CrossMark

FULL LENGTH PAPER

Parallel coordinate descent methods for big data optimization

Peter Richtárik[1] · **Martin Takáč[1]**

Abstract In this work we show that randomized (block) coordinate descent methods can be accelerated by parallelization when applied to the problem of minimizing the sum of a *partially separable* smooth convex function and a simple separable convex function. The theoretical speedup, as compared to the serial method, and referring to the number of iterations needed to approximately solve the problem with high probability, is a simple expression depending on the number of parallel processors and a natural and easily computable measure of separability of the smooth component of the objective function. In the worst case, when no degree of separability is present, there may be no speedup; in the best case, when the problem is separable, the speedup is equal to the number of processors. Our analysis also works in the mode when the number of blocks being updated at each iteration is random, which allows for modeling situations with busy or unreliable processors. We show that our algorithm is able to solve a LASSO problem involving a matrix with 20 billion nonzeros in 2 h on a large memory node with 24 cores.

Keywords Parallel coordinate descent · Big data optimization · Partial separability · Huge-scale optimization · Iteration complexity ·

✉ Peter Richtárik
  peter.richtarik@ed.ac.uk

[1] School of Mathematics, University of Edinburgh, Edinburgh, UK

🐾 Springer

Expected separable over-approximation · Composite objective · Convex optimization · LASSO

Mathematics Subject Classification 90C06 · 90C25 · 49M20 · 49M27 · 65K05 · 68W10 · 68W20 · 68W40

1 Introduction

1.1 Big data optimization

Recently there has been a surge in interest in the design of algorithms suitable for solving convex optimization problems with a huge number of variables [12,15]. Indeed, the size of problems arising in fields such as machine learning [1], network analysis [29], PDEs [27], truss topology design [16] and compressed sensing [5] usually grows with our capacity to solve them, and is projected to grow dramatically in the next decade. In fact, much of computational science is currently facing the "big data" challenge, and this work is aimed at developing optimization algorithms suitable for the task.

1.2 Coordinate descent methods

Coordinate descent methods (CDM) are one of the most successful classes of algorithms in the big data optimization domain. Broadly speaking, CDMs are based on the strategy of updating a single coordinate (or a single block of coordinates) of the vector of variables at each iteration. This often drastically reduces memory requirements as well as the arithmetic complexity of a single iteration, making the methods easily implementable and scalable. In certain applications, a single iteration can amount to as few as 4 multiplications and additions only [16]! On the other hand, many more iterations are necessary for convergence than it is usual for classical gradient methods. Indeed, the number of iterations a CDM requires to solve a smooth convex optimization problem is $O(\frac{n\tilde{L}R^2}{\epsilon})$, where ϵ is the error tolerance, n is the number variables (or blocks of variables), $\tilde{L}$ is the average of the Lipschitz constants of the gradient of the objective function associated with the variables (blocks of variables) and R is the distance from the starting iterate to the set of optimal solutions. On balance, as observed by numerous authors, serial CDMs are much more efficient for big data optimization problems than most other competing approaches, such as gradient methods [10,16].

1.3 Parallelization

We wish to point out that for truly huge-scale problems it is absolutely necessary to *parallelize*. This is in line with the rise and ever increasing availability of high performance computing systems built around multi-core processors, GPU-accelerators and computer clusters, the success of which is rooted in massive parallelization. This simple observation, combined with the remarkable scalability of serial CDMs, leads

to our belief that the study of parallel coordinate descent methods (PCDMs) is a very timely topic.

1.4 Research idea

The work presented in this paper was motivated by the desire to answer the following question:

> Under what natural and easily verifiable structural assumptions on the objective function does parallelization of a coordinate descent method lead to acceleration?

Our starting point was the following simple observation. Assume that we wish to minimize a separable function F of n variables (i.e., a function that can be written as a sum of n functions each of which depends on a single variable only). For simplicity, in this thought experiment, assume that there are no constraints. Clearly, the problem of minimizing F can be trivially decomposed into n independent univariate problems. Now, if we have n processors/threads/cores, each assigned with the task of solving one of these problems, the number of parallel iterations should not depend on the dimension of the problem. In other words, we get an n-times speedup compared to the situation with a single processor only. Any parallel algorithm of this type can be viewed as a parallel coordinate descent method. Hence, PCDM with n processors should be n-times faster than a serial one. If τ processors are used instead, where $1 \le \tau \le n$, one would expect a τ-times speedup.

By extension, one would perhaps expect that optimization problems with objective functions which are "close to being separable" would also be amenable to acceleration by parallelization, where the acceleration factor τ would be reduced with the reduction of the "degree of separability". One of the main messages of this paper is an affirmative answer to this. Moreover, we give explicit and simple formulae for the speedup factors.

As it turns out, and as we discuss later in this section, many real-world big data optimization problems are, quite naturally, "close to being separable". We believe that this means that PCDMs is a very promising class of algorithms for structured big data optimization problems.

1.5 Minimizing a partially separable composite objective

In this paper we study the problem

$$\text{minimize} \quad \left\{ F(x) \stackrel{\text{def}}{=} f(x) + \Omega(x) \right\} \quad \text{subject to} \quad x \in \mathbf{R}^N, \tag{1}$$

where f is a (block) *partially separable* smooth convex function and Ω is a simple (block) separable convex function. We allow Ω to have values in $\mathbf{R} \cup \{\infty\}$, and for regularization purposes we assume Ω is proper and closed. While (1) is seemingly an unconstrained problem, Ω can be chosen to model simple convex constraints on individual blocks of variables. Alternatively, this function can be used to enforce a certain structure (e.g., sparsity) in the solution. For a more detailed account we refer

the reader to [15]. Further, we assume that this problem has a minimum ($F^* > -\infty$). What we mean by "smoothness" and "simplicity" will be made precise in the next section.

Let us now describe the key concept of partial separability. Let $x \in \mathbf{R}^N$ be decomposed into n non-overlapping blocks of variables $x^{(1)}, \ldots, x^{(n)}$ (this will be made precise in Sect. 2). We assume throughout the paper that $f \colon \mathbf{R}^N \to \mathbf{R}$ is *partially separable of degree* ω, i.e., that it can be written in the form

$$\mathrm{f}(x) = \sum_{J \in \mathcal{J}} f_J(x), \tag{2}$$

where $\mathcal{J}$ is a finite collection of nonempty subsets of $[n] \overset{\text{def}}{=} \{1, 2, \ldots, n\}$ (possibly containing identical sets multiple times), f_J are differentiable convex functions such that f_J depends on blocks $x^{(i)}$ for $i \in J$ only, and

$$|J| \leq \omega \quad \text{for all} \quad J \in \mathcal{J}. \tag{3}$$

Clearly, $1 \leq \omega \leq n$. The PCDM algorithms we develop and analyze in this paper only need to know ω, they do not need to know the decomposition of f giving rise to this ω.

1.6 Examples of partially separable functions

Many objective functions naturally encountered in the big data setting are partially separable. Here we give examples of three loss/objective functions frequently used in the machine learning literature and also elsewhere. For simplicity, we assume all blocks are of size 1 (i.e., $N = n$). Let

$$f(x) = \sum_{j=1}^{m} \mathcal{L}(x, A_j, y_j), \tag{4}$$

where m is the number of examples, $x \in \mathbf{R}^n$ is the vector of features, $(A_j, y_j) \in \mathbf{R}^n \times \mathbf{R}$ are labeled examples and $\mathcal{L}$ is one of the three loss functions listed in Table 1. Let $A \in \mathbf{R}^{m \times n}$ with row j equal to A_j^T.

Often, each example depends on a few features only; the maximum over all features is the degree of partial separability ω. More formally, note that the jth function in the sum (4) in all cases depends on $\|A_j\|_0$ coordinates of x (the number of nonzeros in the jth row of A) and hence f is partially separable of degree

Table 1 Three examples of loss of functions		
	Square loss	$\frac{1}{2}(A_j^T x - y_j)^2$
	Logistic loss	$\log(1 + e^{-y_j A_j^T x})$
	Hinge square loss	$\frac{1}{2}\max\{0, 1 - y_j A_j^T x\}^2$

$$\omega = \max_j \|A_j\|_0.$$

All three functions of Table 1 are smooth (based on the definition of smoothness in the next section). We refer the reader to [13] for more examples of interesting (but nonsmooth) partially separable functions arising in graph cuts and matrix completion.

1.7 Brief literature review

Several papers were written recently studying the iteration complexity of *serial* CDMs of various flavours and in various settings. We will only provide a brief summary here, for a more detailed account we refer the reader to [15].

Classical CDMs update the coordinates in a cyclic order; the first attempt at analyzing the complexity of such a method is due to [21]. Stochastic/randomized CDMs, that is, methods where the coordinate to be updated is chosen randomly, were first analyzed for quadratic objectives [4,24], later independently generalized to L_1-regularized problems [23] and smooth block-structured problems [10], and finally unified and refined in [15,19]. The problems considered in the above papers are either unconstrained or have (block) separable constraints. Recently, randomized CDMs were developed for problems with linearly coupled constraints [7,8].

A greedy CDM for L_1-regularized problems was first analyzed in [16]; more work on this topic include [2,5]. A CDM with inexact updates was first proposed and analyzed in [26]. Partially separable problems were independently studied in [13], where an asynchronous parallel stochastic gradient algorithm was developed to solve them.

When writing this paper, the authors were aware only of the parallel CDM proposed and analyzed in [1]. Several papers on the topic appeared around the time this paper was finalized or after [6,14,22,22,28]. Further papers on various aspects of the topic of parallel CDMs, building on the work in this paper, include [3,17,18,25].

1.8 Contents

We start in Sect. 2 by describing the block structure of the problem, establishing notation and detailing assumptions. Subsequently we propose and comment in detail on two parallel coordinate descent methods. In Sect. 3 we summarize the main contributions of this paper. In Sect. 4 we deal with issues related to the selection of the blocks to be updated in each iteration. It will involve the development of some elementary random set theory. Sections 5 and 6 deal with issues related to the computation of the update to the selected blocks and develop a theory of Expected Separable Overapproximation (ESO), which is a novel tool we propose for the analysis of our algorithms. In Sect. 7 we analyze the iteration complexity of our methods and finally, Sect. 8 reports on promising computational results. For instance, we conduct an experiment with a big data (cca 350GB) LASSO problem with a billion variables. We are able to solve the problem using one of our methods on a large memory machine with 24 cores in 2 h, pushing the difference between the objective value at the starting iterate and the

optimal point from 10^{22} down to 10^{-14}. We also conduct experiments on real data problems coming from machine learning.

2 Parallel block coordinate descent methods

In Sect. 2.1 we formalize the block structure of the problem, establish notation[1] that will be used in the rest of the paper and list assumptions. In Sect. 2.2 we propose two parallel block coordinate descent methods and comment in some detail on the steps.

2.1 Block structure, notation and assumptions

The block structure[2] of (1) is given by a decomposition of $\mathbf{R}^N$ into n subspaces as follows. Let $U \in \mathbf{R}^{N \times N}$ be a column permutation[3] of the $N \times N$ identity matrix and further let $U = [U_1, U_2, \ldots, U_n]$ be a decomposition of U into n submatrices, with U_i being of size $N \times N_i$, where $\sum_i N_i = N$.

Proposition 1 (Block decomposition[4]) *Any vector $x \in \mathbf{R}^N$ can be written uniquely as*

$$x = \sum_{i=1}^{n} U_i x^{(i)}, \tag{5}$$

where $x^{(i)} \in \mathbf{R}^{N_i}$. Moreover, $x^{(i)} = U_i^T x$.

Proof Noting that $UU^T = \sum_i U_i U_i^T$ is the $N \times N$ identity matrix, we have $x = \sum_i U_i U_i^T x$. Let us now show uniqueness. Assume that $x = \sum_i U_i x_1^{(i)} = \sum_i U_i x_2^{(i)}$, where $x_1^{(i)}, x_2^{(i)} \in \mathbf{R}^{N_i}$. Since

$$U_j^T U_i = \begin{cases} N_j \times N_j & \text{identity matrix,} & \text{if } i = j, \\ N_j \times N_i & \text{zero matrix,} & \text{otherwise,} \end{cases} \tag{6}$$

for every j we get $0 = U_j^T (x - x) = U_j^T \sum_i U_i (x_1^{(i)} - x_2^{(i)}) = x_1^{(j)} - x_2^{(j)}$.

---

[1] Table 8 in the appendix summarizes some of the key notation used frequently in the paper.

[2] Some elements of the setup described in this section was initially used in the analysis of block coordinate descent methods by Nesterov [10] (e.g., block structure, weighted norms and block Lipschitz constants).

[3] The reason why we work with a *permutation* of the identity matrix, rather than with the identity itself, as in [10], is to enable the blocks being formed by *nonconsecutive* coordinates of x. This way we establish notation which makes it possible to work with (i.e., analyze the properties of) multiple block decompositions, for the sake of picking the best one, subject to some criteria. Moreover, in some applications the coordinates of x have a natural ordering to which the natural or efficient block structure does not correspond.

[4] This is a straightforeard result; we do not claim any novelty and include it solely for the benefit of the reader.

In view of the above proposition, from now on we write $x^{(i)} \stackrel{\text{def}}{=} U_i^T x \in \mathbf{R}^{N_i}$, and refer to $x^{(i)}$ as the *ith block* of x. The definition of partial separability in the introduction is with respect to these blocks. For simplicity, we will sometimes write $x = (x^{(1)}, \ldots, x^{(n)})$.

2.1.1 Projection onto a set of blocks

For $S \subset [n]$ and $x \in \mathbf{R}^N$ we write

$$x_{[S]} \stackrel{\text{def}}{=} \sum_{i \in S} U_i x^{(i)}. \tag{7}$$

That is, given $x \in \mathbf{R}^N$, $x_{[S]}$ is the vector in $\mathbf{R}^N$ whose blocks $i \in S$ are identical to those of x, but whose other blocks are zeroed out. In view of Proposition 1, we can equivalently define $x_{[S]}$ block-by-block as follows

$$(x_{[S]})^{(i)} = \begin{cases} x^{(i)}, & i \in S, \\ 0 \ (\in \mathbf{R}^{N_i}), & \text{otherwise.} \end{cases} \tag{8}$$

2.1.2 Inner products

The standard Euclidean inner product in spaces $\mathbf{R}^N$ and $\mathbf{R}^{N_i}$, $i \in [n]$, will be denoted by $\langle \cdot, \cdot \rangle$. Letting $x, y \in \mathbf{R}^N$, the relationship between these inner products is given by

$$\langle x, y \rangle \stackrel{(5)}{=} \left\langle \sum_{j=1}^n U_j x^{(j)} \sum_{i=1}^n U_i y^{(i)} \right\rangle = \sum_{j=1}^n \sum_{i=1}^n \langle U_i^T U_j x^{(j)}, y^{(i)} \rangle \stackrel{(6)}{=} \sum_{i=1}^n \langle x^{(i)}, y^{(i)} \rangle.$$

For any $w \in \mathbf{R}^n$ and $x, y \in \mathbf{R}^N$ we further define

$$\langle x, y \rangle_w \stackrel{\text{def}}{=} \sum_{i=1}^n w_i \langle x^{(i)}, y^{(i)} \rangle. \tag{9}$$

For vectors $z = (z_1, \ldots, z_n)^T \in \mathbf{R}^n$ and $w = (w_1, \ldots, w_n)^T \in \mathbf{R}^n$ we write $w \odot z \stackrel{\text{def}}{=} (w_1 z_1, \ldots, w_n z_n)^T$.

2.1.3 Norms

Spaces $\mathbf{R}^{N_i}$, $i \in [n]$, are equipped with a pair of conjugate norms: $\|t\|_{(i)} \stackrel{\text{def}}{=} \langle B_i t, t \rangle^{1/2}$, where B_i is an $N_i \times N_i$ positive definite matrix and $\|t\|_{(i)}^* \stackrel{\text{def}}{=} \max_{\|s\|_{(i)} \leq 1} \langle s, t \rangle = \langle B_i^{-1} t, t \rangle^{1/2}$, $t \in \mathbf{R}^{N_i}$. For $w \in \mathbf{R}^n_{++}$, define a pair of conjugate norms in $\mathbf{R}^N$ by

$$\|x\|_w = \left[\sum_{i=1}^{n} w_i \|x^{(i)}\|_{(i)}^2\right]^{1/2},$$

$$\|y\|_w^* \stackrel{\text{def}}{=} \max_{\|x\|_w \leq 1} \langle y, x \rangle = \left[\sum_{i=1}^{n} w_i^{-1}(\|y^{(i)}\|_{(i)}^*)^2\right]^{1/2}. \tag{10}$$

Note that these norms are induced by the inner product (9) and the matrices $B_1, \ldots, B_n$. Often we will use $w = L \stackrel{\text{def}}{=} (L_1, L_2, \ldots, L_n)^T \in \mathbf{R}^n$, where the constants L_i are defined below.

2.1.4 Smoothness of f

We assume throughout the paper that the gradient of f is block Lipschitz, uniformly in x, with positive constants $L_1, \ldots, L_n$, i.e., that for all $x \in \mathbf{R}^N, i \in [n]$ and $t \in \mathbf{R}^{N_i}$,

$$\|\nabla_i f(x + U_i t) - \nabla_i f(x)\|_{(i)}^* \leq L_i \|t\|_{(i)}, \tag{11}$$

where $\nabla_i f(x) \stackrel{\text{def}}{=} (\nabla f(x))^{(i)} = U_i^T \nabla f(x) \in \mathbf{R}^{N_i}$. An important consequence of (11) is the following standard inequality [9]:

$$f(x + U_i t) \leq f(x) + \langle \nabla_i f(x), t \rangle + \frac{L_i}{2} \|t\|_{(i)}^2. \tag{12}$$

2.1.5 Separability of Ω

We assume that[5] $\Omega : \mathbf{R}^N \rightarrow \mathbf{R} \cup \{+\infty\}$ is (block) separable, i.e., that it can be decomposed as follows:

$$\Omega(x) = \sum_{i=1}^{n} \Omega_i(x^{(i)}), \tag{13}$$

where the functions $\Omega_i : \mathbf{R}^{N_i} \rightarrow \mathbf{R} \cup \{+\infty\}$ are convex and closed.

2.1.6 Strong convexity

In one of our two complexity results (Theorem 18) we will assume that either f or Ω (or both) is strongly convex. A function $\phi : \mathbf{R}^N \rightarrow \mathbf{R} \cup \{+\infty\}$ is strongly convex with respect to the norm $\|\cdot\|_w$ with convexity parameter $\mu_\phi(w) \geq 0$ if for all $x, y \in \text{dom}\,\phi$,

$$\phi(y) \geq \phi(x) + \langle \phi'(x), y - x \rangle + \frac{\mu_\phi(w)}{2} \|y - x\|_w^2, \tag{14}$$

---

[5] For examples of separable and block separable functions we refer the reader to [15]. For instance, $\Omega(x) = \|x\|_1$ is separable and block separable (used in sparse optimization); and $\Omega(x) = \sum_i \|x^{(i)}\|$, where the norms are standard Euclidean norms, is block separable (used in group lasso). One can model block constraints by setting $\Omega_i(x^{(i)}) = 0$ for $x \in X_i$, where X_i is some closed convex set, and $\Omega_i(x^{(i)}) = +\infty$ for $x \notin X_i$.

where $\phi'(x)$ is any subgradient of ϕ at x. The case with $\mu_\phi(w) = 0$ reduces to convexity. Strong convexity of F may come from f or Ω (or both); we write $\mu_f(w)$ (resp. $\mu_\Omega(w)$) for the (strong) convexity parameter of f (resp. Ω). It follows from (14) that

$$\mu_F(w) \geq \mu_f(w) + \mu_\Omega(w). \tag{15}$$

The following characterization of strong convexity will be useful:

$$\phi(\lambda x + (1 - \lambda)y) \leq \lambda\phi(x) + (1 - \lambda)\phi(y) - \frac{\mu_\phi(w)\lambda(1-\lambda)}{2}\|x - y\|_w^2,$$
$$x, y \in \operatorname{dom}\phi, \ \lambda \in [0, 1]. \tag{16}$$

It can be shown using (12) and (14) that $\mu_f(w) \leq \frac{L_i}{w_i}$.

