Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function

Peter Richtárik · **Martin Takáč**

Abstract In this paper we develop a randomized block-coordinate descent method for minimizing the sum of a smooth and a simple nonsmooth block-separable convex function and prove that it obtains an ε-accurate solution with probability at least $1 - \rho$ in at most $O((n/\varepsilon)\log(1/\rho))$ iterations, where n is the number of blocks. This extends recent results of Nesterov [Efficiency of coordinate descent methods on huge-scale optimization problems, SIAM J Optimization 22(2), pp. 341–362, 2012], which cover the smooth case, to composite minimization, while at the same time improving the complexity by the factor of 4 and removing ε from the logarithmic term. More importantly, in contrast with the aforementioned work in which the author achieves the results by applying the method to a regularized version of the objective function with an unknown scaling factor, we show that this is not necessary, thus achieving first true iteration complexity bounds. For strongly convex functions the method converges linearly. In the smooth case we also allow for arbitrary probability vectors and non-Euclidean norms. Finally, we demonstrate numerically that the algorithm is able to solve huge-scale ℓ_1-regularized least squares with a billion variables.

Peter Richtárik
School of Mathematics, University of Edinburgh, EH9 3JZ, United Kingdom
E-mail: peter.richtarik@ed.ac.uk

Martin Takáč
School of Mathematics, University of Edinburgh, EH9 3JZ, United Kingdom
E-mail: takac.mt@gmail.com

1 Introduction

The goal of this paper, in the broadest sense, is to develop efficient methods for solving structured convex optimization problems with some or all of these (not necessarily distinct) properties:

1. **Size of Data.** The size of the problem, measured as the dimension of the variable of interest, is so large that the computation of a single function value or gradient is prohibitive. There are several situations in which this is the case, let us mention two of them.
   - **Memory.** If the dimension of the space of variables is larger than the available memory, the task of forming a gradient or even of evaluating the function value may be impossible to execute and hence the usual gradient methods will not work.
   - **Patience.** Even if the memory does not preclude the possibility of taking a gradient step, for large enough problems this step will take considerable time and, in some applications such as image processing, users might prefer to see/have some intermediary results before a single iteration is over.
2. **Nature of Data.** The nature and structure of data describing the problem may be an obstacle in using current methods for various reasons, including the following.
   - **Completeness.** If the data describing the problem is not immediately available in its entirety, but instead arrives incomplete in pieces and blocks over time, with each block "corresponding to" one variable, it may not be realistic (for various reasons such as "memory" and "patience" described above) to wait for the entire data set to arrive before the optimization process is started.
   - **Source.** If the data is distributed on a network not all nodes of which are equally responsive or functioning, it may be necessary to work with whatever data is available at a given time.

It appears that a very reasonable approach to solving *some* problems characterized above is to use *(block) coordinate descent methods* (CD). In the remainder of this section we mix arguments in support of this claim with a brief review of the relevant literature and an outline of our contributions.

1.1 Block Coordinate Descent Methods

The basic algorithmic strategy of CD methods is known in the literature under various names such as alternating minimization, coordinate relaxation, linear and non-linear Gauss-Seidel methods, subspace correction and domain decomposition. As working with all the variables of an optimization problem at each iteration may be inconvenient, difficult or impossible for any or all of the reasons mentioned above, the variables are partitioned into manageable blocks, with each iteration focused on updating a single block only, the remaining blocks being fixed. Both for their conceptual and algorithmic simplicity, CD methods were among the first optimization approaches proposed and studied in the literature (see [1] and the references therein; for a survey of block CD methods in semidefinite programming we refer the reader to [24]). While they seem to have never belonged to the mainstream focus of the optimization community, a renewed interest in CD methods was sparked recently by their successful application in several areas—training support vector machines in machine learning [5], [3], [18], [28], [29], optimization [9], [23], [21], [31], [17], [13], [25], compressed

sensing [8], regression [27], protein loop closure [2] and truss topology design [16]—partly due to a change in the *size* and *nature of data* described above.

