with $F(x) = \|Ax - b\|_1 + \lambda\|x\|_1$.

6.3 Lasso

We now consider L_1 regularized least squares regression on the `KDDB` dataset [13]. It consists of a medium size sparse feature matrix A with $m = 29,890,095$, $N = 19,264,097$ and $\omega = 75$, and a vector $b \in \mathbf{R}^m$. We wish to find $x \in \mathbf{R}^N$ that minimizes

$$F(x) = \frac{1}{2}\|Ax - b\|^2 + \lambda\|x\|_1$$

with $\lambda = 1$.

We compare APPROX (Algorithm 2) with the (non-accelerated) parallel coordinate descent method (PCDM [16]) in Figure 2, both run with $\tau = 16$ processors.

Both algorithms converge quickly. PCDM is faster in the beginning because each iteration is half as expensive. However, APPROX is faster afterwards. For this problem, the optimal value is not known so it is difficult to compare the actual accuracy.

Let us remark that an important feature of the L_1-regularization is that it promotes sparsity in the optimization variable x. As APPROX only involves proximal steps on the z variable, only z_k is encouraged to be sparse but not x_k, y_k or u_k. A possible way to obtain a sparse solution with APPROX is to first compute x_k and then post-process with a few iterations of a sparsity-oriented method (such as iterative hard thresholding, full proximal gradient descent or cyclic/randomized coordinate descent).

6.4 Training linear support vector machines

Our last experiment is the dual of Support Vector Machine problem [18]. For the dual SVM, the coordinates correspond to examples.

We use the Malicious URL dataset [13] with data matrix A of size $m = 2,396,130$, $N = 3,231,961$ and a vector $b \in \mathbb{R}^N$. Here $\omega = n$ (and hence the data set is not particularly suited for parallel coordinate descent methods) but the matrix is still sparse (nnz=$277,058,644 \ll mn$).

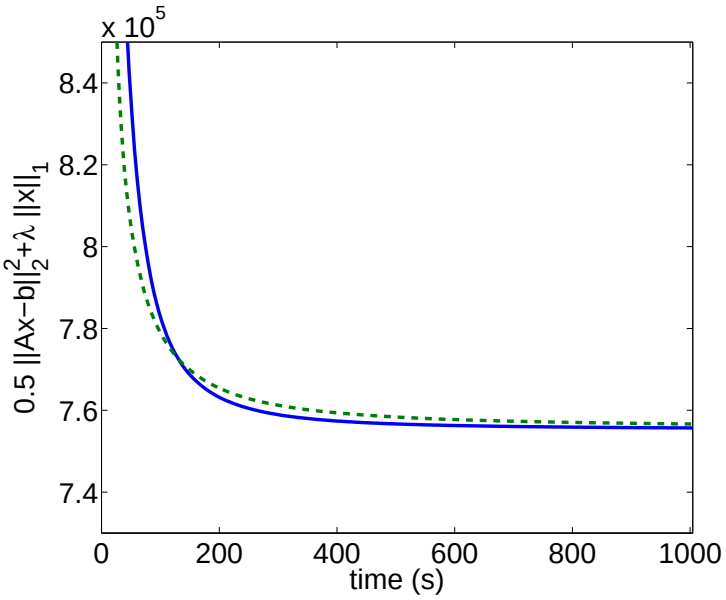

Figure 2: Comparison of PCDM and APPROX for l_1 regularized least squares regression on the kddb dataset. As the decrease is very big in the first seconds (from 8.3 10^8 to 8.5 10^5), we present a zoom for $7.3 \leq F(x) \leq 8.5$. Randomized coordinate descent [16]: dashed green line. Accelerated coordinate descent (Algorithm 2): solid blue line.

We wish to find $x \in [0,1]^N$ that minimizes

$$F(x) = \frac{1}{2\lambda N^2} \sum_{j=1}^{m} \left(\sum_{i=1}^{N} b_i A_{ji} x_i \right)^2 - \frac{1}{N} \sum_{i=1}^{N} x_i + I_{[0,1]^N}(x),$$

with $\lambda = 1/N$. We compare APPROX (Algorithm 2) with Stochastic Dual Coordinate Ascent (SDCA [18, 21]); the results are in Figure 2. We have used a single processor only ($\tau = 1$).

For this problem, one can recover a primal solution [18] and thus we can compare the decrease in the duality gap; summarized in Table 6. One can see that APPROX is about twice as fast as SDCA on this instance.