2.2 Algorithms

In this paper we develop and study two generic parallel coordinate descent methods. The main method is PCDM1; PCDM2 is its "regularized" version which explicitly enforces monotonicity. As we will see, both of these methods come in many variations, depending on how Step 3 is performed.

Algorithm 1 Parallel Coordinate Descent Method 1 (PCDM1)

1: Choose initial point $x_0 \in \mathbf{R}^N$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3:   Randomly generate a set of blocks $S_k \subseteq \{1, 2, \ldots, n\}$
4:   $x_{k+1} \leftarrow x_k + (h(x_k))_{[S_k]}$
5: **end for**

Algorithm 2 Parallel Coordinate Descent Method 2 (PCDM2)

1: Choose initial point $x_0 \in \mathbf{R}^N$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3:   Randomly generate a set of blocks $S_k \subseteq \{1, 2, \ldots, n\}$
4:   $x_{k+1} \leftarrow x_k + (h(x_k))_{[S_k]}$
5:   If $F(x_{k+1}) > F(x_k)$, then $x_{k+1} \leftarrow x_k$
6: **end for**

Let us comment on the individual steps of the two methods.

Step 3. At the beginning of iteration k we pick a random set (S_k) of blocks to be updated (in parallel) during that iteration. The set S_k is a realization of a random set-valued mapping $\hat{S}$ with values in $2^{[n]}$ or, more precisely, it the sets S_k are iid random sets with the distribution of $\hat{S}$. For brevity, in this paper we refer to such a mapping by the name *sampling*. We limit our attention to *uniform* samplings, i.e., random sets having the following property: $\mathbf{P}(i \in \hat{S})$ is independent of i. That is, the probability that a block gets selected is the same for all blocks. Although we give an iteration complexity result covering all such samplings (provided that each block has a chance to be updated, i.e.,

$\mathbf{P}(i \in \hat{S}) > 0$), there are interesting subclasses of uniform samplings (such as doubly uniform and nonoverlapping uniform samplings; see Sect. 4) for which we give better results.

Step 4. For $x \in \mathbf{R}^N$ we define[6]

$$h(x) \stackrel{\text{def}}{=} \arg \min_{h \in \mathbf{R}^N} H_{\beta,w}(x,h), \tag{17}$$

where

$$H_{\beta,w}(x,h) \stackrel{\text{def}}{=} f(x) + \langle \nabla f(x), h \rangle + \tfrac{\beta}{2} \|h\|_w^2 + \Omega(x+h), \tag{18}$$

and $\beta > 0$, $w = (w_1, \ldots, w_n)^T \in \mathbf{R}_{++}^n$ are parameters of the method that we will comment on later. Note that in view of (5, 10) and (13), $H_{\beta,w}(x,\cdot)$ is block separable;

$$H_{\beta,w}(x,h) = f(x) + \sum_{i=1}^n \left\{ \langle \nabla_i f(x), h^{(i)} \rangle + \tfrac{\beta w_i}{2} \|h^{(i)}\|_{(i)}^2 + \Omega_i(x^{(i)} + h^{(i)}) \right\}.$$

Consequently, we have $h(x) = (h^{(1)}(x), \ldots, h^{(n)}(x)) \in \mathbf{R}^N$, where

$$h^{(i)}(x) = \arg \min_{t \in \mathbf{R}^{N_i}} \{ \langle \nabla_i f(x), t \rangle + \tfrac{\beta w_i}{2} \|t\|_{(i)}^2 + \Omega_i(x^{(i)} + t) \}.$$

We mentioned in the introduction that besides (block) separability, we require Ω to be "simple". By this we mean that the above optimization problem leading to $h^{(i)}(x)$ is "simple" (e.g., it has a closed-form solution). Recall from (8) that $(h(x_k))_{[S_k]}$ is the vector in $\mathbf{R}^N$ identical to $h(x_k)$ except for blocks $i \notin S_k$, which are zeroed out. Hence, Step 4 of both methods can be written as follows:

$$\text{In parallel for } i \in S_k \text{ do}: \quad x_{k+1}^{(i)} \leftarrow x_k^{(i)} + h^{(i)}(x_k).$$

Parameters β and w depend on f and $\hat{S}$ and stay constant throughout the algorithm. We are not ready yet to explain *why* the update is computed via (17) and (18) because we need technical tools, which will be developed in Sect. 4, to do so. Here it suffices to say that the parameters β and w come from a separable quadratic overapproximation of $\mathbf{E}[f(x + h_{[\hat{S}]})]$, viewed as a function of $h \in \mathbf{R}^N$. Since expectation is involved, we refer to this by the name Expected Separable Overapproximation (ESO). This novel concept, developed in this paper, is one of the main tools of our complexity analysis. Section 5 motivates and formalizes the concept, answers the *why* question, and develops some basic ESO theory.

Section 6 is devoted to the computation of β and w for partially separable f and various special classes of uniform samplings $\hat{S}$. Typically we will have $w_i = L_i$, while β will depend on *easily computable* properties of f and $\hat{S}$. For example, if $\hat{S}$

---

[6] A similar map was used in [10] (with $\Omega \equiv 0$ and $\beta = 1$) and [15] (with $\beta = 1$) in the analysis of *serial* coordinate descent methods in the smooth and composite case, respectively. In loose terms, the novelty here is the introduction of the parameter β and in developing theory which describes what value β should have. Maps of this type are known as composite gradient mapping in the literature, and were introduced in [11].

is chosen as a subset of $[n]$ of cardinality τ, with each subset chosen with the same probability (we say that $\hat{S}$ is τ-nice) then, assuming $n > 1$, we may choose $w = L$ and $\beta = 1 + \frac{(\omega-1)(\tau-1)}{n-1}$, where ω is the degree of partial separability of f. More generally, if $\hat{S}$ is any uniform sampling with the property $|\hat{S}| = \tau$ with probability 1, then we may choose $w = L$ and $\beta = \min\{\omega, \tau\}$. Note that in both cases $w = L$ and that the latter β is always larger than (or equal to) the former one. This means, as we will see in Sect. 7, that we can give better complexity results for the former, more specialized, sampling. We analyze several more options for $\hat{S}$ than the two just described, and compute parameters β and w that should be used with them (for a summary, see Table 4).

Step 5. The reason why, besides PCDM1, we also consider PCDM2, is the following: in some situations we are not able to analyze the iteration complexity of PCDM1 (non-strongly-convex F where monotonicity of the method is not guaranteed by other means than by directly enforcing it by inclusion of Step 5). Let us remark that this issue arises for general Ω only. It does not exist for $\Omega = 0$, $\Omega(\cdot) = \lambda \| \cdot \|_1$ and for Ω encoding simple constraints on individual blocks; in these cases one does not need to consider PCDM2. Even in the case of general Ω we sometimes get monotonicity for free, in which case there is no need to enforce it. Let us stress, however, that we do not recommend implementing PCDM2 as this would introduce too much overhead; in our experience PCDM1 works well even in cases when we can only analyze PCDM2.

3 Smmary of contributions

In this section we summarize the main contributions of this paper (not in order of significance).

1. *Problem generality* We give the first complexity analysis for *parallel* coordinate descent methods for problem (1) in its full generality.
2. *Complexity* We show theoretically (Sect. 7) and numerically (Sect. 8) that PCDM accelerates on its serial counterpart for partially separable problems. In particular, we establish two complexity theorems giving lower bounds on the number of iterations k sufficient for one or both of the PCDM variants (for details, see the precise statements in Sect. 7) to produce a random iterate x_k for which the problem is approximately solved with high probability, i.e., $\mathbf{P}(F(x_k) - F^* \leq \epsilon) \geq 1 - \rho$. The results, summarized in Table 2, hold under the standard assumptions listed in Sect. 2.1 *and* the additional assumption that f, $\hat{S}$, β and w satisfy the following

Table 2 Summary of the main complexity results for PCDM established in this paper

Setting	Complexity	Theorem
Convex f	$\mathcal{O}\left(\frac{\beta n}{\mathbf{E}[|\hat{S}|]} \frac{1}{\epsilon} \log\left(\frac{1}{\rho}\right) \right)$	17
Strongly convex f $\mu_f(w) + \mu_\Omega(w) > 0$	$\frac{n}{\mathbf{E}[|\hat{S}|]} \frac{\beta+\mu_\Omega(w)}{\mu_f(w)+\mu_\Omega(w)} \log\left(\frac{F(x_0)-F^*}{\epsilon\rho}\right)$	18

Table 3 New and known gradient methods obtained as special cases of our general framework

Method	Parameters	Comments
Gradient descent	$n = 1$	[11]
Serial random CDM	$N_i = 1$ for all i and $\mathbf{P}(|\hat{S}| = 1) = 1$	[15]
Serial block random CDM	$N_i \geq 1$ for all i and $\mathbf{P}(|\hat{S}| = 1) = 1$	[15]
Parallel random CDM	$\mathbf{P}(|\hat{S}| > 1) > 0$	NEW
Distributed random CDM	$\hat{S}$ is a distributed sampling	[17][a]

[a] Remark at revision: This work was planned as a follow-up paper at the time of submitting this paper in Nov 2012; but has appeared in the time between than and revision

inequality for all $x, h \in \mathbf{R}^N$:

$$\mathbf{E}[f(x + h_{[\hat{S}]})] \leq f(x) + \frac{\mathbf{E}[|\hat{S}|]}{n}\left(\langle \nabla f(x), h \rangle + \frac{\beta}{2}\|h\|_w^2\right). \tag{19}$$

This inequality, which we call Expected Separable Overapproximation (ESO), is the main new theoretical tool that we develop in this paper for the analysis of our methods (Sects. 4, 5 and 6 are devoted to the development of this theory).

The main observation here is that as the average number of block updates per iteration increases (say, $\hat{\tau} = \mathbf{E}[|\hat{S}|]$), enabled by the utilization of more processors, the leading term in the complexity estimate, $n/\hat{\tau}$, decreases in proportion. However, β will generally grow with $\hat{\tau}$, which has an adverse effect on the speedup. Much of the theory in this paper goes towards producing formulas for β (and w), for partially separable f and various classes of *uniform* samplings $\hat{S}$. Naturally, the ideal situation is when β does not grow with $\hat{\tau}$ at all, or if it only grows very slowly. We show that this is the case for partially separable functions f with small ω. For instance, in the extreme case when f is separable ($\omega = 1$), we have $\beta = 1$ and we obtain linear speedup in $\hat{\tau}$. As ω increases, so does β, depending on the law governing $\hat{S}$. Formulas for β and ω for various samplings $\hat{S}$ are summarized in Table 4.

3. *Algorithm unification* Depending on the choice of the block structure (as implied by the choice of n and the matrices $U_1, \ldots, U_n$) and the way blocks are selected at every iteration (as given by the choice of $\hat{S}$), our framework encodes a *family* of known and new algorithms[7] (see Table 3).

   In particular, PCDM is the first method which "continuously" interpolates between serial coordinate descent and gradient (by manipulating n and/or $\mathbf{E}[|\hat{S}|]$).

4. *Partial separability* We give the first analysis of a coordinate descent type method dealing with a partially separable loss / objective. In order to run the method, we need to know the Lipschitz constants L_i and the degree of partial separability ω. It is crucial that these quantities are often easily computable/predictable in the huge-scale setting. For example, if $f(x) = \frac{1}{2}\|Ax - b\|^2$ and we choose all blocks to be of size 1, then L_i is equal to the squared Euclidean norm of the ith column of A

[7] All the methods are in their proximal variants due to the inclusion of the term Ω in the objective.

Table 4 Values of parameters β and w for various samplings $\hat{S}$

Sampling $\hat{S}$	$\mathbf{E}[|\hat{S}|]$	β	w	ESO monotonic?	Follows from
Uniform	$\mathbf{E}[|\hat{S}|]$	1	$\nu \odot L$	No	Thm 10
Nonoverlapping uniform	$\frac{n}{\tau}$	1	$\gamma \odot L$	Yes	Thm 11
Doubly uniform	$\mathbf{E}[|\hat{S}|]$	$1 + \dfrac{(\omega-1)\left(\frac{\mathbf{E}[|\hat{S}|^2]}{\mathbf{E}[|\hat{S}|]}-1\right)}{\max(1,n-1)}$	L	No	Thm 13
τ-Uniform	τ	$\min\{\omega, \tau\}$	L	Yes	Thm 10
τ-Nice	τ	$1 + \dfrac{(\omega-1)(\tau-1)}{\max(1,n-1)}$	L	No	Thm 12/13
(τ, p_b)-Binomial	τp_b	$1 + \dfrac{p_b(\omega-1)(\tau-1)}{\max(1,n-1)}$	L	No	Thm 13
Serial	1	1	L	Yes	Thm 11/12/13
Fully parallel	n	ω	L	Yes	Thm 11/12/13

Table 5 Convex F: Parallelization speedup factors for DU samplings

The last four expressions are a special case of the first expressions since the corresponding samplings are all doubly uniform. Maximum speedup is naturally obtained by the fully parallel sampling: $\frac{n}{\omega}$

$\hat{S}$	Parallelization speedup factor
Doubly uniform	$\dfrac{\mathbf{E}[|\hat{S}|]}{1+\dfrac{(\omega-1)\left(\left(\mathbf{E}[|\hat{S}|^2]/\mathbf{E}[|\hat{S}|]\right)-1\right)}{\max(1,n-1)}}$
(τ, p_b)-Binomial	$\dfrac{\tau}{\dfrac{1}{p_b}+\dfrac{(\omega-1)(\tau-1)}{\max(1,n-1)}}$
τ-Nice	$\dfrac{\tau}{1+\dfrac{(\omega-1)(\tau-1)}{\max(1,n-1)}}$
Fully parallel	$\dfrac{n}{\omega}$
Serial	1

and ω is equal to the maximum number of nonzeros in a row of A. Many problems in the big data setting have small ω compared to n.

5. *Choice of blocks* To the best of our knowledge, existing randomized strategies for paralleling gradient-type methods (e.g., [1]) assume that $\hat{S}$ (or an equivalent thereof, based on the method) is chosen as a subset of $[n]$ of a fixed cardinality, uniformly at random. We refer to such $\hat{S}$ by the name *nice sampling* in this paper. We relax this assumption and our treatment is hence much more general. In fact, we allow for $\hat{S}$ to be any *uniform* sampling. It is possible to further consider *nonuniform samplings*,[8] but this is beyond the scope of this paper.

   In particular, as a special case, our method allows for a variable number of blocks to be updated throughout the iterations (this is achieved by the introduction of *doubly uniform samplings*). This may be useful in some settings such as when the problem is being solved in parallel by τ unreliable processors each of which computes its update $h^{(i)}(x_k)$ with probability p_b and is busy/down with probability $1 - p_b$ (*binomial sampling*).

   Uniform, doubly uniform, nice, binomial and other samplings are defined, and their properties studied, in Sect. 4.

6. *ESO and formulas for β and w.* In Table 4 we list parameters β and w for which ESO inequality (19) holds. Each row corresponds to a specific sampling $\hat{S}$ (see Sect. 4 for the definitions). The last 5 samplings are special cases of one or more of the first three samplings. Details such as what is ν, γ and "monotonic" ESO are explained in appropriate sections later in the text. When a specific sampling $\hat{S}$ is used in the algorithm to select blocks in each iteration, the corresponding parameters β and w are to be used in the method for the computation of the update (see Eqs. 17 and 18).

   En route to proving the iteration complexity results for our algorithms, we develop a theory of deterministic and expected separable overapproximation (Sects. 5, 6) which we believe is of independent interest, too. For instance, methods based on ESO can be compared favorably to the Diagonal Quadratic Approximation (DQA) approach used in the decomposition of stochastic optimization programs [20].

7. *Parallelization speedup* Our complexity results can be used to derive theoretical *parallelization speedup factors*. For several variants of our method, in case of a

---

[8] Revision note: see [18].

non-strongly convex objective, these are summarized in Table 5 (see Sect 7.1 for the derivations). For instance, in the case when all block are updated at each iteration (we later refer to $\hat{S}$ having this property by the name *fully parallel* sampling), the speedup factor is equal to $\frac{n}{\omega}$. If the problem is separable ($\omega = 1$), the speedup is equal to n; if the problem is not separable ($\omega = n$), there may be no speedup. For strongly convex F the situation is even better; the details are given in Sect. 7.2.

8. *Relationship to existing results* To the best of our knowledge, there are just two papers analyzing a parallel coordinate descent algorithm for convex optimization problems[1,6]. In the first paper all blocks are of size 1, $\hat{S}$ corresponds to what we call in this paper a τ-*nice* sampling (i.e., all sets of τ coordinates are updated at each iteration with equal probability) and hence their algorithm is somewhat comparable to one of the many variants of our general method. While the analysis in [1] works for a restricted range of values of τ, our results hold for all $\tau \in [n]$. Moreover, the authors consider a more restricted class of functions f and the special case $\Omega = \lambda \|x\|_1$, which is simpler to analyze. Lastly, the theoretical speedups obtained in [1], when compared to the serial CDM method, depend on a quantity σ that is hard to compute in big data settings (it involves the computation of an eigenvalue of a huge-scale matrix). Our speedups are expressed in terms of natural and easily computable quantity: the degree ω of partial separability of f. In the setting considered by [1], in which more structure is available, it turns out that ω is an upper bound[9] on σ. Hence, we show that one can develop the theory in a more general setting, and that it is not necessary to compute σ (which may be complicated in the big data setting). The parallel CDM method of the second paper [6] only allows all blocks to be updated at each iteration. Unfortunately, the analysis (and the method) is too coarse as it does not offer any theoretical speedup when compared to its serial counterpart. In the special case when only a single block is updated in each iteration, uniformly at random, our theoretical results specialize to those established in [15].

9. *Computations* We demonstrate that our method is able to solve a LASSO problem involving a matrix with a billion columns and 2 billion rows on a large memory node with 24 cores in 2 h (Sect. 8), achieving a $20\times$ speedup compared to the serial variant and pushing the residual by more than 30 degrees of magnitude. While this is done on an artificial problem under ideal conditions (controlling for small ω), large speedups are possible in real data with ω small relative to n. We also perform additional experiments on real machine learning data sets (e.g., training linear SVMs) to illustrate that the predictions of our theory match reality.