Order of coordinates. Efficiency of a CD method will necessarily depend on the balance between time spent on choosing the block to be updated in the current iteration and the quality of this choice in terms of function value decrease. One extreme possibility is a *greedy* strategy in which the block with the largest descent or guaranteed descent is chosen. In our setup such a strategy is prohibitive as i) it would require all data to be available and ii) the work involved would be excessive due to the size of the problem. Even if one is able to compute all partial derivatives, it seems better to then take a full gradient step instead of a coordinate one, and avoid throwing almost all of the computed information away. On the other end of the spectrum are two very cheap strategies for choosing the incumbent coordinate: *cyclic* and *random*. Surprisingly, it appears that complexity analysis of a cyclic CD method in satisfying generality has not yet been done. The only attempt known to us is the work of Saha and Tewari [17]; the authors consider the case of minimizing a smooth convex function and proceed by establishing a sequence of comparison theorems between the iterates of their method and the iterates of a simple gradient method. Their result requires an isotonicity assumption[1]. Note that a cyclic strategy assumes that the data describing the next block is available when needed which may not always be realistic. The situation with a random strategy seems better; here are some of the reasons:

 (i) Recent efforts suggest that complexity results are perhaps more readily obtained for randomized methods and that randomization can actually improve the convergence rate [19], [6], [18].
 (ii) Choosing all blocks with equal probabilities should, intuitively, lead to similar results as is the case with a cyclic strategy. In fact, a randomized strategy is able to avoid worst-case order of coordinates, and hence might be preferable.
(iii) Randomized choice seems more suitable in cases when not all data is available at all times.
(iv) One may study the possibility of choosing blocks with different probabilities (we do this in Section 4). The goal of such a strategy may be either to improve the speed of the method (in Section 6.1 we introduce a speedup heuristic based on adaptively changing the probabilities), or a more realistic modeling of the availability frequencies of the data defining each block.

Step size. Once a coordinate (or a block of coordinates) is chosen to be updated in the current iteration, partial derivative can be used to drive the steplength in the same way as it is done in the usual gradient methods. As it is sometimes the case that the computation of a partial derivative is *much cheaper and less memory demanding* than the computation of the entire gradient, CD methods seem to be promising candidates for problems described above. It is important that line search, if any is implemented, is very efficient. The entire data set is either huge or not available and hence it is not reasonable to use function values at any point in the algorithm, including the line search. Instead, cheap partial derivative and other information derived from the problem structure should be used to drive such a method.

---

[1] A function $F : \mathbb{R}^N \to \mathbb{R}$ is *isotone* if $x \geq y$ implies $F(x) \geq F(y)$.

1.2 Problem Description and Our Contribution

In this paper we study the *iteration complexity* of simple randomized block coordinate descent methods applied to the problem of minimizing a *composite objective function*, i.e., a function formed as the sum of a smooth convex and a simple nonsmooth convex term:

$$\min_{x \in \mathbb{R}^N} F(x) \stackrel{\text{def}}{=} f(x) + \Psi(x). \tag{1}$$

We assume that this problem has a minimum ($F^* > -\infty$), f has (block) coordinate Lipschitz gradient, and Ψ is a (block) separable proper closed convex extended real valued function (block separability will be defined precisely in Section 2). Possible choices of Ψ include:

(i) $\Psi \equiv 0$. This covers the case of *smooth minimization* and was considered in [13].
(ii) Ψ is the indicator function of a block-separable convex set (a box), i.e.,

$$\Psi(x) = I_{S_1 \times \cdots \times S_n}(x) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } x^{(i)} \in S_i \quad \forall i, \\ +\infty & \text{otherwise,} \end{cases}$$

where $x^{(i)}$ is block i of $x \in \mathbb{R}^N$ (to be defined precisely in Section 2) and $S_1, \ldots, S_n$ are closed convex sets. This choice of Ψ models problems with *smooth objective and convex constraints on blocks of variables*. Indeed, (1) takes on the form

$$\min \ f(x) \quad \text{subject to} \quad x^{(i)} \in S_i, \quad i = 1, \ldots, n.$$

Iteration complexity results in this case were given in [13].
(iii) $\Psi(x) \equiv \lambda \|x\|_1$ for $\lambda > 0$. In this case we decompose $\mathbb{R}^N$ into N blocks, each corresponding to one coordinate of x. Increasing λ encourages the solution of (1) to be sparser [26]. Applications abound in, for instance, machine learning [3], statistics [20] and signal processing [8]. The first iteration complexity results for the case with a single block were given in [12].
(iv) There are many more choices such as the elastic net [32], group lasso [30], [10], [14] and sparse group lasso [4]. One may combine indicator functions with other block separable functions such as $\Psi(x) = \lambda \|x\|_1 + I_{S_1 \times \cdots \times S_n}(x)$, $S_i = [l_i, u_i]$, where the sets introduce lower and upper bounds on the coordinates of x.