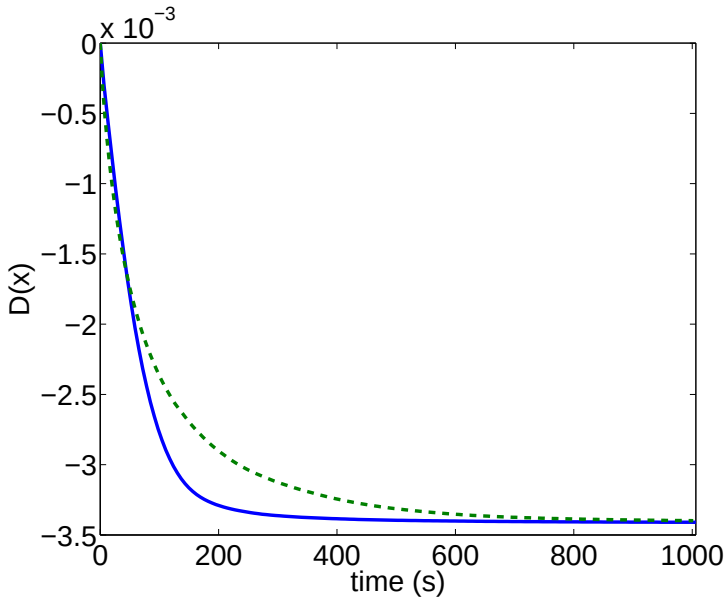

Figure 3: Comparison of PCDM and APPROX for the dual of the Support Vector Machine problem on the Malicious URL dataset. Randomized coordinate descent [16]: dashed green line. Accelerated coordinate descent (Algorithm 2): solid blue line.

Duality gap	APPROX	SDCA
0.0256	33 s	26 s
0.0128	59 s	97 s
0.0064	91 s	206 s
0.0032	137 s	310 s
0.0016	182 s	452 s
0.0008	273 s	606 s
0.0004	407 s	864 s
0.0002	614 s	1148 s
0.0001	954 s	1712 s

Table 6: Decrease of the duality gap for accelerated parallel coordinate descent (APPROX) and stochastic dual coordinate ascent (SDCA).

7 Conclusion

In summary, we proposed APPROX: a stochastic coordinate descent method combining the following *four acceleration strategies:*

1. Our method is *accelerated*, i.e., it achieves a $O(1/k^2)$ convergence rate. Hence, the method is better able to obtain a high-accuracy solution on non-strongly convex problem instances.

2. Our method is *parallel*. Hence, it is able to better utilize modern parallel computing architectures and effectively taming the problem dimension n.

3. We have proposed new *longer stepsizes* for faster convergence on functions whose degree of separability ω is larger than their degree of separability $\bar{\omega}$.

4. We have shown that our method can be implemented *without the need to perform full-dimensional vector operations.*

References

[1] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[2] Joseph K. Bradley, Aapo Kyrola, Danny Bickson, and Carlos Guestrin. Parallel coordinate descent for L1-regularized loss minimization. In *28th International Conference on Machine Learning*, 2011.

[3] Olivier Fercoq and Peter Richtárik. Smooth minimization of nonsmooth functions by parallel coordinate descent. *arXiv:1309.5885*, 2013.

[4] Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. *arXiv:1305.1922*, 2013.

[5] Dennis Leventhal and Adrian S. Lewis. Randomized methods for linear constraints: Convergence rates and conditioning. *Mathematics of Operations Research*, 35(3):641–654, 2010.

[6] Ji Liu, Stephen J. Wright, Christopher Ré, and Victor Bittorf. An asynchronous parallel stochastic coordinate descent algorithm. *arXiv:1311.1873*, 2013.

[7] Ion Necoara and Dragos Clipici. Distributed coordinate descent methods for composite minimization. Technical report, University Politehnica Bucharest, 2013.

[8] Ion Necoara and Dragos Clipici. Efficient parallel coordinate descent algorithm for convex optimization problems with separable constraints: application to distributed mpc. *Journal of Process Control*, 23:243–253, 2013.

[9] Ion Necoara, Yurii Nesterov, and Francois Glineur. Efficiency of randomized coordinate descent methods on optimization problems with linearly coupled constraints. Technical report, Politehnica University of Bucharest, 2012.

[10] Yurii Nesterov. A method of solving a convex programming problem with convergence rate O(1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[11] Yurii Nesterov. Smooth minimization of nonsmooth functions. *Mathematical Programming*, 103:127–152, 2005.

[12] Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[13] John C Platt. Fast training of support vector machines using sequential minimal optimization. In Bernhard Scholkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999. http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/binary.html.

[14] Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. *arXiv:1310.2059*, 2013.

[15] Peter Richtárik and Martin Takáč. On optimal probabilities in stochastic coordinate descent methods. *arXiv:1310.3438*, 2013.

[16] Peter Richtárik and Martin Takáč. Parallel coordinate descent methods for big data optimization problems. *arXiv:1212.0873*, 2012.

[17] Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming, Ser. A (doi: 10.1007/s10107-012-0614-z)*, preprint: April 2011.

[18] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for ℓ_1-regularized loss minimization. *Journal of Machine Learning Research*, 12:1865–1892, 2011.

[19] Shai Shalev-Shwartz and Tong Zhang. Proximal stochastic dual coordinate ascent. Technical report, 2012.

[20] Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *arXiv:1309.2375*, 2013.

[21] Martin Takáč, Avleen Bijral, Peter Richtárik, and Nathan Srebro. Mini-batch primal and dual methods for SVMs. In *30th International Conference on Machine Learning*, 2013.

[22] Rachael Tappenden, Peter Richtárik, and Jacek Gondzio. Inexact block coordinate descent method: complexity and preconditioning. *arXiv:1304.5530*, 2013.