10. *Code* The open source code with an efficient implementation of the algorithm(s) developed in this paper is published here: http://code.google.com/p/ac-dc/.

---

[9] Revision note requested by a reviewer: In the time since this paper was posted to arXiv, a number of follow-up papers were written analyzing parallel coordinate descent methods and establishing connections between a discrete quantity analogous to ω (degree of partial/Nesterov separability) and a spectral quantity analogous to σ (largest eigenvalue of a certain matrix), most notably [3,17]. See also [25], which uses a spectral quantity, which can be directly compared to ω.

4 Block samplings

In Step 3 of both PCDM1 and PCDM2 we choose a random set of blocks S_k to be updated at the current iteration. Formally, S_k is a realization of a *random set-valued mapping* $\hat{S}$ with values in $2^{[n]}$, the collection of subsets of $[n]$. For brevity, in this paper we refer to $\hat{S}$ by the name *sampling*. A sampling $\hat{S}$ is uniquely characterized by the probability mass function

$$\mathbf{P}(S) \stackrel{\text{def}}{=} \mathbf{P}(\hat{S} = S), \quad S \subseteq [n]; \tag{20}$$

that is, by assigning probabilities to all subsets of $[n]$. Further, we let $p = (p_1, \ldots, p_n)^T$, where

$$p_i \stackrel{\text{def}}{=} \mathbf{P}(i \in \hat{S}). \tag{21}$$

In Sect. 4.1 we describe those samplings for which we analyze our methods and in Sect. 4.2 we prove several technical results, which will be useful in the rest of the paper.

4.1 Uniform, doubly uniform and nonoverlapping uniform samplings

A sampling is *proper* if $p_i > 0$ for all blocks i. That is, from the perspective of PCDM, under a proper sampling each block gets updated with a positive probability at each iteration. Clearly, PCDM can not converge for a sampling that is not proper. A sampling $\hat{S}$ is *uniform* if all blocks get updated with the same probability, i.e., if $p_i = p_j$ for all i, j. We show in (33) that, necessarily, $p_i = \frac{\mathbf{E}[|\hat{S}|]}{n}$. Further, we say $\hat{S}$ is *nil* if $\mathbf{P}(\emptyset) = 1$. Note that a uniform sampling is proper if and only if it is not nil.

> All complexity results of this paper are formulated for proper uniform samplings. We give a complexity result covering all such samplings. However, the family of proper uniform samplings is large, with several interesting subfamilies for which we can give better results. We now define these families.

All our iteration complexity results in this paper are for PCDM used with a proper uniform sampling (see Theorems 17 and 18) for which we can compute β and w giving rise to an inequality (we we call "expected separable overapproximation") of the form (43). We derive such inequalities for all proper uniform samplings (Theorem 10) as well as refined results for two special subclasses thereof: doubly uniform samplings (Theorem 13) and nonoverlapping uniform samplings (Theorem 11). We will now give the definitions:

1. *Doubly Uniform (DU) samplings* A DU sampling is one which generates all sets of equal cardinality with equal probability. That is, $\mathbf{P}(S') = \mathbf{P}(S'')$ whenever $|S'| = |S''|$. The name comes from the fact that this definition postulates a different uniformity property, "standard" uniformity is a consequence. Indeed, let us show that a DU sampling is necessarily uniform. Let $q_j = \mathbf{P}(|\hat{S}| = j)$ for $j = 0, 1, \ldots, n$ and note that from the definition we know that whenever S is of cardinality j, we have $\mathbf{P}(S) = q_j / \binom{n}{j}$. Finally, using this we obtain

$$p_i = \sum_{S:i \in S} \mathbf{P}(S) = \sum_{j=1}^{n} \sum_{\substack{S:i \in S \\ |S|=j}} \mathbf{P}(S) = \sum_{j=1}^{n} \sum_{S:i \in S |S|=j} \frac{q_j}{\binom{n}{j}} = \sum_{j=1}^{n} \frac{\binom{n-1}{j-1}}{\binom{n}{j}} q_j$$

$$= \frac{1}{n} \sum_{j=1}^{n} q_j j = \frac{\mathbf{E}[|\hat{S}|]}{n}.$$

It is clear that each DU sampling is uniquely characterized by the vector of probabilities q; its density function is given by

$$\mathbf{P}(S) = q_{|S|}/\binom{n}{|S|}, \quad S \subseteq [n]. \tag{22}$$

2. *Nonoverlapping Uniform (NU) samplings* A NU sampling is one which is uniform and which assigns positive probabilities only to sets forming a partition of $[n]$. Let $S^1, S^2, \ldots, S^l$ be a partition of $[n]$, with $|S^j| > 0$ for all j. The density function of a NU sampling corresponding to this partition is given by

$$\mathbf{P}(S) = \begin{cases} \frac{1}{l}, & \text{if } S \in \{S^1, S^2, \ldots, S^l\}, \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

Note that $\mathbf{E}[|\hat{S}|] = \frac{n}{l}$.

Let us now describe several interesting special cases of DU and NU samplings:

3. *Nice sampling* Fix $1 \leq \tau \leq n$. A τ-nice sampling is a DU sampling with $q_\tau = 1$. *Interpretation:* There are τ processors/threads/cores available. At the beginning of each iteration we choose a set of blocks using a τ-nice sampling (i.e., each subset of τ blocks is chosen with the same probability), and assign each block to a dedicated processor/thread/core. Processor assigned with block i would compute and apply the update $h^{(i)}(x_k)$. This is the sampling we use in our computational experiments.

4. *Independent sampling* Fix $1 \leq \tau \leq n$. A τ-independent sampling is a DU sampling with

$$q_k = \begin{cases} \binom{n}{k} c_k, & k = 1, 2, \ldots, \tau, \\ 0, & k = \tau + 1, \ldots, n, \end{cases}$$

where $c_1 = \left(\frac{1}{n}\right)^\tau$ and $c_k = \left(\frac{k}{n}\right)^\tau - \sum_{i=1}^{k-1} \binom{k}{i} c_i$ for $k \geq 2$.
Interpretation: There are τ processors/threads/cores available. Each processor chooses one of the n blocks, uniformly at random and independently of the other processors. It turns out that the set $\hat{S}$ of blocks selected this way is DU with q as given above. Since in one parallel iteration of our methods each block in $\hat{S}$ is updated exactly once, this means that if two or more processors pick the same block, all but one will be idle. On the other hand, this sampling can be generated extremely easily and in parallel! For $\tau \ll n$ this sampling is a good (and fast)

approximation of the τ-nice sampling. For instance, for $n = 10^3$ and $\tau = 8$ we have $q_8 = 0.9723$, $q_7 = 0.0274$, $q_6 = 0.0003$ and $q_k \approx 0$ for $k = 1, \ldots, 5$.

5. *Binomial sampling* Fix $1 \leq \tau \leq n$ and $0 < p_b \leq 1$. A (τ, p_b)-binomial sampling is defined as a DU sampling with

$$q_k = \binom{\tau}{k} p_b^k (1 - p_b)^k, \quad k = 0, 1, \ldots, \tau. \tag{24}$$

Notice that $\mathbf{E}[|\hat{S}|] = \tau p_b$ and $\mathbf{E}[|\hat{S}|^2] = \tau p_b(1 + \tau p_b - p_b)$.

Interpretation: Consider the following situation with *independent equally unreliable processors*. We have τ processors, each of which is at any given moment available with probability p_b and busy with probability $1 - p_b$, independently of the availability of the other processors. Hence, the number of available processors (and hence blocks that can be updated in parallel) at each iteration is a binomial random variable with parameters τ and p_b. That is, the number of available processors is equal to k with probability q_k.

– *Case 1 (explicit selection of blocks):* We learn that k processors are available *at the beginning* of each iteration. Subsequently, we choose k blocks using a k-nice sampling and "assign one block" to each of the k available processors.
– *Case 2 (implicit selection of blocks):* We choose τ blocks using a τ-nice sampling and assign one to each of the τ processors (we do not know which will be available at the beginning of the iteration). With probability q_k, k of these will send their updates. It is easy to check that the resulting effective sampling of blocks is (τ, p_b)-binomial.

6. *Serial sampling* This is a DU sampling with $q_1 = 1$. Also, this is a NU sampling with $l = n$ and $S^j = \{j\}$ for $j = 1, 2, \ldots, l$. That is, at each iteration we update a single block, uniformly at random. This was studied in [15].
7. *Fully parallel sampling* This is a DU sampling with $q_n = 1$. Also, this is a NU sampling with $l = 1$ and $S^1 = [n]$. That is, at each iteration we update all blocks.

Example 2 (Examples of Samplings) Let $n = 3$.

• Sampling $\hat{S}$ defined by $\mathbf{P}(\hat{S} = \{1\}) = 0.5, \mathbf{P}(\hat{S} = \{2\}) = 0.4$ and $\mathbf{P}(\hat{S} = \{3\}) = 0.1$ is not uniform.
• Sampling $\hat{S}$ defined by $\mathbf{P}(\hat{S} = \{1, 2\}) = 2/3$ and $\mathbf{P}(\hat{S} = \{3\}) = 1/3$ is uniform and NU, but it is not DU (and, particular, it is not τ-nice for any τ).
• Sampling $\hat{S}$ defined by $\mathbf{P}(\hat{S} = \{1, 2\}) = 1/3$, $\mathbf{P}(\hat{S} = \{2, 3\}) = 1/3$ and $\mathbf{P}(\hat{S} = \{3, 1\}) = 1/3$ is 2-nice. Since all τ-nice samplings are DU, it is DU. Since all DU samplings are uniform, it is uniform.
• Sampling $\hat{S}$ defined by $\mathbf{P}(\hat{S} = \{1, 2, 3\}) = 1$ is 3-nice. This is the fully parallel sampling. It is both DU and NU.

The following simple result says that the intersection between the class of DU and NU samplings is very thin. A sampling is called *vacuous* if $\mathbf{P}(\emptyset) > 0$.

Proposition 3 *There are precisely two nonvacuous samplings which are both DU and NU: i) the serial sampling and ii) the fully parallel sampling.*

Proof Assume $\hat{S}$ is nonvacuous, NU and DU. Since $\hat{S}$ is nonvacuous, $\mathbf{P}(\hat{S} = \emptyset) = 0$. Let $S \subset [n]$ be any set for which $\mathbf{P}(\hat{S} = S) > 0$. If $1 < |S| < n$, then there exists $S' \neq S$ of the same cardinality as S having a nonempty intersection with S. Since $\hat{S}$ is doubly uniform, we must have $\mathbf{P}(\hat{S} = S') = \mathbf{P}(\hat{S} = S') > 0$. However, this contradicts the fact that $\hat{S}$ is non-overlapping. Hence, $\hat{S}$ can only generate sets of cardinalities 1 or n with positive probability, but not both. One option leads to the fully parallel sampling, the other one leads to the serial sampling.

4.2 Technical results

For a given sampling $\hat{S}$ and $i, j \in [n]$ we let

$$p_{ij} \stackrel{\text{def}}{=} \mathbf{P}(i \in \hat{S}, j \in \hat{S}) = \sum_{S:\{i,j\} \subset S} \mathbf{P}(S). \tag{25}$$

The following simple result has several consequences which will be used throughout the paper.

Lemma 4 (Sum over a random index set) *Let $\emptyset \neq J \subset [n]$ and $\hat{S}$ be any sampling. If θ_i, $i \in [n]$, and θ_{ij}, for $(i, j) \in [n] \times [n]$ are real constants, then*[10]

$$\mathbf{E}\left[\sum_{i \in J \cap \hat{S}} \theta_i\right] = \sum_{i \in J} p_i \theta_i,$$

$$\mathbf{E}\left[\sum_{i \in J \cap \hat{S}} \theta_i \mid |J \cap \hat{S}| = k\right] = \sum_{i \in J} \mathbf{P}(i \in \hat{S} \mid |J \cap \hat{S}| = k)\theta_i, \tag{26}$$

$$\mathbf{E}\left[\sum_{i \in J \cap \hat{S}} \sum_{j \in J \cap \hat{S}} \theta_{ij}\right] = \sum_{i \in J} \sum_{j \in J} p_{ij} \theta_{ij}. \tag{27}$$

Proof We prove the first statement, proof of the remaining statements is essentially identical:

$$\mathbf{E}\left[\sum_{i \in J \cap \hat{S}} \theta_i\right] \stackrel{(20)}{=} \sum_{S \subset [n]} \left(\sum_{i \in J \cap S} \theta_i\right) \mathbf{P}(S) = \sum_{i \in J} \sum_{S:i \in S} \theta_i \mathbf{P}(S)$$

$$= \sum_{i \in J} \theta_i \sum_{S:i \in S} \mathbf{P}(S) = \sum_{i \in J} p_i \theta_i.$$

$\square$

---

[10] Sum over an empty index set will, for convenience, be defined to be zero.

The consequences are summarized in the next theorem and the discussion that follows.

Theorem 5 *Let $\emptyset \neq J \subset [n]$ and $\hat{S}$ be an arbitrary sampling. Further, let $a, h \in \mathbf{R}^N$, $w \in \mathbf{R}_+^n$ and let g be a block separable function, i.e., $g(x) = \sum_i g_i(x^{(i)})$. Then*

$$\mathbf{E}\left[|J \cap \hat{S}|\right] = \sum_{i \in J} p_i, \tag{28}$$

$$\mathbf{E}\left[|J \cap \hat{S}|^2\right] = \sum_{i \in J} \sum_{j \in J} p_{ij}, \tag{29}$$

$$\mathbf{E}\left[\langle a, h_{[\hat{S}]}\rangle_w\right] = \langle a, h \rangle_{p \odot w}, \tag{30}$$

$$\mathbf{E}\left[\|h_{[\hat{S}]}\|_w^2\right] = \|h\|_{p \odot w}^2, \tag{31}$$

$$\mathbf{E}\left[g(x + h_{[\hat{S}]})\right] = \sum_{i=1}^n \left[p_i g_i(x^{(i)} + h^{(i)}) + (1 - p_i)g_i(x^{(i)})\right]. \tag{32}$$

Moreover, the matrix $P \overset{def}{=} (p_{ij})$ is positive semidefinite.

Proof Noting that $|J \cap \hat{S}| = \sum_{i \in J \cap \hat{S}} 1, |J \cap \hat{S}|^2 = (\sum_{i \in J \cap \hat{S}} 1)^2 = \sum_{i \in J \cap \hat{S}} \sum_{j \in J \cap \hat{S}} 1$, $\langle a, h_{[\hat{S}]}\rangle_w = \sum_{i \in \hat{S}} w_i \langle a^{(i)}, h^{(i)}\rangle, \|h_{[\hat{S}]}\|_w^2 = \sum_{i \in \hat{S}} w_i \|h^{(i)}\|_{(i)}^2$ and

$$g(x + h_{[\hat{S}]}) = \sum_{i \in \hat{S}} g_i(x^{(i)} + h^{(i)}) + \sum_{i \notin \hat{S}} g_i(x^{(i)}) = \sum_{i \in \hat{S}} g_i(x^{(i)} + h^{(i)})$$
$$+ \sum_{i=1}^n g_i(x^{(i)}) - \sum_{i \in \hat{S}} g_i(x^{(i)}),$$

all five identities follow directly by applying Lemma 4. Finally, for any $\theta = (\theta_1, \ldots, \theta_n)^T \in \mathbf{R}^n$,

$$\theta^T P \theta = \sum_{i=1}^n \sum_{j=1}^n p_{ij} \theta_i \theta_j \overset{(27)}{=} \mathbf{E}\left[\left(\sum_{i \in \hat{S}} \theta_i\right)^2\right] \geq 0.$$

□

The above results hold for arbitrary samplings. Let us specialize them, in order of decreasing generality, to uniform, doubly uniform and nice samplings.

- *Uniform samplings.* If $\hat{S}$ is uniform, then from (28) using $J = [n]$ we get

$$p_i = \frac{\mathbf{E}\left[|\hat{S}|\right]}{n}, \qquad i \in [n]. \tag{33}$$

Plugging (33) into (28, 30, 31) and (32) yields

$$\mathbf{E}\left[|J \cap \hat{S}|\right] = \tfrac{|J|}{n}\mathbf{E}[|\hat{S}|], \tag{34}$$

$$\mathbf{E}\left[\langle a, h_{[\hat{S}]}\rangle_w\right] = \tfrac{\mathbf{E}\left[|\hat{S}|\right]}{n}\langle a, h\rangle_w, \tag{35}$$

$$\mathbf{E}\left[\|h_{[\hat{S}]}\|_w^2\right] = \tfrac{\mathbf{E}\left[|\hat{S}|\right]}{n}\|h\|_w^2, \tag{36}$$

$$\mathbf{E}\left[g(x + h_{[\hat{S}]})\right] = \tfrac{\mathbf{E}[|\hat{S}|]}{n}g(x + h) + \left(1 - \tfrac{\mathbf{E}[|\hat{S}|]}{n}\right)g(x). \tag{37}$$

- *Doubly uniform samplings.* Consider the case $n > 1$; the case $n = 1$ is trivial. For doubly uniform $\hat{S}$, p_{ij} is constant for $i \neq j$:

$$p_{ij} = \tfrac{\mathbf{E}[|\hat{S}|^2 - |\hat{S}|]}{n(n-1)}. \tag{38}$$

Indeed, this follows from

$$p_{ij} = \sum_{k=1}^{n} \mathbf{P}(\{i, j\} \subseteq \hat{S} \mid |\hat{S}| = k)\mathbf{P}(|\hat{S}| = k) = \sum_{k=1}^{n} \tfrac{k(k-1)}{n(n-1)}\mathbf{P}(|\hat{S}| = k).$$

Substituting (38) and (33) into (29) then gives

$$\mathbf{E}[|J \cap \hat{S}|^2] = (|J|^2 - |J|)\tfrac{\mathbf{E}[|\hat{S}|^2 - |\hat{S}|]}{n\max\{1, n-1\}} + |J|\tfrac{|\hat{S}|}{n}. \tag{39}$$

- *Nice samplings.* Finally, if $\hat{S}$ is τ-nice (and $\tau \neq 0$), then $\mathbf{E}[|\hat{S}|] = \tau$ and $\mathbf{E}[|\hat{S}|^2] = \tau^2$, which used in (39) gives

$$\mathbf{E}[|J \cap \hat{S}|^2] = \tfrac{|J|\tau}{n}\left(1 + \tfrac{(|J|-1)(\tau-1)}{\max\{1, n-1\}}\right). \tag{40}$$

Moreover, assume that $\mathbf{P}(|J \cap \hat{S}| = k) \neq 0$ (this happens precisely when $0 \leq k \leq |J|$ and $k \leq \tau \leq n - |J| + k$). Then for all $i \in J$,

$$\mathbf{P}(i \in \hat{S} \mid |J \cap \hat{S}| = k) = \frac{\binom{|J|-1}{k-1}\binom{n-|J|}{\tau-k}}{\binom{|J|}{k}\binom{n-|J|}{\tau-k}} = \frac{k}{|J|}.$$

Substituting this into (26) yields

$$\mathbf{E}\left[\sum_{i \in J \cap \hat{S}} \theta_i \;\middle|\; |J \cap \hat{S}| = k\right] = \tfrac{k}{|J|}\sum_{i \in J}\theta_i. \tag{41}$$

5 Expected separable overapproximation

Recall that given x_k, in PCDM1 the next iterate is the random vector $x_{k+1} = x_k + h_{[\hat{S}]}$
for a particular choice of $h \in \mathbf{R}^N$. Further recall that in PCDM2,

$$
x_{k+1} = \begin{cases} x_k + h_{[\hat{S}]}, & \text{if } F(x_k + h_{[\hat{S}]}) \leq F(x_k), \\ x_k, & \text{otherwise,} \end{cases}
$$

again for a particular choice of h. While in Sect. 2 we mentioned how h is computed,
i.e., that h is the minimizer of $H_{\beta,w}(x, \cdot)$ (see Eqs. 17 and 18), we did not explain *why*
is h computed this way. The reason for this is that the tools needed for this were not
yet developed at that point (as we will see, some results from Sect. 4 are needed). In
this section we give an answer to this *why* question.