Iteration complexity results. Strohmer and Vershynin [19] have recently proposed a randomized Karczmarz method for solving overdetermined consistent systems of linear equations and proved that the method enjoys global linear convergence whose rate can be expressed in terms of the condition number of the underlying matrix. The authors claim that for certain problems their approach can be more efficient than the conjugate gradient method. Motivated by these results, Leventhal and Lewis [6] studied the problem of solving a system of linear equations and inequalities and in the process gave iteration complexity bounds for a randomized CD method applied to the problem of minimizing a convex quadratic function. In their method the probability of choice of each coordinate is proportional to the corresponding diagonal element of the underlying positive semidefinite matrix defining the objective function. These diagonal elements can be interpreted as Lipschitz constants of the derivative of a restriction of the quadratic objective onto one-dimensional lines parallel to the coordinate axes. In the general (as opposed to quadratic) case considered in this paper (1), these Lipschitz constants will play an important role as well. Lin et al. [3] derived iteration complexity results for several smooth objective functions appearing in machine learning.

Shalev-Schwarz and Tewari [18] proposed a randomized coordinate descent method with uniform probabilities for minimizing ℓ_1-regularized smooth convex problems. They first transform the problem into a box constrained smooth problem by doubling the dimension and then apply a coordinate gradient descent method in which each coordinate is chosen with equal probability. Nesterov [13] has recently analyzed randomized coordinate descent methods in the smooth unconstrained and box-constrained setting, in effect extending and improving upon some of the results in [6], [3], [18] in several ways.

While the *asymptotic convergence rates* of some variants of CD methods are well understood [9], [23], [21], [31], *iteration complexity* results are very rare. To the best of our knowledge, randomized CD algorithms for minimizing a composite function have been proposed and analyzed (in the iteration complexity sense) in a few special cases only: a) the unconstrained convex quadratic case [6], b) the smooth unconstrained ($\Psi \equiv 0$) and the smooth block-constrained case (Ψ is the indicator function of a direct sum of boxes) [13] and c) the ℓ_1-regularized case [18]. As the approach in [18] is to rewrite the problem into a smooth box-constrained format first, the results of [13] can be viewed as a (major) generalization and improvement of those in [18] (the results were obtained independently).

Our contribution. In this paper we further improve upon and extend and simplify the iteration complexity results of Nesterov [13], treating the problem of minimizing the sum of a smooth convex and a simple nonsmooth convex block separable function (1). We focus exclusively on simple (as opposed to accelerated) methods. The reason for this is that the per-iteration work of the accelerated algorithm in [13] on huge scale instances of problems with *sparse* data (such as the Google problem where sparsity corresponds to each website linking only to a few other websites or the sparse problems we consider in Section 6) is excessive. In fact, even the author does not recommend using the accelerated method for solving such problems; the simple methods seem to be more efficient.

Each algorithm of this paper is supported by a high probability iteration complexity result. That is, for any given *confidence level* $0 < \rho < 1$ and *error tolerance* $\varepsilon > 0$, we give an explicit expression for the number of iterations k which guarantee that the method produces a random iterate x_k for which

$$\mathbf{P}(F(x_k) - F^* \leq \varepsilon) \geq 1 - \rho.$$

Table 1 summarizes the main complexity results of this paper. Algorithm 2—Uniform (block) Coordinate Descent for Composite functions (UCDC)—is a method where at each iteration the block of coordinates to be updated (out of a total of $n \leq N$ blocks) is chosen uniformly at random. Algorithm 3—Randomized (block) Coordinate Descent for Smooth functions (RCDS)—is a method where at each iteration block $i \in \{1, \ldots, n\}$ is chosen with probability $p_i > 0$. Both of these methods are special cases of the generic Algorithm 1 (Section 2); Randomized (block) Coordinate Descent for Composite functions (RCDC).

The symbols $P, L, \mathcal{R}_W^2(x_0)$ and $\mu_\phi(W)$ appearing in Table 1 will be defined precisely in Section 2. For now it suffices to say that L is a diagonal matrix encoding the (block) coordinate Lipschitz constants of the gradient of f, P is a diagonal matrix encoding the probabilities $\{p_i\}$, $\mathcal{R}_W^2(x_0)$ is a measure of the squared distance of the initial iterate x_0 from the set of minimizers of problem (1) in a norm defined by a diagonal matrix W and $\mu_\phi(W)$ is the strong convexity parameter of function ϕ with respect to that norm.

Let us now briefly outline the main similarities and differences between our results and those in [13]. A more detailed and expanded discussion can be found in Section 5.