[23] Paul Tseng. On accelerated proximal gradient methods for convex-concave optimization. *Submitted to SIAM Journal on Optimization*, 2008.

[24] Tong Tong Wu and Kenneth Lange. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, 2(1):224–244, 2008.

[25] Lin Xiao and Zhaosong Lu. On the complexity analysis of randomized block-coordinate descent methods. *arXiv:1305.4723*, 2013.

A Proof of Proposition 1 (equivalence)

It is straightforward to see that $x_0 = y_0 = z_0 = \tilde{x}_0 = \tilde{y}_0 = \tilde{z}_0$ and hence the statement holds for $k = 0$. By induction, assume it holds for some k. Note that for $i \notin S_k$, $\tilde{z}_{k+1}^{(i)} = \tilde{z}_k^{(i)} = z_k^{(i)} = z_{k+1}^{(i)}$. If $i \in S_k$, then

$$\tilde{z}_{k+1}^{(i)} = \tilde{z}_k^{(i)} + t_k^{(i)},$$

where

$$\begin{aligned}
t_k^{(i)} &= \arg\min_{t \in \mathbf{R}^{N_i}} \left\{ \langle \nabla_i f(\theta_k^2 u_k + \tilde{z}_k), t \rangle + \frac{n\theta_k v_i}{2\tau} \|t\|_{(i)}^2 + \psi_i(\tilde{z}_k^{(i)} + t) \right\} \\
&\overset{(51)}{=} \arg\min_{t \in \mathbf{R}^{N_i}} \left\{ \langle \nabla_i f(\tilde{y}_k), t \rangle + \frac{n\theta_k v_i}{2\tau} \|t\|_{(i)}^2 + \psi_i(\tilde{z}_k^{(i)} + t) \right\} \\
&= \arg\min_{t \in \mathbf{R}^{N_i}} \left\{ \langle \nabla_i f(y_k), t \rangle + \frac{n\theta_k v_i}{2\tau} \|t\|_{(i)}^2 + \psi_i(z_k^{(i)} + t) \right\} \\
&= -z_k^{(i)} + \arg\min_{t \in \mathbf{R}^{N_i}} \left\{ \langle \nabla_i f(y_k), z - z_k^{(i)} \rangle + \frac{n\theta_k v_i}{2\tau} \|z - z_k^{(i)}\|_{(i)}^2 + \psi_i(z) \right\} \\
&= -z_k^{(i)} + \arg\min_{t \in \mathbf{R}^{N_i}} \left\{ \langle \nabla_i f(y_k), z - y_k^{(i)} \rangle + \frac{n\theta_k v_i}{2\tau} \|z - z_k^{(i)}\|_{(i)}^2 + \psi_i(z) \right\} \\
&= -z_k^{(i)} + z_{k+1}^{(i)},
\end{aligned}$$

whence $\tilde{z}_{k+1}^{(i)} = \tilde{z}_k^{(i)} - z_k^{(i)} + z_{k+1}^{(i)} = z_{k+1}^{(i)}$. Combining the two cases, $i \in S_k$ and $i \notin S_k$, we arrive at

$$\tilde{z}_{k+1} = z_{k+1}. \tag{54}$$

Now looking at the steps of Algorithm 2, we see that

$$u_{k+1} - u_k = -\frac{1 - \frac{n}{\tau}\theta_k}{\theta_k^2}(\tilde{z}_{k+1} - \tilde{z}_k), \tag{55}$$

can thus write

$$\begin{aligned}
\tilde{x}_{k+1} &\overset{(50)}{=} \theta_k^2 u_{k+1} + \tilde{z}_{k+1} \\
&\overset{(55)}{=} \theta_k^2 \left(u_k - \frac{1 - \frac{n}{\tau}\theta_k}{\theta_k^2}(\tilde{z}_{k+1} - \tilde{z}_k) \right) + \tilde{z}_{k+1} \\
&= \theta_k^2 u_k + \tilde{z}_k + \frac{n}{\tau}\theta_k(\tilde{z}_{k+1} - \tilde{z}_k) \\
&\overset{(51)}{=} \tilde{y}_k + \frac{n}{\tau}\theta_k(\tilde{z}_{k+1} - \tilde{z}_k) \\
&\overset{(54)}{=} y_k + \frac{n}{\tau}\theta_k(z_{k+1} - z_k) \\
&= x_{k+1}.
\end{aligned} \tag{56}$$

Finally,

$$\begin{aligned}
\tilde{y}_{k+1} &\overset{(51)}{=} \theta_{k+1}^2 u_{k+1} + \tilde{z}_{k+1} \\
&\overset{(50)}{=} \frac{\theta_{k+1}^2}{\theta_k^2}(\tilde{x}_{k+1} - \tilde{z}_{k+1}) + \tilde{z}_{k+1} \\
&\overset{(34)}{=} (1 - \theta_{k+1})(\tilde{x}_{k+1} - \tilde{z}_{k+1}) + \tilde{z}_{k+1} \\
&\overset{(54)+(56)}{=} (1 - \theta_{k+1})(x_{k+1} - z_{k+1}) + z_{k+1} \\
&= y_{k+1},
\end{aligned}$$

which concludes the proof.