Given $x_k \in \mathbf{R}^N$, after one step of PCDM1 performed with update h we get
$\mathbf{E}[F(x_{k+1}) \mid x_k] = \mathbf{E}[F(x_k + h_{[\hat{S}]}) \mid x_k]$. On the the other hand, after one step
of PCDM2 we have

$$
\mathbf{E}[F(x_{k+1}) \mid x_k] = \mathbf{E}[\min\{F(x_k + h_{[\hat{S}]}), F(x_k)\} \mid x_k]
$$
$$
\leq \min\{\mathbf{E}[F(x_k + h_{[\hat{S}]}) \mid x_k], F(x_k)\}.
$$

So, for both PCDM1 and PCDM2 the following estimate holds,

$$
\mathbf{E}[F(x_{k+1}) \mid x_k] \leq \mathbf{E}[F(x_k + h_{[\hat{S}]}) \mid x_k]. \tag{42}
$$

A good choice for h to be used in the algorithms would be one minimizing the right
hand side of inequality (42). At the same time, we would like the minimization process
to be decomposable so that the updates $h^{(i)}$, $i \in \hat{S}$, could be computed in parallel.
However, the problem of finding such h is intractable in general even if we do not
require parallelizability. Instead, we propose to construct/compute a "simple" sepa-
rable overapproximation of the right-hand side of (42). Since the overapproximation
will be separable, parallelizability is guaranteed; "simplicity" means that the updates
$h^{(i)}$ can be computed easily (e.g., in closed form).

From now on we replace, for simplicity and w.l.o.g., the random vector x_k by a fixed
deterministic vector $x \in \mathbf{R}^N$. We can thus remove conditioning in (42) and instead
study the quantity $\mathbf{E}[F(x + h_{[\hat{S}]})]$. Further, fix $h \in \mathbf{R}^N$. Note that if we can find $\beta > 0$
and $w \in \mathbf{R}^n_{++}$ such that

$$
\mathbf{E}\left[f\left(x + h_{[\hat{S}]}\right)\right] \leq f(x) + \frac{\mathbf{E}[|\hat{S}|]}{n}\left(\langle \nabla f(x), h \rangle + \frac{\beta}{2}\|h\|_w^2\right), \tag{43}
$$

we indeed find a simple separable overapproximation of $\mathbf{E}[F(x + h_{[\hat{S}]})]$:

$$\mathbf{E}[F(x + h_{[\hat{S}]})] \stackrel{(1)}{=} \mathbf{E}[f(x + h_{[\hat{S}]}) + \Omega(x + h_{[\hat{S}]})]$$

$$\stackrel{(43),(37)}{\leq} f(x) + \frac{\mathbf{E}[|\hat{S}|]}{n}\left(\langle\nabla f(x), h\rangle + \frac{\beta}{2}\|h\|_w^2\right) + \left(1 - \frac{\mathbf{E}[|\hat{S}|]}{n}\right)\Omega(x)$$

$$+ \frac{\mathbf{E}[|\hat{S}|]}{n}\Omega(x + h)$$

$$= \left(1 - \frac{\mathbf{E}[|\hat{S}|]}{n}\right)F(x) + \frac{\mathbf{E}[|\hat{S}|]}{n}H_{\beta,w}(x, h), \tag{44}$$

where we recall from (18) that $H_{\beta,w}(x, h) = f(x) + \langle\nabla f(x), h\rangle + \frac{\beta}{2}\|h\|_w^2 + \Omega(x + h)$.

That is, (44) says that the expected objective value after one parallel step of our methods, if block $i \in \hat{S}$ is updated by $h^{(i)}$, is bounded above by a convex combination of $F(x)$ and $H_{\beta,w}(x, h)$. The natural choice of h is to set

$$h(x) = \arg\min_{h \in \mathbf{R}^N} H_{\beta,w}(x, h). \tag{45}$$

Note that this is precisely the choice we make in our methods. Since $H_{\beta,w}(x, 0) = F(x)$, both PCDM1 and PCDM2 are monotonic in expectation.

The above discussion leads to the following definition.

Definition 6 (*Expected Separable Overapproximation (ESO)*) Let $\beta > 0$, $w \in \mathbf{R}_{++}^n$ and let $\hat{S}$ be a proper uniform sampling. We say that $f : \mathbf{R}^N \to \mathbf{R}$ admits a (β, w)-ESO with respect to $\hat{S}$ if inequality (43) holds for all $x, h \in \mathbf{R}^N$. For simplicity, we write $(f, \hat{S}) \sim ESO(\beta, w)$.

A few remarks:

1. *Inflation* If $(f, \hat{S}) \sim ESO(\beta, w)$, then for $\beta' \geq \beta$ and $w' \geq w$, $(f, \hat{S}) \sim ESO(\beta', w')$.
2. *Reshuffling* Since for any $c > 0$ we have $\|h\|_{cw}^2 = c\|h\|_w^2$, one can "shuffle" constants between β and w as follows:

$$(f, \hat{S}) \sim ESO(c\beta, w) \Leftrightarrow (f, \hat{S}) \sim ESO(\beta, cw), \qquad c > 0. \tag{46}$$

3. *Strong convexity* If $(f, \hat{S}) \sim ESO(\beta, w)$, then

$$\beta \geq \mu_f(w). \tag{47}$$

Indeed, it suffices to take expectation in (14) with y replaced by $x + h_{[\hat{S}]}$ and compare the resulting inequality with (43) (this gives $\beta\|h\|_w^2 \geq \mu_f(w)\|h\|_w^2$, which must hold for all h).

Recall that Step 5 of PCDM2 was introduced so as to explicitly enforce monotonicity into the method as in some situations, as we will see in Sect. 7, we can only analyze a monotonic algorithm. However, sometimes even PCDM1 behaves monotonically (without enforcing this behavior externally as in PCDM2). The following definition captures this.

Definition 7 (*Monotonic ESO*) Assume $(f, \hat{S}) \sim ESO(\beta, w)$ and let $h(x)$ be as in (45). We say that the ESO is *monotonic* if $F(x + (h(x))_{[\hat{S}]}) \leq F(x)$, with probability 1, for all $x \in \operatorname{dom} F$.

5.1 Deterministic separable overapproximation (DSO) of partially separable functions

The following theorem will be useful in deriving ESO for uniform samplings (Sect. 6.1) and nonoverlapping uniform samplings (Sect. 6.2). It will also be useful in establishing monotonicity of some ESOs (Theorems 10 and 11).

Theorem 8 (DSO) *Assume f is partially separable (i.e., it can be written in the form (2)). Letting* $\operatorname{Supp}(h) \stackrel{def}{=} \{i \in [n] : h^{(i)} \neq 0\}$, *for all $x, h \in \mathbf{R}^N$ we have*

$$f(x + h) \leq f(x) + \langle \nabla f(x), h \rangle + \frac{\max_{J \in \mathcal{J}} |J \cap \operatorname{Supp}(h)|}{2} \|h\|_L^2. \quad (48)$$

Proof Let us fix x and define $\phi(h) \stackrel{def}{=} f(x + h) - f(x) - \langle \nabla f(x), h \rangle$. Fixing h, we need to show that $\phi(h) \leq \frac{\theta}{2} \|h\|_L^2$ for $\theta = \max_{J \in \mathcal{J}} \theta^J$, where $\theta^J \stackrel{def}{=} |J \cap \operatorname{Supp}(h)|$. One can define functions ϕ^J in an analogous fashion from the constituent functions f_J, which satisfy

$$\phi(h) = \sum_{J \in \mathcal{J}} \phi^J(h), \quad (49)$$

$$\phi^J(0) = 0, \quad J \in \mathcal{J}. \quad (50)$$

Note that (12) can be written as

$$\phi(U_i h^{(i)}) \leq \frac{L_i}{2} \|h^{(i)}\|_{(i)}^2, \quad i = 1, 2, \ldots, n. \quad (51)$$

Now, since ϕ^J depends on the intersection of J and the support of its argument only, we have

$$\phi(h) \stackrel{(49)}{=} \sum_{J \in \mathcal{J}} \phi^J(h) = \sum_{J \in \mathcal{J}} \phi^J \left(\sum_{i=1}^n U_i h^{(i)} \right) = \sum_{J \in \mathcal{J}} \phi^J \left(\sum_{i \in J \cap \operatorname{Supp}(h)} U_i h^{(i)} \right). \quad (52)$$

The argument in the last expression can be written as a convex combination of $1 + \theta^J$ vectors: the zero vector (with weight $\frac{\theta - \theta^J}{\theta}$) and the θ^J vectors $\{\theta U_i h^{(i)} : i \in J \cap \operatorname{Supp}(h)\}$ (with weights $\frac{1}{\theta}$):

$$\sum_{i \in J \cap \operatorname{Supp}(h)} U_i h^{(i)} = \left(\frac{\theta - \theta^J}{\theta} \times 0 \right) + \left(\frac{1}{\theta} \times \sum_{i \in J \cap \operatorname{Supp}(h)} \theta U_i h^{(i)} \right). \quad (53)$$

Finally, we plug (53) into (52) and use convexity and some simple algebra:

$$
\phi(h) \leq \sum_{J \in \mathcal{J}} \left[\frac{\theta - \theta^J}{\theta} \phi^J(0) + \frac{1}{\theta} \sum_{i \in J \cap \operatorname{Supp}(h)} \phi^J(\theta U_i h^{(i)}) \right]
$$

$$
\overset{(50)}{=} \frac{1}{\theta} \sum_{J \in \mathcal{J}} \sum_{i \in J \cap \operatorname{Supp}(h)} \phi^J(\theta U_i h^{(i)})
$$

$$
= \frac{1}{\theta} \sum_{J \in \mathcal{J}} \sum_{i=1}^{n} \phi^J(\theta U_i h^{(i)}) = \frac{1}{\theta} \sum_{i=1}^{n} \sum_{J \in \mathcal{J}} \phi^J(\theta U_i h^{(i)}) \overset{(49)}{=} \frac{1}{\theta} \sum_{i=1}^{n} \phi(\theta U_i h^{(i)})
$$

$$
\overset{(51)}{\leq} \frac{1}{\theta} \sum_{i=1}^{n} \frac{L_i}{2} \|\theta h^{(i)}\|_{(i)}^2 = \frac{\theta}{2} \|h\|_L^2.
$$

$\square$

Besides the usefulness of the above result in deriving ESO inequalities, it is interesting on its own for the following reasons.

1. *Block Lipschitz continuity of* ∇f The DSO inequality (48) is a generalization of (12) since (12) can be recovered from (48) by choosing h with $\operatorname{Supp}(h) = \{i\}$ for $i \in [n]$.
2. *Global Lipschitz continuity of* ∇f The DSO inequality also says that the gradient of f is Lipschitz with Lipschitz constant ω with respect to the norm $\|\cdot\|_L$:

$$
f(x + h) \leq f(x) + \langle \nabla f(x), h \rangle + \frac{\omega}{2} \|h\|_L^2. \tag{54}
$$

Indeed, this follows from (48) via $\max_{J \in \mathcal{J}} |J \cap \operatorname{Supp}(h)| \leq \max_{J \in \mathcal{J}} |J| = \omega$. For $\omega = n$ this has been shown in [10]; our result for partially separable functions appears to be new.
3. *Tightness of the global Lipschitz constant* The Lipschitz constant ω is "tight" in the following sense: there are functions for which ω cannot be replaced in (54) by any smaller number. We will show this on a simple example. Let $f(x) = \frac{1}{2} \|Ax\|^2$ with $A \in \mathbf{R}^{m \times n}$ (blocks are of size 1). Note that we can write $f(x + h) = f(x) + \langle \nabla f(x), h \rangle + \frac{1}{2} h^T A^T A h$, and that $L = (L_1, \ldots, L_n) = \operatorname{diag}(A^T A)$. Let $D = \operatorname{Diag}(L)$. We need to argue that there exists A for which $\sigma \overset{\text{def}}{=} \max_{h \neq 0} \frac{h^T A^T A h}{\|h\|_L^2} = \omega$. Since we know that $\sigma \leq \omega$ (otherwise (54) would not hold), all we need to show is that there is A and h for which

$$
h^T A^T A h = \omega h^T D h. \tag{55}
$$

Since $f(x) = \sum_{i=1}^{m} (A_j^T x)^2$, where A_j is the jth row of A, we assume that each row of A has at most ω nonzeros (i.e., f is partially separable of degree ω). Let us pick A with the following further properties: a) A is a 0-1 matrix, b) all rows of A have exactly ω ones, c) all columns of A have exactly the same number (k) of ones. Immediate consequences: $L_i = k$ for all i, $D = k I_n$ and $\omega m = kn$. If we let

e_m be the $m \times 1$ vector of all ones and e_n be the $n \times 1$ vector of all ones, and set $h = k^{-1/2} e_n$, then

$$
\begin{aligned}
h^T A^T A h &= \tfrac{1}{k} e_n^T A^T A e_n = \tfrac{1}{k} (\omega e_m)^T (\omega e_m) \\
&= \tfrac{\omega^2 m}{k} = \omega n = \omega \tfrac{1}{k} e_n^T k I_n e_n = \omega h^T D h,
\end{aligned}
$$

establishing (55). Using similar techniques one can easily prove the following more general result: Tightness also occurs for matrices A which in each row contain ω identical nonzero elements (but which can vary from row to row).

6 Expected separable overapproximation (ESO) of partially separable functions

Here we derive ESO inequalities for partially separable smooth functions f and (proper) uniform (Sect. 6.1), nonoverlapping uniform (Sect. 6.2), nice (Sect. 6.3) and doubly uniform (Sect. 6.4) samplings.

6.1 Uniform samplings

Consider an arbitrary proper sampling $\hat{S}$ and let $v = (v_1, \ldots, v_n)^T$ be defined by

$$
v_i \overset{\text{def}}{=} \mathbf{E} \left[\min\{\omega, |\hat{S}|\} \mid i \in \hat{S} \right] = \tfrac{1}{p_i} \sum_{S: i \in S} \mathbf{P}(S) \min\{\omega, |S|\}, \qquad i \in [n].
$$

Lemma 9 *If $\hat{S}$ is proper, then*

$$
\mathbf{E} \left[f(x + h_{[\hat{S}]}) \right] \leq f(x) + \langle \nabla f(x), h \rangle_p + \tfrac{1}{2} \| h \|_{p \odot v \odot L}^2. \tag{56}
$$

Proof Let us use Theorem 8 with h replaced by $h_{[\hat{S}]}$. Note that $\max_{J \in \mathcal{J}} |J \cap \mathrm{Supp}(h_{[\hat{S}]})| \leq \max_{J \in \mathcal{J}} |J \cap \hat{S}| \leq \min\{\omega, |\hat{S}|\}$. Taking expectations of both sides of (48) we therefore get

$$
\begin{aligned}
\mathbf{E} \left[f(x + h_{[\hat{S}]}) \right] &\overset{(48)}{\leq} f(x) + \mathbf{E} \left[\langle \nabla f(x), h_{[\hat{S}]} \rangle \right] + \tfrac{1}{2} \mathbf{E} \left[\min\{\omega, |\hat{S}|\} \| h_{[\hat{S}]} \|_L^2 \right] \\
&\overset{(30)}{=} f(x) + \langle \nabla f(x), h \rangle_p + \tfrac{1}{2} \mathbf{E} \left[\min\{\omega, |\hat{S}|\} \| h_{[\hat{S}]} \|_L^2 \right]. \tag{57}
\end{aligned}
$$

It remains to bound the last term in the expression above. Letting $\theta_i = L_i \| h^{(i)} \|_{(i)}^2$, we have

$$\mathbf{E}\left[\min\{\omega, |\hat{S}|\}\|h_{[\hat{S}]}\|_L^2\right] = \mathbf{E}\left[\sum_{i\in\hat{S}}\min\{\omega, |\hat{S}|\}L_i\|h^{(i)}\|_{(i)}^2\right]$$

$$= \sum_{S\subset[n]}\mathbf{P}(S)\sum_{i\in S}\min\{\omega, |S|\}\theta_i$$

$$= \sum_{i=1}^n \theta_i \sum_{S:i\in S}\min\{\omega, |S|\}\mathbf{P}(S)$$

$$= \sum_{i=1}^n \theta_i\, p_i\, \mathbf{E}\left[\min\{\omega, |\hat{S}|\} \mid i \in \hat{S}\right]$$

$$= \sum_{i=1}^n \theta_i\, p_i\, v_i = \|h\|_{p\odot v\odot L}^2.$$

$\square$

The above lemma will now be used to establish ESO for arbitrary (proper) uniform samplings.

Theorem 10 *If $\hat{S}$ is proper and uniform, then*

$$(f, \hat{S}) \sim ESO(1, v \odot L). \tag{58}$$

If, in addition, $\mathbf{P}(|\hat{S}| = \tau) = 1$ (we say that $\hat{S}$ is τ-uniform), then

$$(f, \hat{S}) \sim ESO(\min\{\omega, \tau\}, L). \tag{59}$$

Moreover, ESO (59) is monotonic.