Algorithm	Objective	Complexity
Algorithm 2 (UCDC) (Theorem 5)	convex composite	$\frac{2n\max\{\mathscr{R}_L^2(x_0),F(x_0)-F^*\}}{\varepsilon}(1+\log\frac{1}{\rho})$ $\frac{2n\mathscr{R}_L^2(x_0)}{\varepsilon}\log\left(\frac{F(x_0)-F^*}{\varepsilon\rho}\right)$
Algorithm 2 (UCDC) (Theorem 8)	strongly convex composite	$n\frac{1+\mu_\Psi(L)}{\mu_f(L)+\mu_\Psi(L)}\log\left(\frac{F(x_0)-F^*}{\varepsilon\rho}\right)$
Algorithm 3 (RCDS) (Theorem 12)	convex smooth	$\frac{2\mathscr{R}_{LP^{-1}}^2(x_0)}{\varepsilon}(1+\log\frac{1}{\rho})-2$
Algorithm 3 (RCDS) (Theorem 13)	strongly convex smooth	$\frac{1}{\mu_f(LP^{-1})}\log\left(\frac{f(x_0)-f^*}{\varepsilon\rho}\right)$

Table 1 Summary of complexity results obtained in this paper.

1. **Composite setting.** We consider the composite setting[2] (1), whereas [13] covers the unconstrained and constrained smooth setting only.
2. **No need for regularization.** Nesterov's high probability results in the case of minimizing a function which is not strongly convex are based on regularizing the objective to make it strongly convex and then running the method on the regularized function. The regularizing term depends on the distance of the initial iterate to the optimal point, and hence is *unknown*, which means that the analysis in [13] does not lead to true iteration complexity results. Our contribution here is that we show that no regularization is needed by doing a more detailed analysis using a thresholding argument (Theorem 1).
3. **Better complexity.** Our complexity results are better by the constant factor of 4. Also, we have removed ε from the logarithmic term.
4. **General probabilities.** Nesterov considers probabilities p_i proportional to L_i^α, where $\alpha \geq 0$ is a parameter. High probability results are proved in [13] for $\alpha \in \{0,1\}$ only. Our results in the smooth case hold for an arbitrary probability vector p.
5. **General norms.** Nesterov's expectation results (Theorems 1 and 2) are proved for general norms. However, his high probability results are proved for Euclidean norms only. In our approach all results in the smooth case hold for general norms.
6. **Simplification.** Our analysis is shorter.

In the numerical experiments section we focus on sparse regression. For these problems we introduce a powerful *speedup heuristic* based on adaptively changing the probability vector throughout the iterations.

1.3 Contents

This paper is organized as follows. We start in Section 2 by defining basic notation, describing the block structure of the problem, stating assumptions and describing the generic randomized block-coordinate descent algorithm (RCDC). In Section 3 we study the performance of a uniform variant (UCDC) of RCDC as applied to a composite objective function

---

[2] Note that in [12] Nesterov considered the composite setting and developed standard and accelerated gradient methods with iteration complexity guarantees for minimizing composite objective functions. These can be viewed as block coordinate descent methods with a *single* block.

and in Section 4 we analyze a smooth variant (RCDS) of RCDC; that is, we study the performance of RCDC on a smooth objective function. In Section 5 we compare known complexity results for CD methods with the ones established in this paper. Finally, in Section 6 we demonstrate the efficiency of the method on ℓ_1-regularized least squares and linear support vector machine problems.

2 Preliminaries

In Section 2.1 we describe the setting, basic assumptions and notation, Section 2.2 describes the algorithm and in Section 2.3 we present the key technical tool of our complexity analysis.

2.1 Assumptions and Notation

Block structure. We model the block structure of the problem by decomposing the space $\mathbb{R}^N$ into n subspaces as follows. Let $U \in \mathbb{R}^{N \times N}$ be a column permutation 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 $\sum_i N_i = N$. Clearly, any vector $x \in \mathbb{R}^N$ can be written uniquely as $x = \sum_i U_i x^{(i)}$, where $x^{(i)} = U_i^T x \in \mathbb{R}_i \equiv \mathbb{R}^{N_i}$. Also note that

$$U_i^T U_j = \begin{cases} N_i \times N_i & \text{identity matrix,} & \text{if } i = j, \\ N_i \times N_j & \text{zero matrix,} & \text{otherwise.} \end{cases} \tag{2}$$

For simplicity we will write $x = (x^{(1)}, \ldots, x^{(n)})^T$. We equip $\mathbb{R}_i$ with a pair of conjugate Euclidean norms:

$$\|t\|_{(i)} = \langle B_i t, t \rangle^{1/2}, \qquad \|t\|_{(i)}^* = \langle B_i^{-1} t, t \rangle^{1/2}, \qquad t \in \mathbb{R}_i, \tag{3}$$

where $B_i \in \mathbb{R}^{N_i \times N_i}$ is a positive definite matrix and $\langle \cdot, \cdot \rangle$ is the standard Euclidean inner product.

Example: Let $n = N$, $N_i = 1$ for all i and $U = [e_1, e_2, \ldots, e_n]$ be the $n \times n$ identity matrix. Then $U_i = e_i$ is the i-th unit vector and $x^{(i)} = e_i^T x \in \mathbb{R}_i = \mathbb{R}$ is the i-th coordinate of x. Also, $x = \sum_i e_i x^{(i)}$. If we let $B_i = 1$ for all i, then $\|t\|_{(i)} = \|t\|_{(i)}^* = |t|$ for all $t \in \mathbb{R}$.

Smoothness of f. We assume throughout the paper that the gradient of f is block coordinate-wise Lipschitz, uniformly in x, with positive constants $L_1, \ldots, L_n$, i.e., that for all $x \in \mathbb{R}^N$, $i = 1, 2, \ldots, n$ and $t \in \mathbb{R}_i$ we have

$$\|\nabla_i f(x + U_i t) - \nabla_i f(x)\|_{(i)}^* \leq L_i \|t\|_{(i)}, \tag{4}$$

where

$$\nabla_i f(x) \stackrel{\text{def}}{=} (\nabla f(x))^{(i)} = U_i^T \nabla f(x) \in \mathbb{R}_i. \tag{5}$$

An important consequence of (4) is the following standard inequality [11]:

$$f(x + U_i t) \leq f(x) + \langle \nabla_i f(x), t \rangle + \frac{L_i}{2} \|t\|_{(i)}^2. \tag{6}$$

Separability of Ψ. We assume that $\Psi : \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$ is block separable, i.e., that it can be decomposed as follows:

$$\Psi(x) = \sum_{i=1}^n \Psi_i(x^{(i)}), \tag{7}$$

where the functions $\Psi_i : \mathbb{R}_i \to \mathbb{R} \cup \{+\infty\}$ are convex and closed.

Global structure. For fixed positive scalars $w_1, \ldots, w_n$ let $W = \mathrm{Diag}(w_1, \ldots, w_n)$ and define a pair of conjugate norms in $\mathbb{R}^N$ by

$$\|x\|_W \stackrel{\mathrm{def}}{=} \left[\sum_{i=1}^n w_i \|x^{(i)}\|_{(i)}^2 \right]^{1/2}, \qquad \|y\|_W^* \stackrel{\mathrm{def}}{=} \max_{\|x\|_W \leq 1} \langle y, x \rangle = \left[\sum_{i=1}^n w_i^{-1} (\|y^{(i)}\|_{(i)}^*)^2 \right]^{1/2}. \qquad (8)$$

We write $\mathrm{tr}(W) = \sum_i w_i$. In the the subsequent analysis we will use $W = L$ (Section 3) and $W = LP^{-1}$ (Section 4), where $L = \mathrm{Diag}(L_1, \ldots, L_n)$ and $P = \mathrm{Diag}(p_1, \ldots, p_n)$.

Level set radius. The set of optimal solutions of (1) is denoted by X^* and x^* is any element of that set. Define

$$\mathscr{R}_W(x) \stackrel{\mathrm{def}}{=} \max_y \max_{x^* \in X^*} \{ \|y - x^*\|_W \ : \ F(y) \leq F(x) \},$$

which is a measure of the size of the level set of F given by x. In some of the results in this paper we will need to assume that $\mathscr{R}_W(x_0)$ is finite for the initial iterate x_0 and $W = L$ or $W = LP^{-1}$.

Strong convexity of F. In some of our results we assume, and we always explicitly mention this if we do, that F is strongly convex with respect to the norm $\|\cdot\|_W$ for some W (we use $W = L$ in the nonsmooth case and $W = LP^{-1}$ in the smooth case), with (strong) convexity parameter $\mu_F(W) > 0$. A function $\phi : \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$ is strongly convex with respect to the norm $\|\cdot\|_W$ with convexity parameter $\mu_\phi(W) \geq 0$ if for all $x, y \in \mathrm{dom}\, \phi$,

$$\phi(y) \geq \phi(x) + \langle \phi'(x), y - x \rangle + \tfrac{\mu_\phi(W)}{2} \|y - x\|_W^2, \qquad (9)$$