Proof First, (58) follows from (56) since for a uniform sampling one has $p_i = \mathbf{E}[|\hat{S}|]/n$ for all i. If $\mathbf{P}(|\hat{S}| = \tau) = 1$, we get $v_i = \min\{\omega, \tau\}$ for all i; (59) therefore follows from (58). Let us now establish monotonicity. Using the deterministic separable overapproximation (48) with $h = h_{[\hat{S}]}$,

$$F(x + h_{[\hat{S}]}) \leq f(x) + \langle\nabla f(x), h_{[\hat{S}]}\rangle + \max_{J\in\mathcal{J}}\frac{|J\cap\hat{S}|}{2}\|h_{[\hat{S}]}\|_L^2 + \Omega(x + h_{[\hat{S}]})$$

$$\leq f(x) + \langle\nabla f(x), h_{[\hat{S}]}\rangle + \frac{\beta}{2}\|h_{[\hat{S}]}\|_w^2 + \Omega(x + h_{[\hat{S}]}) \tag{60}$$

$$= f(x) + \sum_{i\in\hat{S}}\underbrace{\left(\langle\nabla f(x), U_i h^{(i)}\rangle + \frac{\beta w_i}{2}\|h^{(i)}\|_{(i)}^2 + \Omega_i(x^{(i)} + h^{(i)})\right)}_{\overset{\text{def}}{=}\kappa_i(h^{(i)})}$$

$$+ \sum_{i\notin\hat{S}}\Omega_i(x^{(i)}). \tag{61}$$

Now let $h(x) = \arg\min_h H_{\beta,w}(x, h)$ and recall that

$$
\begin{aligned}
H_{\beta,w}(x, h) &= f(x) + \langle \nabla f(x), h \rangle + \tfrac{\beta}{2}\|h\|_w^2 + \Omega(x + h) \\
&= f(x) + \sum_{i=1}^{n} \left(\langle \nabla f(x), U_i h^{(i)} \rangle + \tfrac{\beta w_i}{2}\|h^{(i)}\|_{(i)}^2 + \Omega_i(x^{(i)} + h^{(i)}) \right) \\
&= f(x) + \sum_{i=1}^{n} \kappa_i(h^{(i)}).
\end{aligned}
$$

So, by definition, $(h(x))^{(i)}$ minimizes $\kappa_i(t)$ and hence, $(h(x))_{[\hat{S}]}$ (recall (7)) minimizes the upper bound (61). In particular, $(h(x))_{[\hat{S}]}$ is better than a nil update, which immediately gives $F(x + (h(x))_{[\hat{S}]}) \le f(x) + \sum_{i \in \hat{S}} \kappa_i(0) + \sum_{i \notin \hat{S}} \Omega_i(x^{(i)}) = F(x)$.   □

Besides establishing an ESO result, we have just shown that, in the case of τ-uniform samplings with a conservative estimate for β, PCDM1 is monotonic, i.e., $F(x_{k+1}) \le F(x_k)$. In particular, PCDM1 and PCDM2 coincide. We call the estimate $\beta = \min\{\omega, \tau\}$ "conservative" because it can be improved (made smaller) in special cases; e.g., for the τ-nice sampling. Indeed, Theorem 12 establishes an ESO for the τ-nice sampling with the same w ($w = L$), but with $\beta = 1 + \frac{(\omega-1)(\tau-1)}{n-1}$, which is better (and can be much better than) $\min\{\omega, \tau\}$. Other things equal, smaller β directly translates into better complexity. The price for the small β in the case of the τ-nice sampling is the loss of monotonicity. This is not a problem for strongly convex objective, but for merely convex objective this is an issue as the analysis techniques we developed are only applicable to the monotonic method PCDM2 (see Theorem 17).

6.2 Nonoverlapping uniform samplings

Let $\hat{S}$ be a (proper) nonoverlapping uniform sampling as defined in (23). If $i \in S^j$, for some $j \in \{1, 2, \ldots, l\}$, define

$$
\gamma_i \overset{\text{def}}{=} \max_{J \in \mathcal{J}} |J \cap S^j|, \tag{62}
$$

and let $\gamma = (\gamma_1, \ldots, \gamma_n)^T$. Note that, for example, if $\hat{S}$ is the serial uniform sampling, then $l = n$ and $S^j = \{j\}$ for $j = 1, 2, \ldots, l$, whence $\gamma_i = 1$ for all $i \in [n]$. For the fully parallel sampling we have $l = 1$ and $S^1 = \{1, 2, \ldots, n\}$, whence $\gamma_i = \omega$ for all $i \in [n]$.

Theorem 11 *If $\hat{S}$ a nonoverlapping uniform sampling, then*

$$
(f, \hat{S}) \sim ESO(1, \gamma \odot L). \tag{63}
$$

Moreover, this ESO is monotonic.

Proof By Theorem 8, used with h replaced by $h_{[S^j]}$ for $j = 1, 2, \ldots, l$, we get

$$f(x + h_{[S^j]}) \le f(x) + \langle \nabla f(x), h_{[S^j]} \rangle + \max_{J \in \mathcal{J}} \frac{|J \cap S^j|}{2} \|h_{[S^j]}\|_L^2. \tag{64}$$

Since $\hat{S} = S^j$ with probability $\frac{1}{l}$,

$$\mathbf{E}\left[f(x + h_{[\hat{S}]})\right] \overset{(64)}{\le} \frac{1}{l} \sum_{j=1}^{l} \left(f(x) + \langle \nabla f(x), h_{[S^j]} \rangle + \max_{J \in \mathcal{J}} \frac{|J \cap S^j|}{2} \|h_{[S^j]}\|_L^2 \right)$$

$$\overset{(62)}{=} f(x) + \frac{1}{l} \left(\langle \nabla f(x), h \rangle + \frac{1}{2} \sum_{j=1}^{l} \sum_{i \in S^j} \gamma_i L_i \|h^{(i)}\|_{(i)}^2 \right)$$

$$= f(x) + \frac{1}{l} \left(\langle \nabla f(x), h \rangle + \frac{1}{2} \|h\|_{\gamma \odot L}^2 \right),$$

which establishes (63). It now only remains to establish monotonicity. Adding $\Omega(x + h_{[\hat{S}]})$ to (64) with S^j replaced by $\hat{S}$, we get $F(x + h_{[\hat{S}]}) \le f(x) + \langle \nabla f(x), h_{[\hat{S}]} \rangle + \frac{\beta}{2} \|h_{[\hat{S}]}\|_w^2 + \Omega(x + h_{[\hat{S}]})$. From this point on the proof is identical to that in Theorem 10, following Eq. (60). $\qquad\square$

6.3 Nice samplings

In this section we establish an ESO for nice samplings.

Theorem 12 *If $\hat{S}$ is the τ-nice sampling and $\tau \ne 0$, then*

$$(f, \hat{S}) \sim ESO \left(1 + \frac{(\omega - 1)(\tau - 1)}{\max(1, n - 1)}, L \right). \tag{65}$$

Proof Let us fix x and define ϕ and ϕ^J as in the proof of Theorem 8. Since

$$\mathbf{E}\left[\phi(h_{[\hat{S}]})\right] = \mathbf{E}\left[f(x + h_{[\hat{S}]}) - f(x) - \langle \nabla f(x), h_{[\hat{S}]} \rangle\right]$$

$$\overset{(35)}{=} \mathbf{E}\left[f(x + h_{[\hat{S}]})\right] - f(x) - \frac{\tau}{n} \langle \nabla f(x), h \rangle,$$

it now only remains to show that

$$\mathbf{E}\left[\phi(h_{[\hat{S}]})\right] \le \frac{\tau}{2n} \left(1 + \frac{(\omega-1)(\tau-1)}{\max(1,n-1)} \right) \|h\|_L^2. \tag{66}$$

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

$$\mathbf{E}\left[\phi(h_{[\hat{S}]})\right] = \sum_{J \in \mathcal{J}} \mathbf{E}\left[\phi^J(h_{[\hat{S}]})\right] = \sum_{k=0}^{n} \sum_{J \in \mathcal{J}} \mathbf{E}\left[\phi^J(h_{[\hat{S}]}) \mid \eta_J = k\right] \mathbf{P}(\eta_J = k)$$
$$= \sum_{k=0}^{n} \mathbf{P}(\eta_J = k) \sum_{J \in \mathcal{J}} \mathbf{E}\left[\phi^J(h_{[\hat{S}]}) \mid \eta_J = k\right].$$

$$(67)$$

Note that the last identity follows if we assume, without loss of generality, that all sets J have the same cardinality ω (this can be achieved by introducing "dummy" dependencies). Indeed, in such a case $\mathbf{P}(\eta_J = k)$ does not depend on J. Now, for any $k \geq 1$ for which $\mathbf{P}(\eta_J = k) > 0$ (for some J and hence for all), using convexity of ϕ^J, we can now estimate

$$\mathbf{E}\left[\phi^J(h_{[\hat{S}]}) \mid \eta_J = k\right] = \mathbf{E}\left[\phi^J\left(\frac{1}{k} \sum_{i \in J \cap \hat{S}} kU_i h^{(i)}\right) \mid \eta_J = k\right]$$
$$\leq \mathbf{E}\left[\frac{1}{k} \sum_{i \in J \cap \hat{S}} \phi^J\left(kU_i h^{(i)}\right) \mid \eta_J = k\right]$$
$$\overset{(41)}{=} \frac{1}{\omega} \sum_{i \in J} \phi^J\left(kU_i h^{(i)}\right).$$

$$(68)$$

If we now sum the inequalities (68) for all $J \in \mathcal{J}$, we get

$$\sum_{J \in \mathcal{J}} \mathbf{E}\left[\phi^J(h_{[\hat{S}]}) \mid \eta_J = k\right] \overset{(68)}{\leq} \frac{1}{\omega} \sum_{J \in \mathcal{J}} \sum_{i \in J} \phi^J\left(kU_i h^{(i)}\right) = \frac{1}{\omega} \sum_{J \in \mathcal{J}} \sum_{i=1}^{n} \phi^J\left(kU_i h^{(i)}\right)$$
$$= \frac{1}{\omega} \sum_{i=1}^{n} \sum_{J \in \mathcal{J}} \phi^J\left(kU_i h^{(i)}\right) = \frac{1}{\omega} \sum_{i=1}^{n} \phi\left(kU_i h^{(i)}\right)$$
$$\overset{(51)}{\leq} \frac{1}{\omega} \sum_{i=1}^{n} \frac{L_i}{2} \|kh^{(i)}\|_{(i)}^2 = \frac{k^2}{2\omega} \|h\|_L^2.$$

$$(69)$$

Finally, (66) follows after plugging (69) into (67):

$$\mathbf{E}\left[\phi(h_{[\hat{S}]})\right] \leq \sum_{k} \mathbf{P}(\eta_J = k) \frac{k^2}{2\omega} \|h\|_L^2 = \frac{1}{2\omega} \|h\|_L^2 \mathbf{E}[|J \cap \hat{S}|^2]$$
$$\overset{(40)}{=} \frac{\tau}{2n}\left(1 + \frac{(\omega-1)(\tau-1)}{\max(1, n-1)}\right) \|h\|_L^2.$$

$$\square$$

6.4 Doubly uniform samplings

We are now ready, using a bootstrapping argument, to formulate and prove a result covering all doubly uniform samplings.

Theorem 13 *If $\hat{S}$ is a (proper) doubly uniform sampling, then*

$$(f, \hat{S}) \sim ESO\left(1 + \frac{(\omega - 1)\left(\frac{\mathbf{E}[|\hat{S}|^2]}{\mathbf{E}[|\hat{S}|]} - 1\right)}{\max(1, n - 1)}, L\right). \tag{70}$$

Proof Letting $q_k = \mathbf{P}(|\hat{S}| = k)$ and $d = \max\{1, n - 1\}$, we have

$$
\begin{aligned}
\mathbf{E}\left[f(x + h_{[\hat{S}]})\right] &= \mathbf{E}\left[\mathbf{E}\left[f(x + h_{[\hat{S}]}) \mid |\hat{S}|\right]\right] = \sum_{k=0}^{n} q_k \, \mathbf{E}\left[f(x + h_{[\hat{S}]}) \mid |\hat{S}| = k\right] \\
&\overset{(65)}{\leq} \sum_{k=0}^{n} q_k \left[f(x) + \tfrac{k}{n}\left(\langle \nabla f(x), h\rangle + \tfrac{1}{2}\left(1 + \tfrac{(\omega-1)(k-1)}{d}\right) \|h\|_L^2\right)\right] \\
&= f(x) + \tfrac{1}{n}\sum_{k=0}^{n} q_k k \langle \nabla f(x), h\rangle \\
&\quad + \tfrac{1}{2n}\sum_{k=1}^{n} q_k \left[k\left(1 - \tfrac{\omega-1}{d}\right) + k^2 \tfrac{\omega-1}{d}\right] \|h\|_L^2 \\
&= f(x) + \tfrac{\mathbf{E}[|\hat{S}|]}{n}\langle \nabla f(x), h\rangle \\
&\quad + \tfrac{1}{2n}\left(\mathbf{E}[|\hat{S}|]\left(1 - \tfrac{\omega-1}{d}\right) + \mathbf{E}[|\hat{S}|^2]\tfrac{\omega-1}{d}\right) \|h\|_L^2.
\end{aligned}
$$

This theorem could have alternatively been proved by writing $\hat{S}$ as a convex combination of nice samplings and applying Theorem 20.

Note that Theorem 13 reduces to that of Theorem 12 in the special case of a nice sampling, and gives the same result as Theorem 11 in the case of the serial and fully parallel samplings.

7 Iteration complexity

In this section we prove two iteration complexity theorems.[11] The first result (Theorem 17) is for non-strongly-convex F and covers PCDM2 with no restrictions and PCDM1 only in the case when a monotonic ESO is used. The second result (Theorem 18) is for strongly convex F and covers PCDM1 without any monotonicity restrictions. Let us first establish two auxiliary results.

Lemma 14 *For all $x \in \operatorname{dom} F$, $H_{\beta,w}(x, h(x)) \leq \min_{y \in \mathbf{R}^N}\{F(y) + \frac{\beta - \mu_f(w)}{2}\|y - x\|_w^2\}$.*

---

[11] The development is similar to that in [15] for the *serial* block coordinate descent method, in the composite case. However, the results are vastly different.

Proof

$$H_{\beta,w}(x, h(x)) \overset{(17)}{=} \min_{y \in \mathbf{R}^N} H_{\beta,w}(x, y - x)$$

$$= \min_{y \in \mathbf{R}^N} f(x) + \langle \nabla f(x), y - x \rangle$$

$$+ \Omega(y) + \tfrac{\beta}{2} \|y - x\|_w^2$$

$$\overset{(14)}{\leq} \min_{y \in \mathbf{R}^N} f(y) - \tfrac{\mu_f(w)}{2} \|y - x\|_w^2 + \Omega(y) + \tfrac{\beta}{2} \|y - x\|_w^2.$$

$\square$

Lemma 15 *(i) Let x^* be an optimal solution of* (1)*, $x \in \mathrm{dom}\, F$ and let $R = \|x - x^*\|_w$. Then*

$$H_{\beta,w}(x, h(x)) - F^* \leq \begin{cases} \left(1 - \frac{F(x)-F^*}{2\beta R^2}\right)(F(x) - F^*), & \text{if } F(x) - F^* \leq \beta R^2, \\ \tfrac{1}{2}\beta R^2 < \tfrac{1}{2}(F(x) - F^*), & \text{otherwise.} \end{cases}$$
(71)

(ii) If $\mu_f(w) + \mu_\Omega(w) > 0$ and $\beta \geq \mu_f(w)$, then for all $x \in \mathrm{dom}\, F$,

$$H_{\beta,w}(x, h(x)) - F^* \leq \frac{\beta - \mu_f(w)}{\beta + \mu_\Omega(w)}(F(x) - F^*).$$
(72)

Proof Part (i): Since we do not assume strong convexity, we have $\mu_f(w) = 0$, and hence

$$H_{\beta,w}(x, h(x)) \overset{\text{(Lemma 14)}}{\leq} \min_{y \in \mathbf{R}^N} \left\{ F(y) + \tfrac{\beta}{2} \|y - x\|_w^2 \right\}$$

$$\leq \min_{\lambda \in [0,1]} \left\{ F(\lambda x^* + (1 - \lambda)x) + \tfrac{\beta \lambda^2}{2} \|x - x^*\|_w^2 \right\}$$

$$\leq \min_{\lambda \in [0,1]} \left\{ F(x) - \lambda(F(x) - F^*) + \tfrac{\beta \lambda^2}{2} R^2 \right\}.$$

Minimizing the last expression in λ gives $\lambda^* = \min\left\{1, (F(x) - F^*)/(\beta R^2)\right\}$; the result follows. Part (ii): Letting $\mu_f = \mu_f(w)$, $\mu_\Omega = \mu_\Omega(w)$ and $\lambda^* = (\mu_f + \mu_\Omega)/(\beta + \mu_\Omega) \leq 1$, we have

$$H_{\beta,w}(x, h(x)) \overset{\text{(Lemma 14)}}{\leq} \min_{y \in \mathbf{R}^N} \left\{ F(y) + \tfrac{\beta - \mu_f}{2} \|y - x\|_w^2 \right\}$$

$$\leq \min_{\lambda \in [0,1]} \left\{ F(\lambda x^* + (1 - \lambda)x) + \tfrac{(\beta - \mu_f)\lambda^2}{2} \|x - x^*\|_w^2 \right\}$$

$$\overset{(16)+(15)}{\leq} \min_{\lambda \in [0,1]} \left\{ \lambda F^* + (1 - \lambda)F(x) \right.$$

$$\left. - \tfrac{(\mu_f + \mu_\Omega)\lambda(1-\lambda) - (\beta - \mu_f)\lambda^2}{2} \|x - x^*\|_w^2 \right\}$$

$$\leq F(x) - \lambda^*(F(x) - F^*).$$

The last inequality follows from the identity $(\mu_f + \mu_\Omega)(1 - \lambda^*) - (\beta - \mu_f)\lambda^* = 0$. □

We could have formulated part (ii) of the above result using the weaker assumption $\mu_F(w) > 0$, leading to a slightly stronger result. However, we prefer the above treatment as it gives more insight.

7.1 Iteration complexity: convex case

The following lemma will be used to finish off the proof of the complexity result of this section.

Lemma 16 (Theorem 1 in [15]) *Fix $x_0 \in \mathbf{R}^N$ and let $\{x_k\}_{k \geq 0}$ be a sequence of random vectors in $\mathbf{R}^N$ with x_{k+1} depending on x_k only. Let $\phi : \mathbf{R}^N \to \mathbf{R}$ be a nonnegative function and define $\xi_k = \phi(x_k)$. Lastly, choose accuracy level $0 < \epsilon < \xi_0$, confidence level $0 < \rho < 1$, and assume that the sequence of random variables $\{\xi_k\}_{k \geq 0}$ is nonincreasing and has one of the following properties:*

(i) $\mathbf{E}[\xi_{k+1} \mid x_k] \leq (1 - \frac{\xi_k}{c_1})\xi_k$, for all k, where $c_1 > \epsilon$ is a constant,

(ii) $\mathbf{E}[\xi_{k+1} \mid x_k] \leq (1 - \frac{1}{c_2})\xi_k$, for all k such that $\xi_k \geq \epsilon$, where $c_2 > 1$ is a constant.

If property (i) holds and we choose $K \geq 2 + \frac{c_1}{\epsilon}(1 - \frac{\epsilon}{\xi_0} + \log(\frac{1}{\rho}))$, or if property (ii) holds, and we choose $K \geq c_2 \log(\frac{\xi_0}{\epsilon\rho})$, then $\mathbf{P}(\xi_K \leq \epsilon) \geq 1 - \rho$.

This lemma was recently extended in [26] so as to aid the analysis of a *serial* coordinate descent method with *inexact* updates, i.e., with $h(x)$ chosen as an *approximate* rather than exact minimizer of $H_{1,L}(x, \cdot)$ (see (17)). While in this paper we deal with exact updates only, the results can be extended to the inexact case.

Theorem 17 *Assume that $(f, \hat{S}) \sim ESO(\beta, w)$, where $\hat{S}$ is a proper uniform sampling, and let $\alpha = \frac{\mathbf{E}[|\hat{S}|]}{n}$. Choose $x_0 \in \mathrm{dom}\, F$ satisfying*

$$\mathcal{R}_w(x_0, x^*) \stackrel{def}{=} \max_x \{\|x - x^*\|_w \ : \ F(x) \leq F(x_0)\} < +\infty, \tag{73}$$

where x^* is an optimal point of (1). Further, choose target confidence level $0 < \rho < 1$, target accuracy level $\epsilon > 0$ and iteration counter K in any of the following two ways:

(i) $\epsilon < F(x_0) - F^*$ and

$$K \geq 2 + \frac{2\left(\frac{\beta}{\alpha}\right) \max\left\{\mathcal{R}_w^2(x_0, x^*), \frac{F(x_0)-F^*}{\beta}\right\}}{\epsilon}\left(1 - \frac{\epsilon}{F(x_0) - F^*} + \log\left(\frac{1}{\rho}\right)\right), \tag{74}$$

(ii) $\epsilon < \min\{2\left(\frac{\beta}{\alpha}\right)\mathcal{R}_w^2(x_0, x^*), F(x_0) - F^*\}$ and

$$K \geq \frac{2\left(\frac{\beta}{\alpha}\right)\mathcal{R}_w^2(x_0, x^*)}{\epsilon} \log\left(\frac{F(x_0) - F^*}{\epsilon\rho}\right). \tag{75}$$

If $\{x_k\}$, $k \geq 0$, are the random iterates of PCDM (use PCDM1 if the ESO is monotonic, otherwise use PCDM2), then $\mathbf{P}(F(x_K) - F^* \leq \epsilon) \geq 1 - \rho$.

Proof Since either PCDM2 is used (which is monotonic) or otherwise the ESO is monotonic, we must have $F(x_k) \leq F(x_0)$ for all k. In particular, in view of (73) this implies that $\|x_k - x^*\|_w \leq \mathcal{R}_w(x_0, x^*)$. Letting $\xi_k = F(x_k) - F^*$, we have

$$\mathbf{E}[\xi_{k+1} \mid x_k] \overset{(44)}{\leq} (1 - \alpha)\xi_k + \alpha(H_{\beta,w}(x_k, h(x_k)) - F^*)$$

$$\overset{(71)}{\leq} (1 - \alpha)\xi_k + \alpha \max\left\{1 - \frac{\xi_k}{2\beta\|x_k - x^*\|_w^2}, \frac{1}{2}\right\}\xi_k$$

$$= \max\left\{1 - \frac{\alpha\xi_k}{2\beta\|x_k - x^*\|_w^2}, 1 - \frac{\alpha}{2}\right\}\xi_k$$

$$\leq \max\left\{1 - \frac{\alpha\xi_k}{2\beta\mathcal{R}_w^2(x_0, x^*)}, 1 - \frac{\alpha}{2}\right\}\xi_k. \tag{76}$$

Consider case (i) and let $c_1 = 2\frac{\beta}{\alpha} \max\{\mathcal{R}_w^2(x_0, x^*), \frac{\xi_0}{\beta}\}$. Continuing with (76), we then get

$$\mathbf{E}[\xi_{k+1} \mid x_k] \leq (1 - \frac{\xi_k}{c_1})\xi_k$$

for all $k \geq 0$. Since $\epsilon < \xi_0 < c_1$, it suffices to apply Lemma 16(i). Consider now case (ii) and let $c_2 = 2\frac{\beta}{\alpha}\frac{\mathcal{R}_w^2(x_0, x^*)}{\epsilon}$. Observe now that whenever $\xi_k \geq \epsilon$, from (76) we get $\mathbf{E}[\xi_{k+1} \mid x_k] \leq (1 - \frac{1}{c_2})\xi_k$. By assumption, $c_2 > 1$, and hence it remains to apply Lemma 16(ii). $\square$

The important message of the above theorem is that the iteration complexity of our methods in the convex case is $O(\frac{\beta}{\alpha}\frac{1}{\epsilon})$. Note that for the serial method (PCDM1 used with $\hat{S}$ being the serial sampling) we have $\alpha = \frac{1}{n}$ and $\beta = 1$ (see Table 4), and hence $\frac{\beta}{\alpha} = n$. It will be interesting to study the *parallelization speedup factor* defined by

$$\text{Parallelization speedup factor} = \frac{\frac{\beta}{\alpha} \text{ of the serial method}}{\frac{\beta}{\alpha} \text{ of a parallel method}} = \frac{n}{\frac{\beta}{\alpha} \text{ of a parallel method}}. \tag{77}$$

Table 5, computed from the data in Table 4, gives expressions for the parallelization speedup factors for PCDM based on a DU sampling (expressions for 4 special cases are given as well).

The speedup of the serial sampling (i.e., of the algorithm based on it) is 1 as we are comparing it to itself. On the other end of the spectrum is the fully parallel sampling with a speedup of $\frac{n}{\omega}$. If the degree of partial separability is small, then this factor will be high — especially so if n is huge, which is the domain we are interested in. This provides an affirmative answer to the research question stated in italics in the introduction.
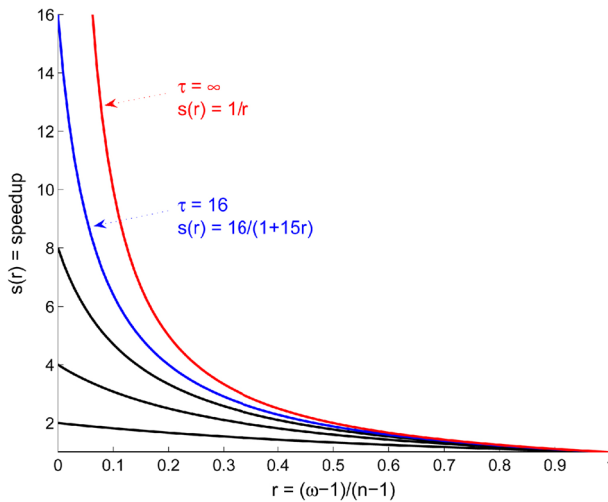
Fig. 1 Parallelization speedup factor of PCDM1/PCDM2 used with τ-nice sampling as a function of the normalized/relative degree of partial separability r

Let us now look at the speedup factor in the case of a τ-nice sampling. Letting $r = \frac{\omega - 1}{\max\{1, n-1\}} \in [0, 1]$ (degree of partial separability normalized), the speedup factor can be written as

$$s(r) = \frac{\tau}{1 + r(\tau - 1)}.$$

Note that as long as $r \leq \frac{k-1}{\tau - 1} \approx \frac{k}{\tau}$, the speedup factor will be at least $\frac{\tau}{k}$. Also note that $\max\{1, \frac{\tau}{\omega}\} \leq s(r) \leq \min\{\tau, \frac{n}{\omega}\}$. Finally, if a speedup of at least s is desired, where $s \in [0, \frac{n}{\omega}]$, one needs to use at least $\frac{1-r}{1/s - r}$ processors. For illustration, in Fig. 1 we plotted $s(r)$ for a few values of τ. Note that for small values of τ, the speedup is significant and can be as large as the number of processors (in the separable case). We wish to stress that in many applications ω will be a constant independent of n, which means that r will indeed be very small in the huge-scale optimization setting.

7.2 Iteration complexity: strongly convex case

In this section we assume that F is strongly convex with respect to the norm $\|\cdot\|_w$ and show that $F(x_k)$ converges to F^* linearly, with high probability.

Theorem 18 *Assume F is strongly convex with $\mu_f(w) + \mu_\Omega(w) > 0$. Further, assume $(f, \hat{S}) \sim ESO(\beta, w)$, where $\hat{S}$ is a proper uniform sampling and let $\alpha = \frac{\mathbf{E}[|\hat{S}|]}{n}$. Choose initial point $x_0 \in \mathrm{dom}\, F$, target confidence level $0 < \rho < 1$, target accuracy level $0 < \epsilon < F(x_0) - F^*$ and*

$$K \geq \frac{1}{\alpha} \frac{\beta + \mu_\Omega(w)}{\mu_f(w) + \mu_\Omega(w)} \log\left(\frac{F(x_0) - F^*}{\epsilon\rho}\right). \tag{78}$$

If $\{x_k\}$ are the random points generated by PCDM1 or PCDM2, then $\mathbf{P}(F(x_K) - F^* \leq \epsilon) \geq 1 - \rho$.

Proof Letting $\xi_k = F(x_k) - F^*$, we have

$$\mathbf{E}[\xi_{k+1} \mid x_k] \overset{(44)}{\leq} (1 - \alpha)\xi_k + \alpha(H_{\beta,w}(x_k, h(x_k)) - F^*)$$
$$\overset{(72)}{\leq} \left(1 - \alpha \frac{\mu_f(w) + \mu_\Omega(w)}{\beta + \mu_\Omega(w)}\right)\xi_k \overset{\text{def}}{=} (1 - \gamma)\xi_k.$$

Note that $0 < \gamma \leq 1$ since $0 < \alpha \leq 1$ and $\beta \geq \mu_f(w)$ by (47). By taking expectation in x_k, we obtain $\mathbf{E}[\xi_k] \leq (1 - \gamma)^k \xi_0$. Finally, it remains to use Markov inequality:

$$\mathbf{P}(\xi_K > \epsilon) \leq \frac{\mathbf{E}[\xi_K]}{\epsilon} \leq \frac{(1 - \gamma)^K \xi_0}{\epsilon} \overset{(78)}{\leq} \rho.$$

$\square$

Instead of doing a direct calculation, we could have finished the proof of Theorem 18 by applying Lemma 16(ii) to the inequality $\mathbf{E}[\xi_{k+1} \mid x_k] \leq (1 - \gamma)\xi_k$. However, in order to be able to use Lemma 16, we would have to first establish monotonicity of the sequence $\{\xi_k\}$, $k \geq 0$. This is not necessary using the direct approach of Theorem 18. Hence, in the strongly convex case we can analyze PCDM1 and are not forced to resort to PCDM2. Consider now the following situations:

1. $\mu_f(w) = 0$. Then the leading term in (78) is $\frac{1 + \beta/\mu_\Omega(w)}{\alpha}$.
2. $\mu_\Omega(w) = 0$. Then the leading term in (78) is $\frac{\beta/\mu_f(w)}{\alpha}$.
3. $\mu_\Omega(w)$ is "large enough". Then $\frac{\beta + \mu_\Omega(w)}{\mu_f(w) + \mu_\Omega(w)} \approx 1$ and the leading term in (78) is $\frac{1}{\alpha}$.

In a similar way as in the non-strongly convex case, define the parallelization speedup factor as the ratio of the leading term in (78) for the serial method (which has $\alpha = \frac{1}{n}$ and $\beta = 1$) and the leading term for a parallel method:

$$\text{Parallelization speedup factor} = \frac{n \frac{1 + \mu_\Omega(w)}{\mu_f(w) + \mu_\Omega(w)}}{\frac{1}{\alpha} \frac{\beta + \mu_\Omega(w)}{\mu_f(w) + \mu_\Omega(w)}} = \frac{n}{\frac{\beta + \mu_\Omega(w)}{\alpha(1 + \mu_\Omega(w))}}. \tag{79}$$

First, note that the speedup factor is independent of μ_f. Further, note that as $\mu_\Omega(w) \to 0$, the speedup factor approaches the factor we obtained in the non-strongly convex case (see (77) and also Table 5). That is, for large values of $\mu_\Omega(w)$, the speedup factor is approximately equal $\alpha n = \mathbf{E}[|\hat{S}|]$, which is the average number of blocks updated in a single parallel iteration. Note that thuis quantity *does not* depend on the degree of partial separability of f.

Fig. 2 Histogram of the cardinalities of the rows of A
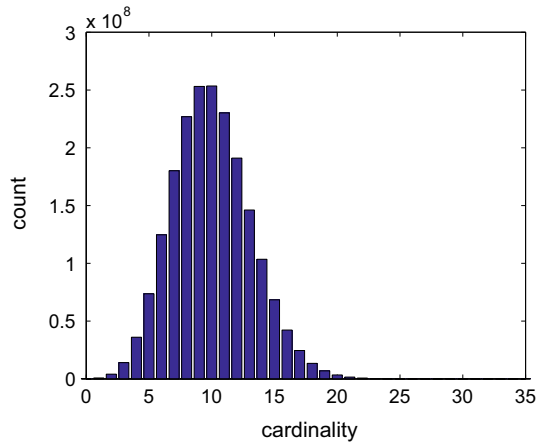
8 Numerical experiments

In Sect. 8.1 we present preliminary but very encouraging results showing that PCDM1 run on a system with 24 cores can solve huge-scale partially-separable LASSO problems with a billion variables in 2 h, compared with 41 h on a single core. In Sect. 8.2 we demonstrate that our analysis is in some sense tight. In particular, we show that the speedup predicted by the theory can be matched almost exactly by actual wall time speedup for a particular problem.

8.1 A LASSO problem with 1 billion variables

In this experiment we solve a single randomly generated huge-scale LASSO instance, i.e., (1) with

$$f(x) = \tfrac{1}{2}\|Ax - b\|_2^2, \qquad \Omega(x) = \|x\|_1,$$

where $A = [a_1, \ldots, a_n]$ has 2×10^9 rows and $N = n = 10^9$ columns. We generated the problem using a modified primal-dual generator [15] enabling us to choose the optimal solution x^* (and hence, indirectly, F^*) and thus to control its cardinality $\|x^*\|_0$, as well as the sparsity level of A. In particular, we made the following choices: $\|x^*\|_0 = 10^5$, each column of A has exactly 20 nonzeros and the maximum cardinality of a row of A is $\omega = 35$ (the degree of partial separability of f). The histogram of cardinalities is displayed in Fig. 2.

We solved the problem using PCDM1 with τ-nice sampling $\hat{S}$, $\beta = 1 + \frac{(\omega-1)(\tau-1)}{n-1}$ and $w = L = (\|a_1\|_2^2, \ldots, \|a_n\|_2^2)$, for $\tau = 1, 2, 4, 8, 16, 24$, on a single large-memory computer utilizing τ of its 24 cores. The problem description took around 350GB of memory space. In fact, in our implementation we departed from the just described setup in two ways. First, we implemented an *asynchronous* version of the method; i.e., one in which cores do not wait for others to update the current iterate within an iteration before reading x_{k+1} and proceeding to another update step. Instead,

each core reads the current iterate whenever it is ready with the previous update step and applies the new update as soon as it is computed. Second, as mentioned in Sect. 4, the τ-independent sampling is for $\tau \ll n$ a very good approximation of the τ-nice sampling. We therefore allowed each processor to pick a block uniformly at random, independently from the other processors.

Choice of the first column of Table 6 In Table 6 we show the development of the gap $F(x_k) - F^*$ as well as the elapsed time. The choice and meaning of the first column of the table, $\frac{\tau k}{n}$, needs some commentary. Note that exactly τk coordinate updates are performed after k iterations. Hence, the first column denotes the total number of coordinate updates normalized by the number of coordinates n. As an example, let $\tau_1 = 1$ and $\tau_2 = 24$. Then if the serial method is run for $k_1 = 24$ iterations and the parallel one for $k_2 = 1$ iteration, both methods would have updated the same number ($\tau_1 k_1 = \tau_2 k_2 = 24$) of coordinates; that is, they would "be" in the same row of Table 6. In summary, each row of the table represents, in the sense described above, the "same amount of work done" for each choice of τ. We have highlighted in bold elapsed time after 13 and 26 passes over data for $\tau = 1, 2, 4, 8, 16$. Note that for any fixed τ, the elapsed time has approximately doubled, as one would expect. More importantly, note that we can clearly observe close to linear speedup in the number of processors τ.

8.1.1 Progress to solving the problem

One can conjecture that the above meaning of the phrase "same amount of work done" would perhaps be roughly equivalent to a different one: "same progress to solving the problem". Indeed, it turns out, as can be seen from the table and also from Fig. 3a, that in each row for all algorithms the value of $F(x_k) - F^*$ is roughly of the same order of magnitude. This is not a trivial finding since, with increasing τ, older information is used to update the coordinates, and hence one would expect that convergence would be slower. It does seem to be slower—the gap $F(x_k) - F^*$ is generally higher if more processors are used—but the slowdown is limited. Looking at Table 6 and/or Fig. 3a, we see that for all choices of τ, PCDM1 managed to push the gap below 10^{-13} after $34n$ to $37n$ coordinate updates.

The progress to solving the problem during the final 1 billion coordinate updates (i.e., when moving from the last-but-one to the last nonempty line in each of the columns of Table 6 showing $F(x_k) - F^*$) is remarkable. The method managed to push the optimality gap by 9–12 degrees of magnitude. We do not have an explanation for this phenomenon; we do not give local convergence estimates in this paper. It is certainly the case though that once the method managed to find the nonzero places of x^*, fast local convergence comes in.