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 Ψ or both; we will write $\mu_f(W)$ (resp. $\mu_\Psi(W)$) for the (strong) convexity parameter of f (resp. Ψ). It follows from (9) that

$$\mu_F(W) \geq \mu_f(W) + \mu_\Psi(W). \qquad (10)$$

The following characterization of strong convexity will also be useful. For all $x, y \in \mathrm{dom}\, \phi$ and $\alpha \in [0, 1]$,

$$\phi(\alpha x + (1 - \alpha)y) \leq \alpha \phi(x) + (1 - \alpha)\phi(y) - \tfrac{\mu_\phi(W)\alpha(1-\alpha)}{2} \|x - y\|_W^2. \qquad (11)$$

From the first order optimality conditions for (1) we obtain $\langle F'(x^*), x - x^* \rangle \geq 0$ for all $x \in \mathrm{dom}\, F$ which, combined with (9) used with $y = x$ and $x = x^*$, yields

$$F(x) - F^* \geq \tfrac{\mu_F(W)}{2} \|x - x^*\|_W^2, \quad x \in \mathrm{dom}\, F. \qquad (12)$$

Also, it can be shown using (6) and (9) that $\mu_f(L) \leq 1$.

Norm scaling. Note that since

$$\mu_\phi(tW) = \tfrac{1}{t} \mu_\phi(W), \qquad t > 0, \qquad (13)$$

the size of the (strong) convexity parameter depends inversely on the size of W. Hence, if we want to compare convexity parameters for different choices of W, we need to normalize W first. A natural way of normalizing W is to require $\mathrm{tr}(W) = \mathrm{tr}(I) = n$. If we now define

$$\widetilde{W} \stackrel{\mathrm{def}}{=} \tfrac{n}{\mathrm{tr}(W)} W, \qquad (14)$$

we have $\mathrm{tr}(\widetilde{W}) = n$ and

$$\mu_\phi(\widetilde{W}) \stackrel{(13),(14)}{=} \tfrac{\mathrm{tr}(W)}{n} \mu_\phi(W). \qquad (15)$$

2.2 The Algorithm

Notice that an upper bound on $F(x+U_i t)$, viewed as a function of $t \in \mathbb{R}_i$, is readily available:

$$F(x+U_i t) \overset{(1)}{=} f(x+U_i t) + \Psi(x+U_i t) \overset{(6)}{\leq} f(x) + V_i(x,t) + C_i(x), \tag{16}$$

where

$$V_i(x,t) \overset{\text{def}}{=} \langle \nabla_i f(x), t \rangle + \tfrac{L_i}{2} \|t\|_{(i)}^2 + \Psi_i(x^{(i)} + t) \tag{17}$$

and

$$C_i(x) \overset{\text{def}}{=} \sum_{j \neq i} \Psi_j(x^{(j)}). \tag{18}$$

We are now ready to describe the generic randomized (block) coordinate descent method for solving (1). Given iterate x_k, Algorithm 1 picks block $i_k = i \in \{1, 2, \ldots, n\}$ with probability $p_i > 0$ and then updates the i-th block of x_k so as to minimize (exactly) in t the upper bound (16) on $F(x_k + U_i t)$. Note that in certain cases it is possible to minimize $F(x_k + U_i t)$ directly; perhaps in a closed form. This is the case, for example, when f is a convex quadratic.

Algorithm 1 RCDC(p, x_0) (**R**andomized **C**oordinate **D**escent for **C**omposite Functions)

for $k = 0, 1, 2, \ldots$ **do**
    Choose $i_k = i \in \{1, 2, \ldots, n\}$ with probability p_i
    $T^{(i)}(x_k) \overset{\text{def}}{=} \arg\min\{V_i(x_k, t) : t \in \mathbb{R}_i\}$
    $x_{k+1} = x_k + U_i T^{(i)}(x_k)$
end for

The iterates $\{x_k\}$ are random vectors and the values $\{F(x_k)\}$ are random variables. Clearly, x_{k+1} depends on x_k only. As our analysis will be based on the (expected) per-iteration decrease of the objective function, our results hold if we replace $V_i(x_k, t)$ by $F(x_k + U_i t)$ in Algorithm 1.

2.3 Key Technical Tool

Here we present the main technical tool which is used at the end of of our iteration complexity proofs.