8.1.2 Parallelization speedup

Since a parallel method utilizing τ cores manages to do the same number of coordinate updates as the serial one τ times faster, a direct consequence of the above observation is that doubling the number of cores corresponds to roughly halving the number of iterations (see Fig. 3b. This is due to the fact that $\omega \ll n$ and $\tau \ll n$. It turns out that the num-

Table 6 A LASSO problem with 10^9 variables solved by PCDM1 with $\tau = 1, 2, 4, 8, 16$ and 24

$\frac{\tau k}{n}$	$F(x_k) - F^*$						Elapsed time					
	$\tau = 1$	$\tau = 2$	$\tau = 4$	$\tau = 8$	$\tau = 16$	$\tau = 24$	$\tau = 1$	$\tau = 2$	$\tau = 4$	$\tau = 8$	$\tau = 16$	$\tau = 24$
0	6.27e+22	6.27e+22	6.27e+22	6.27e+22	6.27e+22	6.27e+22	0.00	0.00	0.00	0.00	0.00	0.00
1	2.24e+22	2.24e+22	2.24e+22	2.24e+22	2.24e+22	2.24e+22	0.89	0.43	0.22	0.11	0.06	0.05
2	2.24e+22	2.24e+22	2.24e+22	3.64e+19	2.24e+22	8.13e+18	1.97	1.06	0.52	0.27	0.14	0.10
3	1.15e+20	2.72e+19	8.37e+19	1.94e+19	1.37e+20	5.74e+18	3.20	1.68	0.82	0.43	0.21	0.16
4	5.25e+19	1.45e+19	2.22e+19	1.42e+18	8.19e+19	5.06e+18	4.28	2.28	1.13	0.58	0.29	0.22
5	1.59e+19	2.26e+18	1.13e+19	1.05e+17	3.37e+19	3.14e+18	5.37	2.91	1.44	0.73	0.37	0.28
6	1.97e+18	4.33e+16	1.11e+19	1.17e+16	1.33e+19	3.06e+18	6.64	3.53	1.75	0.89	0.45	0.34
7	2.40e+16	2.94e+16	7.81e+18	3.18e+15	8.39e+17	3.05e+18	7.87	4.15	2.06	1.04	0.53	0.39
8	5.13e+15	8.18e+15	6.06e+18	2.19e+14	5.81e+16	9.22e+15	9.15	4.78	2.37	1.20	0.61	0.45
9	8.90e+14	7.87e+15	2.09e+16	2.08e+13	2.24e+16	5.63e+15	10.43	5.39	2.67	1.35	0.69	0.51
10	5.81e+14	6.52e+14	7.75e+15	3.42e+12	2.89e+15	2.20e+13	11.73	6.02	2.98	1.51	0.77	0.57
11	5.13e+14	1.97e+13	2.55e+15	1.54e+12	2.55e+15	7.30e+12	12.81	6.64	3.29	1.66	0.84	0.63
12	5.04e+14	1.32e+13	1.84e+13	2.18e+11	2.12e+14	1.44e+12	14.08	7.26	3.60	1.83	0.92	0.68
13	2.18e+12	7.06e+11	6.31e+12	1.33e+10	1.98e+14	6.37e+11	**15.35**	**7.88**	**3.91**	**1.99**	**1.00**	0.74
14	7.77e+11	7.74e+10	3.10e+12	3.43e+09	1.89e+12	1.20e+10	16.65	8.50	4.21	2.14	1.08	0.80
15	1.80e+10	6.23e+10	1.63e+11	1.60e+09	5.29e+11	4.34e+09	17.94	9.12	4.52	2.30	1.16	0.86

Table 6 continued

$\frac{\tau k}{n}$	$F(x_k) - F^*$						Elapsed time					
	$\tau = 1$	$\tau = 2$	$\tau = 4$	$\tau = 8$	$\tau = 16$	$\tau = 24$	$\tau = 1$	$\tau = 2$	$\tau = 4$	$\tau = 8$	$\tau = 16$	$\tau = 24$
16	1.38e+09	2.27e+09	7.86e+09	1.15e+09	1.46e+11	1.38e+09	19.23	9.74	4.83	2.45	1.24	0.91
17	3.63e+08	3.99e+08	3.07e+09	6.47e+08	2.92e+09	7.06e+08	20.49	10.36	5.14	2.61	1.32	0.97
18	2.10e+08	1.39e+08	2.76e+08	1.88e+08	1.17e+09	5.93e+08	21.76	10.98	5.44	2.76	1.39	1.03
19	3.81e+07	1.92e+07	7.47e+07	1.55e+06	6.51e+08	5.38e+08	23.06	11.60	5.75	2.91	1.47	1.09
20	1.27e+07	1.59e+07	2.93e+07	6.78e+05	5.49e+07	8.44e+06	24.34	12.22	6.06	3.07	1.55	1.15
21	4.69e+05	2.65e+05	8.87e+05	1.26e+05	3.84e+07	6.32e+06	25.42	12.84	6.36	3.22	1.63	1.21
22	1.47e+05	1.16e+05	1.83e+05	2.62e+04	3.09e+06	1.41e+05	26.64	13.46	6.67	3.38	1.71	1.26
23	5.98e+04	7.24e+03	7.94e+04	1.95e+04	5.19e+05	6.09e+04	27.92	14.08	6.98	3.53	1.79	1.32
24	3.34e+04	3.26e+03	5.61e+04	1.75e+04	3.03e+04	5.52e+04	29.21	14.70	7.28	3.68	1.86	1.38
25	3.19e+04	2.54e+03	2.17e+03	5.00e+03	6.43e+03	4.94e+04	30.43	15.32	7.58	3.84	1.94	1.44
26	3.49e+02	9.62e+01	1.57e+03	4.11e+01	3.68e+03	4.91e+04	**31.71**	**15.94**	**7.89**	**3.99**	**2.02**	1.49
27	1.92e+02	8.38e+01	6.23e+01	5.70e+00	7.77e+02	4.90e+04	33.00	16.56	8.20	4.14	2.10	1.55
28	1.07e+02	2.37e+01	2.38e+01	2.14e+00	6.69e+02	4.89e+04	34.23	17.18	8.49	4.30	2.17	1.61
29	6.18e+00	1.35e+00	1.52e+01	2.35e-01	3.64e+01	4.89e+04	35.31	17.80	8.79	4.45	2.25	1.67
30	4.31e+00	3.93e-01	6.25e-01	4.03e-02	2.74e+00	3.15e+01	36.60	18.43	9.09	4.60	2.33	1.73
31	6.17e-01	3.19e-01	1.24e-01	3.50e-02	6.20e-01	9.29e+00	37.90	19.05	9.39	4.75	2.41	1.78
32	1.83e-02	3.06e-01	3.25e-02	2.41e-03	2.34e-01	3.10e-01	39.17	19.67	9.69	4.91	2.48	1.84
33	3.80e-03	1.75e-03	1.55e-02	1.63e-03	1.57e-02	2.06e-02	40.39	20.27	9.99	5.06	2.56	1.90
34	7.28e-14	7.28e-14	1.52e-02	7.46e-14	1.20e-02	1.58e-02	41.47	20.89	10.28	5.21	2.64	1.96
35	–	–	1.24e-02	–	1.23e-03	8.70e-14	–	–	10.58	–	2.72	2.02
36	–	–	2.70e-03	–	3.99e-04	–	–	–	10.88	–	2.80	–
37	–	–	7.28e-14	–	7.46e-14	–	–	–	11.19	–	2.87	–
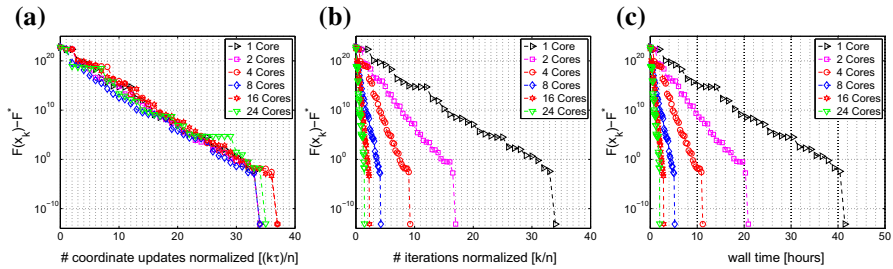
Fig. 3 Computational insights into the workings of PCDM1. **a** For each τ, PCDM1 needs roughly the same number of coordinate updates to solve the problem, **b** doubling the number of cores corresponds to roughly halving the number of iterations, **c** doubling the number of cores corresponds to roughly halving the wall time

Fig. 4 Theoretical speedup factor predicts the actual speedup almost exactly for a carefully constructed problem
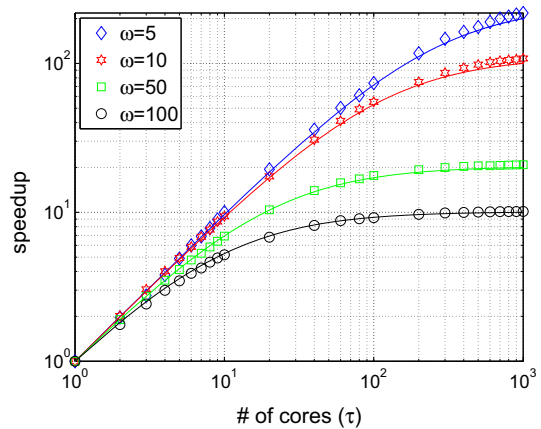
ber of iterations is an excellent predictor of wall time; this can be seen by comparing Fig. 3b, c. Finally, it follows from the above, and can be seen in Fig. 3d, that the speedup of PCDM1 utilizing τ cores is roughly equal to τ. Note that this is caused by the fact that the problem is, relative to its dimension, partially separable to a very high degree.

8.2 Theory versus reality

In our second experiment we demonstrate numerically that our parallelization speedup estimates are in some sense tight. For this purpose it is not necessary to reach for complicated problems and high dimensions; we hence minimize the function $\frac{1}{2}\|Ax - b\|_2^2$ with $A \in \mathbf{R}^{3000 \times 1000}$. Matrix A was generated so that its every row contains exactly ω non-zero values all of which are equal (recall the construction in point 3 at the end of Sect. 5.1).

We generated 4 matrices with $\omega = 5, 10, 50$ and 100 and measured the number of iterations needed for PCDM1 used with τ-nice sampling to get within $\epsilon = 10^{-6}$ of the optimal value. The experiment was done for a range of values of τ (between 1 core and 1000 cores).

The solid lines in Fig. 4 present the *theoretical speedup factor* for the τ-nice sampling, as presented in Table 5. The markers in each case correspond to *empirical speedup factor* defined as

$$\frac{\text{\# of iterations till } \epsilon\text{-solution is found by PCDM1 used with serial sampling}}{\text{\# of iterations till } \epsilon\text{-solution is found by PCDM1 used with } \tau\text{-nice sampling}}.$$

As can be seen in Fig. 4, the match between theoretical prediction and reality is remarkable! A partial explanation of this phenomenon lies in the fact that we have carefully designed the problem so as to ensure that the degree of partial separability is equal to the Lipschitz constant σ of ∇f (i.e., that it is not a gross overestimation of it; see Sect. 5.1). This fact is useful since it is possible to prove complexity results with ω replaced by σ. However, this answer is far from satisfying, and a deeper understanding of the phenomenon remains an open problem.

8.3 Training linear SVMs with bad data for PCDM

In this experiment we test PCDM on the problem of training a linear Support Vector Machine (SVM) based on n labeled training examples: $(y_i, A_i) \in \{+1, -1\} \times \mathbf{R}^d$, $i = 1, 2, \ldots, n$. In particular, we consider the primal problem of minimizing L2-regularized average hinge-loss,

$$\min_{w \in \mathbf{R}^d} \left\{ g(w) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} [1 - y_i \langle w, a_i \rangle]_+ + \frac{\lambda}{2} \|w\|_2^2 \right\},$$

and the dual problem of maximizing a concave quadratic subject to zero-one box constraints,

$$\max_{x \in \mathbf{R}^n, \, 0 \le x^{(i)} \le 1} \left\{ -f(x) \stackrel{\text{def}}{=} -\frac{1}{2\lambda n^2} x^T Z x + \frac{1}{n} \sum_{i=1}^{n} x^{(i)} \right\},$$

where $Z \in \mathbf{R}^{n \times n}$ with $Z_{ii} = y_i y_j \langle A_i, A_j \rangle$. It is a standard practice to apply *serial* coordinate descent to the dual. Here we apply *parallel* coordinate descent (PCDM; with τ-nice sampling of coordinates) to the dual; i.e., minimize the convex function f subject to box constraints. In this setting all blocks are of size $N_i = 1$. The dual can be written in the form (1), i.e.,

$$\min_{x \in \mathbf{R}^n} \{ F(x) = f(x) + \Omega(x) \},$$

where $\Omega(x) = 0$ whenever $x^{(i)} \in [0, 1]$ for all $i = 1, 2, \ldots, n$, and $\Omega(x) = +\infty$ otherwise.

We consider the `rcv1.binary` dataset.[12] The training data has $n = 677,399$ examples, $d = 47,236$ features, $49,556,258$ nonzero elements and requires cca 1GB of RAM for storage. Hence, this is a small-scale problem. The degree of partial separability of f is $\omega = 291,516$ (i.e., the maximum number of examples sharing a given feature). This is a very large number relative to n, and hence our theory would predict rather bad behavior for PCDM. We use PCDM1 with τ-nice sampling (approximating it by τ-independent sampling for added efficiency) with β following Theorem 12: $\beta = 1 + \frac{(\tau-1)(\omega-1)}{n-1}$.

The results of our experiments are summarized in Fig. 5. Each column corresponds to a different level of regularization: $\lambda \in \{1, 10^{-3}, 10^{-5}\}$. The rows show the (1) duality gap, (2) dual suboptimality, (3) train error and (4) test error; each for 1,4 and 16 processors ($\tau = 1, 4, 16$). Observe that the plots in the first two rows are nearly identical; which means that the method is able to solve the primal problem at about the same speed as it can solve the dual problem.[13]

Observe also that in all cases, duality gap of around 0.01 is sufficient for training as training error (classification performance of the SVM on the train data) does not decrease further after this point. Also observe the effect of λ on training accuracy: accuracy increases from about 92 % for $\lambda = 1$, through 95.3 % for $\lambda = 10^{-3}$ to above 97.8 % with $\lambda = 10^{-5}$. In our case, choosing smaller λ does not lead to overfitting; the test error on test dataset (# features =677,399, # examples = 20,242) increases as λ decreases, quickly reaching about 95 % (after 2 seconds of training) for $\lambda = 0.001$ and for the smallest λ going beyond 97 %.

Note that PCDM with $\tau = 16$ is about $2.5\times$ faster than PCDM with $\tau = 1$. This is much less than linear speedup, but is fully in line with our theoretical predictions. Indeed, for $\tau = 16$ we get $\beta = 7.46$. Consulting Table 5, we see that the theory says that with $\tau = 16$ processors we should expect the parallelization speedup to be $PSF = \tau/\beta = 2.15$.

### 8.4 $L2$-regularized logistic regression with good data for PCDM

In our last experiment we solve a problem of the form (1) with f being a sum of logistic losses and Ω being an L2 regularizer,

$$\min_{x \in \mathbf{R}^n} \left\{ \sum_{j=1}^{d} \log(1 + e^{-y_j A_j^T x}) + \lambda \|x\|_2^2 \right\},$$

where $(y_j, A_j) \in \{+1, -1\} \times \mathbf{R}^n$, $j = 1, 2, \ldots, d$, are labeled examples. We have used the the KDDB dataset from the same source as the `rcv1.binary` dataset considered in the previous experiment. The data contains $n = 29,890,095$ features and is divided into two parts: a training set with $d = 19,264,097$ examples (and $566,345,888$ nonzeros; cca 8.5 GB) and a testing with $d = 748,401$ examples (and $21,965,075$ nonzeros; cca 0.32 GB). This training dataset is good for PCDM as each example

---

[12] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#rcv1.binary.

[13] Revision comment: We did not propose primal-dual versions of PCDM in this paper, but we do so in the follow up work [25]. In this paper, for the SVM problem, our methods and theory apply to the dual only.
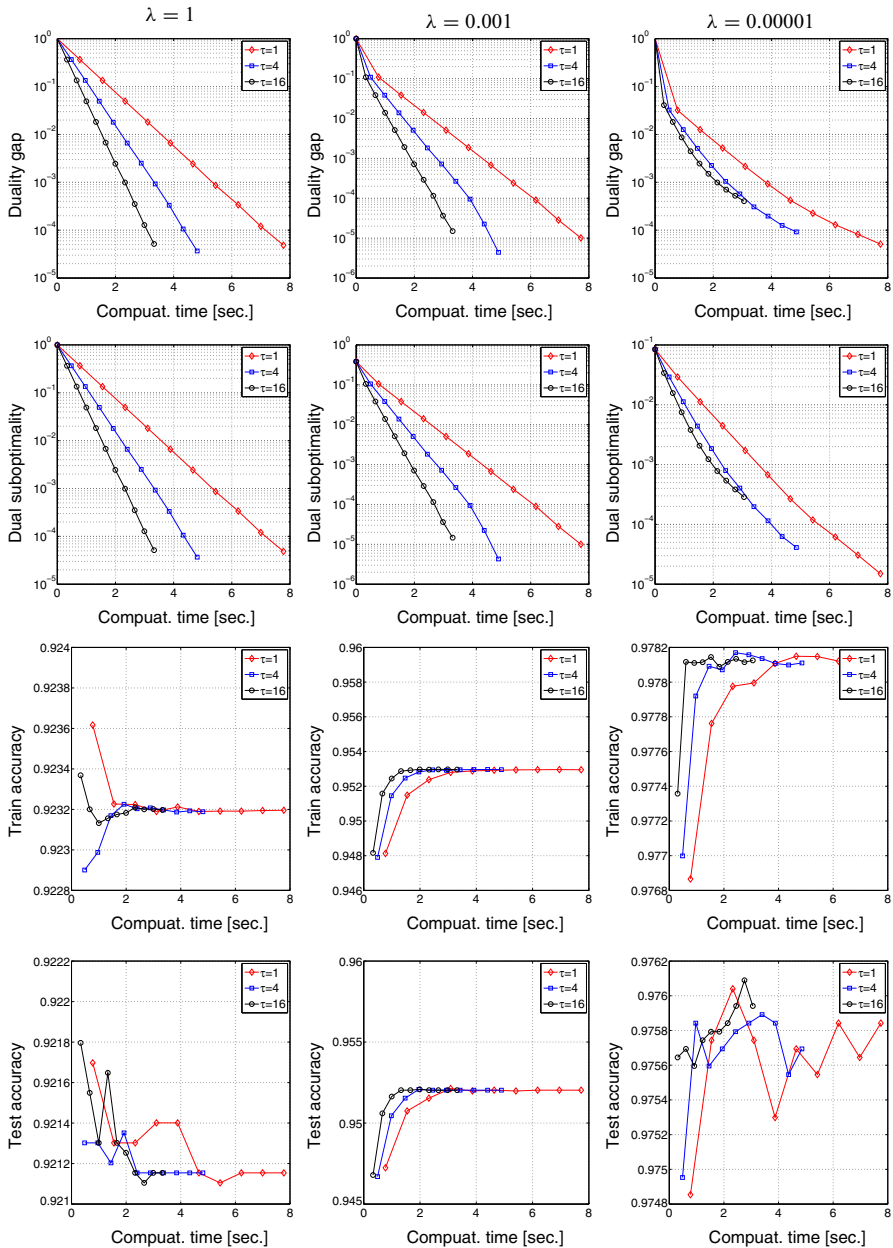
Fig. 5 The performance of PCDM on the rcv1 dataset (this dataset is *not* good for the method)

depends on at most 75 features ($\omega = 75 \ll n$). As before, we will use PCDM1 with τ-nice sampling (approximated by τ-independent sampling) for $\tau = 1, 2, 4, 8$ and set $\lambda = 1$.

Fig. 6 PCDM accelerates well
with more processors on a
dataset with small ω
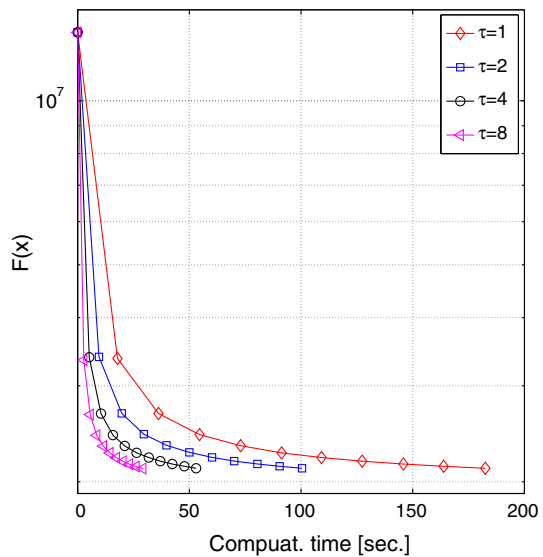
Table 7 PCDM accelerates linearly in τ on a good dataset

Epoch	$F(x_0)/F(x_k)$				Time			
	$\tau = 1$	$\tau = 2$	$\tau = 4$	$\tau = 8$	$\tau = 1$	$\tau = 2$	$\tau = 4$	$\tau = 8$
1	3.96490	3.93909	3.94578	3.99407	17.83	9.57	5.20	2.78
2	5.73498	5.72452	5.74053	5.74427	73.00	39.77	21.11	11.54
3	6.12115	6.11850	6.12106	6.12488	127.35	70.13	37.03	20.29

Figure 6 depicts the evolution of the regularized loss $F(x_k)$ throughout the run of
the 4 versions of PCDM (starting with x_0 for which $F(x_0) = 13,352,855$). Each
marker corresponds to approximately $n/3$ coordinate updates (n coordinate updates
will be referred to as an "epoch"). Observe that as more processors are used, it takes
less time to achieve any given level of loss; nearly in exact proportion to the increase
in the number of processors.