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

(i) $\mathbf{E}[\xi_{k+1} \mid x_k] \leq \xi_k - \frac{\xi_k^2}{c_1}$*, for all k, where $c_1 > 0$ 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 $\xi_k \geq \varepsilon$, where $c_2 > 1$ is a constant.*

If property (i) holds and we choose $\varepsilon < c_1$ and

$$K \geq \tfrac{c_1}{\varepsilon}(1 + \log \tfrac{1}{\rho}) + 2 - \tfrac{c_1}{\xi_0}, \tag{19}$$

or if property (ii) holds, and we choose

$$K \geq c_2 \log \frac{\xi_0}{\varepsilon\rho}, \tag{20}$$

then

$$\mathbf{P}(\xi_K \leq \varepsilon) \geq 1 - \rho. \tag{21}$$

Proof. First, notice that the sequence $\{\xi_k^\varepsilon\}_{k \geq 0}$ defined by

$$\xi_k^\varepsilon = \begin{cases} \xi_k & \text{if } \xi_k \geq \varepsilon, \\ 0 & \text{otherwise}, \end{cases} \tag{22}$$

satisfies $\xi_k^\varepsilon > \varepsilon \Leftrightarrow \xi_k > \varepsilon$. Therefore, by Markov inequality, $\mathbf{P}(\xi_k > \varepsilon) = \mathbf{P}(\xi_k^\varepsilon > \varepsilon) \leq \frac{\mathbf{E}[\xi_k^\varepsilon]}{\varepsilon}$, and hence it suffices to show that

$$\theta_K \leq \varepsilon\rho, \tag{23}$$

where $\theta_k \stackrel{\text{def}}{=} \mathbf{E}[\xi_k^\varepsilon]$. Assume now that property (i) holds. We first claim that then

$$\mathbf{E}[\xi_{k+1}^\varepsilon \mid x_k] \leq \xi_k^\varepsilon - \frac{(\xi_k^\varepsilon)^2}{c_1}, \qquad \mathbf{E}[\xi_{k+1}^\varepsilon \mid x_k] \leq (1 - \frac{\varepsilon}{c_1})\xi_k^\varepsilon, \qquad k \geq 0. \tag{24}$$

Consider two cases. Assuming that $\xi_k \geq \varepsilon$, from (22) we see that $\xi_k^\varepsilon = \xi_k$. This, combined with the simple fact that $\xi_{k+1}^\varepsilon \leq \xi_{k+1}$ and property (i), gives

$$\mathbf{E}[\xi_{k+1}^\varepsilon \mid x_k] \leq \mathbf{E}[\xi_{k+1} \mid x_k] \leq \xi_k - \frac{\xi_k^2}{c_1} = \xi_k^\varepsilon - \frac{(\xi_k^\varepsilon)^2}{c_1}.$$

Assuming that $\xi_k < \varepsilon$, we get $\xi_k^\varepsilon = 0$ and, from monotonicity assumption, $\xi_{k+1} \leq \xi_k < \varepsilon$. Hence, $\xi_{k+1}^\varepsilon = 0$. Putting these together, we get $\mathbf{E}[\xi_{k+1}^\varepsilon \mid x_k] = 0 = \xi_k^\varepsilon - (\xi_k^\varepsilon)^2/c_1$, which establishes the first inequality in (24). The second inequality in (24) follows from the first by again analyzing the two cases: $\xi_k \geq \varepsilon$ and $\xi_k < \varepsilon$. Now, by taking expectations in (24) (and using convexity of $t \mapsto t^2$ in the first case) we obtain, respectively,

$$\theta_{k+1} \leq \theta_k - \frac{\theta_k^2}{c_1}, \qquad k \geq 0, \tag{25}$$

$$\theta_{k+1} \leq (1 - \frac{\varepsilon}{c_1})\theta_k, \qquad k \geq 0. \tag{26}$$

Notice that (25) is better than (26) precisely when $\theta_k > \varepsilon$. Since

$$\frac{1}{\theta_{k+1}} - \frac{1}{\theta_k} = \frac{\theta_k - \theta_{k+1}}{\theta_{k+1}\theta_k} \geq \frac{\theta_k - \theta_{k+1}}{\theta_k^2} \stackrel{(25)}{\geq} \frac{1}{c_1},$$

we have $\frac{1}{\theta_k} \geq \frac{1}{\theta_0} + \frac{k}{c_1} = \frac{1}{\xi_0} + \frac{k}{c_1}$. Therefore, if we let $k_1 \geq \frac{c_1}{\varepsilon} - \frac{c_1}{\xi_0}$, we obtain $\theta_{k_1} \leq \varepsilon$. Finally, letting $k_2 \geq \frac{c_1}{\varepsilon} \log \frac{1}{\rho}$, (23) follows from

$$\theta_K \stackrel{(19)}{\leq} \theta_{k_1+k_2} \stackrel{(26)}{\leq} (1 - \frac{\varepsilon}{c_1})^{k_2}\theta_{k_1} \leq ((1 - \frac{\varepsilon}{c_1})^{\frac{1}{\varepsilon}})^{c_1 \log \frac{1}{\rho}}\varepsilon \leq (e^{-\frac{1}{c_1}})^{c_1 \log \frac{1}{\rho}}\varepsilon = \varepsilon\rho.$$