Table 7 offers an alternative view of the same experiment. In the first 4 columns
($F(x_0)/F(x_k)$) we can see that no matter how many processors are used, the methods
produce similar loss values after working through the same number of coordinates.
However, since the method utilizing $\tau = 8$ processors updates 8 coordinates in parallel,
it does the job approximately 8 times faster.

Let us remark that the training and testing accuracy stopped increasing after hav-
ing trained the classifier for 1 epoch; they were 86.07 and 88.77 %, respectively.
This is in agreement with the common wisdom in machine learning that training
beyond a single pass through the data rarely improves testing accuracy (as it may
lead to overfitting). This is also the reason behind the success of light-touch meth-
ods, such as coordinate descent and stochastic gradient descent, in machine learning
applications.

9 Appendix 1: Notation glossary

See Table 8.

10 Appendix 2: More ESO theory

In this section we establish certain ESO results which do not play a key role in the main development of the paper, but which are nevertheless fundamental.

10.1 ESO for a convex combination of samplings

Let $\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_m$ be a collection of samplings and let $q \in \mathbf{R}^m$ be a probability vector. By $\sum_j q_j \hat{S}_j$ we denote the sampling $\hat{S}$ given by

$$\mathbf{P}\left(\hat{S} = S\right) = \sum_{j=1}^m q_j \mathbf{P}(\hat{S}_j = S). \tag{80}$$

This procedure allows us to build new samplings from existing ones. A natural interpretation of $\hat{S}$ is that it arises from a two stage process as follows. Generating a set via $\hat{S}$ is equivalent to first choosing j with probability q_j, and then generating a set via $\hat{S}_j$.

Lemma 19 *Let $\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_m$ be arbitrary samplings, $q \in \mathbf{R}^m$ a probability vector and $\kappa : 2^{[n]} \to \mathbf{R}$ any function mapping subsets of $[n]$ to reals. If we let $\hat{S} = \sum_j q_j \hat{S}_j$, then*

(i) $\mathbf{E}[\kappa(\hat{S})] = \sum_{j=1}^m q_j \mathbf{E}[\kappa(\hat{S}_j)]$,
(ii) $\mathbf{E}[|\hat{S}|] = \sum_{j=1}^m q_j \mathbf{E}[|\hat{S}_j|]$,
(iii) $\mathbf{P}(i \in \hat{S}) = \sum_{j=1}^m q_j \mathbf{P}(i \in \hat{S}_j)$, *for any* $i = 1, 2, \ldots, n$,
(iv) If $\hat{S}_1, \ldots, \hat{S}_m$ are uniform (resp. doubly uniform), so is $\hat{S}$.

Proof Statement (i) follows by writing $\mathbf{E}[\kappa(\hat{S})]$ as

$$\sum_S \mathbf{P}(\hat{S} = S)\kappa(S) \stackrel{(80)}{=} \sum_S \sum_{j=1}^m q_j \mathbf{P}(\hat{S}_j = S)\kappa(S) = \sum_{j=1}^m q_j \sum_S \mathbf{P}(\hat{S}_j = S)\kappa(S)$$

$$= \sum_{j=1}^m q_j \mathbf{E}[\kappa(\hat{S}_j)].$$

Statement (ii) follows from (i) by choosing $\kappa(S) = |S|$, and (iii) follows from (i) by choosing κ as follows: $\kappa(S) = 1$ if $i \in S$ and $\kappa(S) = 0$ otherwise. Finally, if the

Table 8 The main notation used in the paper

Optimization problem (Sect. 1)		
N	Dimension of the optimization variable	(1)
x, h	Vectors in $\mathbf{R}^N$	
f	Smooth convex function ($f : \mathbf{R}^N \to \mathbf{R}$)	(1)
Ω	Convex block separable function ($\Omega : \mathbf{R}^N \to \mathbf{R} \cup \{+\infty\}$)	(1)
F	$F = f + \Omega$ (loss / objective function)	(1)
ω	Degree of partial separability of f	(2,3)
Block structure (Sect. 2.1)		
n	Number of blocks	
$[n]$	$[n] = \{1, 2, \ldots, n\}$ (the set of blocks)	Sect. 2.1
N_i	Dimension of block i ($N_1 + \ldots + N_n = N$)	Sect. 2.1
U_i	An $N_i \times N$ column submatrix of the $N \times N$ identity matrix	Prop. 1
$x^{(i)}$	$x^{(i)} = U_i^T x, x \in \mathbf{R}^{N_i}$ (block i of vector x)	Prop. 1
$\nabla_i f(x)$	$\nabla_i f(x) = U_i^T \nabla f(x)$ (block gradient of f associated with block i)	(11)
L_i	Block Lipschitz constant of the gradient of f	(11)
L	$L = (L_1, \ldots, L_n)^T \in \mathbf{R}^n$ (vector of block Lipschitz constants)	
w	$w = (w_1, \ldots, w_n)^T \in \mathbf{R}^n$ (vector of positive weights)	
$\mathrm{Supp}(h)$	$\mathrm{Supp}(h) = \{i \in [n] : x^{(i)} \neq 0\}$ (set of nonzero blocks of x)	
B_i	An $N_i \times N_i$ positive definite matrix	
$\| \cdot \|_{(i)}$	$\|x^{(i)}\|_{(i)} = \langle B_i x^{(i)}, x^{(i)} \rangle^{1/2}$ (norm associated with block of i)	(10)
$\|x\|_w$	$\|x\|_w = (\sum_{i=1}^n w_i \|x^{(i)}\|_{(i)}^2)^{1/2}$ (weighted norm associated with x)	(13)
Ω_i	ith compoent of $\Omega = \Omega_1 + \ldots + \Omega_n$	
$\mu_\Omega(w)$	Strong convexity constant of Ω with respect to the norm $\| \cdot \|_w$	(14)
$\mu_f(w)$	Strong convexity constant of f with respect to the norm $\| \cdot \|_w$	(14)

Table 8 continued

Block samplings (Sect. 4)

S, J	Subsets of $\{1, 2, \ldots, n\}$	
$\hat{S}, S_k$	Block samplings (random subsets of $\{1, 2, \ldots, n\}$)	
$x_{[S]}$	Vector in $\mathbf{R}^N$ formed from x by zeroing out blocks $x^{(i)}$ for $i \notin S$	(7), (8)
τ	# of blocks updated in 1 iteration (when $\mathbf{P}(|\hat{S}| = \tau) = 1$)	
$\mathbf{E}[|\hat{S}|]$	Average # of blocks updated in 1 iteration (when $\mathbf{Var}[|\hat{S}|] > 0$)	
$p(S)$	$p(S) = \mathbf{P}(\hat{S} = S)$	(20)
p_i	$p_i := \mathbf{P}(i \in \hat{S})$	(21)
p	$p = (p_1, \ldots, p_n)^T \in \mathbf{R}^n$	(21)

Algorithm (Sect. 2.2)

β	Stepsize parameter depending on f and $\hat{S}$ (a central object in this paper)	
$H_{\beta,w}(x, h)$	$H_{\beta,w}(x, h) = f(x) + \langle \nabla f(x), h \rangle + \frac{\beta}{2}\|h\|_w^2 + \Omega(x + h)$	(18)
$h(x)$	$h(x) = \arg\min_{h \in \mathbf{R}^N} H_{\beta,w}(x, h)$	(17)
$h^{(i)}(x)$	$h^{(i)}(x) = (h(x))^{(i)} = \arg\min_{t \in \mathbf{R}^{N_i}} \langle \nabla_i f(x), t \rangle + \frac{\beta w_i}{2}\|t\|_{(i)}^2 + \Omega_i(x^{(i)} + t)$	(17)
x_{k+1}	$x_{k+1} = x_k + \sum_{i \in S_k} U_i h^{(i)}(x_k)$ (x_k is the kth iterate of PCDM)	

samplings $\hat{S}_j$ are uniform, from (33) we know that $\mathbf{P}(i \in \hat{S}_j) = \mathbf{E}[|\hat{S}_j|]/n$ for all i and j. Plugging this into identity (iii) shows that $\mathbf{P}(i \in \hat{S})$ is independent of i, which shows that $\hat{S}$ is uniform. Now assume that $\hat{S}_j$ are doubly uniform. Fixing arbitrary $\tau \in \{0\} \cup [n]$, for every $S \subset [n]$ such that $|S| = \tau$ we have

$$\mathbf{P}(\hat{S} = S) \overset{(80)}{=} \sum_{j=1}^{m} q_j \mathbf{P}(\hat{S}_j = S) = \sum_{j=1}^{m} q_j \frac{\mathbf{P}(|\hat{S}_j| = \tau)}{\binom{n}{\tau}}.$$

As the last expression depends on S via $|S|$ only, $\hat{S}$ is doubly uniform. □

Remark 1. If we fix $S \subset [n]$ and define $k(S') = 1$ if $S' = S$ and $k(S') = 0$ otherwise, then statement (i) of Lemma 19 reduces to (80).
2. All samplings arise as a combination of *elementary* samplings, i.e., samplings whose all weight is on one set only. Indeed, let $\hat{S}$ be an arbitrary sampling. For all subsets S_j of $[n]$ define $\hat{S}_j$ by $\mathbf{P}(\hat{S}_j = S_j) = 1$ and let $q_j = \mathbf{P}(\hat{S} = S_j)$. Then clearly, $\hat{S} = \sum_j q_j \hat{S}_j$.
3. All doubly uniform samplings arise as convex combinations of nice samplings.

Often it is easier to establish ESO for a simple class of samplings (e.g., nice samplings) and then use it to obtain an ESO for a more complicated class (e.g., doubly uniform samplings as they arise as convex combinations of nice samplings). The following result is helpful in this regard.

Theorem 20 (Convex Combination of Uniform Samplings) *Let $\hat{S}_1, \ldots, \hat{S}_m$ be uniform samplings satisfying $(f, \hat{S}_j) \sim ESO(\beta_j, w_j)$ and let $q \in \mathbf{R}^m$ be a probability vector. If $\sum_j q_j \hat{S}_j$ is not nil, then*

$$\left(f, \sum_{j=1}^{m} q_j \hat{S}_j \right) \sim ESO \left(\frac{1}{\sum_{j=1}^{m} q_j \mathbf{E}[|\hat{S}_j|]}, \sum_{j=1}^{m} q_j \mathbf{E}[|\hat{S}_j|] \beta_j w_j \right).$$

Proof First note that from part (iv) of Lemma 19 we know that $\hat{S} \overset{\text{def}}{=} \sum_j q_j \hat{S}_j$ is uniform and hence it makes sense to speak about ESO involving this sampling. Next, we can write

$$\mathbf{E}\left[f(x + h_{[\hat{S}]}) \right] = \sum_{S} \mathbf{P}(\hat{S} = S) f(x + h_{[S]}) \overset{(80)}{=} \sum_{S} \sum_{j} q_j \mathbf{P}(\hat{S}_j = S) f(x + h_{[S]})$$

$$= \sum_{j} q_j \sum_{S} \mathbf{P}(\hat{S}_j = S) f(x + h_{[S]}) = \sum_{j} q_j \mathbf{E}\left[f(x + h_{[\hat{S}_j]}) \right].$$

It now remains to use (43) and part (ii) of Lemma 19:

$$\sum_{j=1}^{m} q_j \mathbf{E}\left[f(x + h_{[\hat{S}_j]}) \right]$$

$$\overset{(43)}{\leq} \sum_{j=1}^{m} q_j \left(f(x) + \frac{\mathbf{E}[|\hat{S}_j|]}{n} \left(\langle \nabla f(x), h \rangle + \frac{\beta_j}{2} \|h\|_{w_j}^2 \right) \right)$$

$$= f(x) + \frac{\sum_j q_j \mathbf{E}[|\hat{S}_j|]}{n} \langle \nabla f(x), h \rangle$$

$$+ \frac{1}{2n} \sum_j q_j \mathbf{E}[|\hat{S}_j|]\beta_j \|h\|_{w_j}^2$$

$$\overset{(\text{Lemma}(19)\ (ii))}{=} f(x) + \frac{\mathbf{E}[|\hat{S}|]}{n} \left(\langle \nabla f(x), h \rangle + \frac{\sum_j q_j \mathbf{E}[|\hat{S}_j|]\beta_j \|h\|_{w_j}^2}{2 \sum_j q_j \mathbf{E}[|\hat{S}_j|]} \right)$$

$$= f(x) + \frac{\mathbf{E}[|\hat{S}|]}{n} \left(\langle \nabla f(x), h \rangle + \frac{1}{2 \sum_j q_j \mathbf{E}[|\hat{S}_j|]} \|h\|_w^2 \right),$$

where $w = \sum_j q_j \mathbf{E}[|\hat{S}_j|]\beta_j w_j$. In the third step we have also used the fact that $\mathbf{E}[|\hat{S}|] > 0$ which follows from the assumption that $\hat{S}$ is not nil. $\square$

10.2 ESO for a conic combination of functions

We now establish an ESO for a conic combination of functions each of which is already equipped with an ESO. It offers a complementary result to Theorem 20.

Theorem 21 (Conic Combination of Functions) *If $(f_j, \hat{S}) \sim ESO(\beta_j, w_j)$ for $j = 1, \ldots, m$, then for any $c_1, \ldots, c_m \geq 0$ we have*

$$\left(\sum_{j=1}^{m} c_j f_j, \hat{S} \right) \sim ESO \left(1, \sum_{j=1}^{m} c_j \beta_j w_j \right).$$

Proof Letting $f = \sum_j c_j f_j$, we get

$$\mathbf{E}\left[\sum_j c_j f_j \left(x + h_{[\hat{S}]} \right) \right] = \sum_j c_j \mathbf{E}\left[f_j \left(x + h_{[\hat{S}]} \right) \right]$$

$$\leq \sum_j c_j \left(f_j(x) + \frac{\mathbf{E}[|\hat{S}|]}{n} \left(\langle \nabla f_j(x), h \rangle + \frac{\beta_j}{2} \|h\|_{w_j}^2 \right) \right)$$

$$= \sum_j c_j f_j(x) + \frac{\mathbf{E}[|\hat{S}|]}{n} \left(\sum_j c_j \langle \nabla f_j(x), h \rangle \right.$$

$$\left. + \sum_j \frac{c_j \beta_j}{2} \|h\|_{w_j}^2 \right)$$

$$= f(x) + \frac{\mathbf{E}[|\hat{S}|]}{n} \left(\langle \nabla f(x), h \rangle + \frac{1}{2} \|h\|_{\sum_j c_j \beta_j w_j}^2 \right).   \square$$

References

1. Bradley, J.K., Kyrola, A., Bickson, D., Guestrin, C.: Parallel coordinate descent for L1-regularized loss minimization. In ICML (2011)
2. Dhillon, I., Ravikumar, P., Tewari, A.: Nearest neighbor based greedy coordinate descent. NIPS **24**, 2160–2168 (2011)
3. Fercoq, O., Richtárik, P.: Smooth minimization of nonsmooth functions with parallel coordinate descent methods. arXiv:1309.5885 (2013)
4. Leventhal, D., Lewis, A.S.: Randomized methods for linear constraints: convergence rates and conditioning. Math. Oper. Res. **35**(3), 641–654 (2010)
5. Li, Y., Osher, S.: Coordinate descent optimization for l_1 minimization with application to compressed sensing; a greedy algorithm. Inverse Probl. Imaging **3**, 487–503 (2009)
6. Necoara, Ion, Clipici, Dragos: Efficient parallel coordinate descent algorithm for convex optimization problems with separable constraints: application to distributed mpc. J. Process Control **23**(3), 243–253 (2013)
7. Necoara, I., Nesterov, Y., Glineur, F.: Efficiency of randomized coordinate descent methods on optimization problems with linearly coupled constraints. Technical report, Politehnica University of Bucharest (2012)
8. Necoara, I., Patrascu, A.: A random coordinate descent algorithm for optimization problems with composite objective function and linear coupled constraints. Comput. Optim. Appl. **57**(2), 307–337 (2014)
9. Nesterov, Yurii: Introductory Lectures on Convex Optimization: A Basic Course (Applied Optimization). Kluwer, Dordrecht (2004)
10. Nesterov, Y.: Efficiency of coordinate descent methods on huge-scale optimization problems. SIAM J. Optim. **22**(2), 341–362 (2012)
11. Nesterov, Yurii: Gradient methods for minimizing composite objective function. Math. Program. Ser. B **140**(1), 125–161 (2013)
12. Nesterov, Y.: Subgradient methods for huge-scale optimization problems. Math. Program. **146**(1–2), 275–297 (2014)
13. Niu, F., Recht, B., Ré, C., Wright, S.: Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In NIPS (2011)
14. Peng, Z., Yan, M., Yin, W.: Parallel and distributed sparse optimization. In Signals, Systems and Computers, 2013 Asilomar Conference on IEEE, pp. 659–646 (2013)
15. Richtárik, P., Takáč, M.: Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. Math Program **144**(1–2), 1–38 (2014)
16. Richtárik, P, Takáč, M: Efficient serial and parallel coordinate descent method for huge-scale truss topology design. In Klatte, D., Lüthi, H-J., Schmedders, K. (eds.) Operations Research Proceedings, pp. 27–32. Springer, Berlin (2012)
17. Richtárik, P., Takáč, M.: Distributed coordinate descent method for learning with big data. arXiv:1310.2059 (2013)
18. Richtárik, P., Takáč, M.: On optimal probabilities in stochastic coordinate descent methods. arXiv:1310.3438 (2013)
19. Richtárik, P., Takáč, M.: Efficiency of randomized coordinate descent methods on minimization problems with a composite objective function. In 4th Workshop on Signal Processing with Adaptive Sparse Structured Representations (June 2011)
20. Ruszczynski, A.: On convergence of an augmented Lagrangian decomposition method for sparse convex optimization. Math. Oper. Res. **20**(3), 634–656 (1995)
21. Saha, A., Tewari, A.: On the nonasymptotic convergence of cyclic coordinate descent methods. SIAM J. Optim. **23**(1), 576–601 (2013)
22. Scherrer, C., Tewari, A., Halappanavar, M., Haglin, D.J.: Feature clustering for accelerating parallel coordinate descent. In NIPS, pp. 28–36 (2012)
23. Shalev-Shwartz, S., Tewari, A.: Stochastic methods for ℓ_1-regularized loss minimization. J. Mach. Learn. Res. **12**, 1865–1892 (2011)
24. Strohmer, T., Vershynin, R.: A randomized Kaczmarz algorithm with exponential convergence. J. Fourier Anal. Appl. **15**, 262–278 (2009)
25. Takáč, M., Bijral, A., Richtárik, P., Srebro, N.: Mini-batch primal and dual methods for SVMs. In ICML (2013)

26. Tappenden, R., Richtárik, P., Gondzio, J.: Inexact coordinate descent: complexity and preconditioning. arXiv:1304.5530, (April 2013)
27. Jinchao, X.: Iterative methods by space decomposition and subspace correction. SIAM Rev. **34**(4), 581–613 (1992)
28. Yu, H.F., Hsieh, C. J., Si, S., Dhillon, I.: Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In IEEE 12th International Conference on Data Mining, pp. 765–774 (2012)
29. Zargham, M., Ribeiro, A., Ozdaglar, A., Jadbabaie, A.: Accelerated dual descent for network optimization. In American Control Conference (ACC), 2011, pp. 2663–2668. IEEE (2011)