Now assume that property (ii) holds. Using similar arguments as those leading to (24), we get $\mathbf{E}[\xi_{k+1}^\varepsilon \mid \xi_k^\varepsilon] \leq (1 - \frac{1}{c_2})\xi_k^\varepsilon$ for all k, which implies

$$\theta_K \leq (1 - \frac{1}{c_2})^K\theta_0 = (1 - \frac{1}{c_2})^K\xi_0 \stackrel{(20)}{\leq} ((1 - \frac{1}{c_2})^{c_2})^{\log \frac{\xi_0}{\varepsilon\rho}}\xi_0 \leq (e^{-1})^{\log \frac{\xi_0}{\varepsilon\rho}}\xi_0 = \varepsilon\rho,$$

again establishing (23). $\qquad\square$

The above theorem will be used with $\{x_k\}_{k \geq 0}$ corresponding to the iterates of Algorithm 1 and $\phi(x) = F(x) - F^*$.

Restarting. Note that similar, albeit *slightly weaker,* high probability results can be achieved by *restarting* as follows. We run the random process $\{\xi_k\}$ repeatedly $r = \lceil \log \frac{1}{\rho} \rceil$ times, always starting from ξ_0, each time for the same number of iterations k_1 for which $\mathbf{P}(\xi_{k_1} > \varepsilon) \leq \frac{1}{e}$. It then follows that the probability that all r values ξ_{k_1} will be larger than ε is at most $(\frac{1}{e})^r \leq \rho$. Note that the restarting technique demands that we perform r evaluations of the objective function; this is not needed in the one-shot approach covered by the theorem.

It remains to estimate k_1 in the two cases of Theorem 1. We argue that in case (i) we can choose $k_1 = \lceil \frac{c_1}{\varepsilon/e} - \frac{c_1}{\xi_0} \rceil$. Indeed, using similar arguments as in Theorem 1 this leads to $\mathbf{E}[\xi_{k_1}] \leq \frac{\varepsilon}{e}$, which by Markov inequality implies that in a single run of the process we have

$$\mathbf{P}(\xi_{k_1} > \varepsilon) \leq \frac{\mathbf{E}[\xi_{k_1}]}{\varepsilon} \leq \frac{\varepsilon/e}{\varepsilon} = \frac{1}{e}.$$

Therefore,

$$K = \lceil \frac{ec_1}{\varepsilon} - \frac{c_1}{\xi_0} \rceil \lceil \log \frac{1}{\rho} \rceil$$

iterations suffice in case (i). A similar restarting technique can be applied in case (ii).

Tightness. It can be shown on simple examples that the bounds in the above result are *tight*.

3 Coordinate Descent for Composite Functions

In this section we study the performance of Algorithm 1 in the special case when all probabilities are chosen to be the same, i.e., $p_i = \frac{1}{n}$ for all i. For easier future reference we set this method apart and give it a name (Algorithm 2).

Algorithm 2 UCDC(x_0) (**U**niform **C**oordinate **D**escent for **C**omposite Functions)

---
for $k = 0, 1, 2, \ldots$ **do**
     Choose $i_k = i \in \{1, 2, \ldots, n\}$ with probability $\frac{1}{n}$
     $T^{(i)}(x_k) = \arg\min\{V_i(x_k, t) : t \in \mathbb{R}_i\}$
     $x_{k+1} = x_k + U_i T^{(i)}(x_k)$
end for

The following function plays a central role in our analysis:

$$H(x, T) \overset{\text{def}}{=} f(x) + \langle \nabla f(x), T \rangle + \frac{1}{2} \|T\|_L^2 + \Psi(x + T). \tag{27}$$

Comparing (27) with (17) using (2), (5), (7) and (8) we get

$$H(x, T) = f(x) + \sum_{i=1}^{n} V_i(x, T^{(i)}). \tag{28}$$

Therefore, the vector $T(x) = (T^{(1)}(x), \ldots, T^{(n)}(x))$, with the components $T^{(i)}(x)$ defined in Algorithm 1, is the minimizer of $H(x, \cdot)$:

$$T(x) = \arg\min_{T \in \mathbb{R}^N} H(x, T). \tag{29}$$

Let us start by establishing two auxiliary results which will be used repeatedly.

Lemma 2. *Let $\{x_k\}$, $k \geq 0$, be the random iterates generated by $UCDC(x_0)$. Then*

$$\mathbf{E}[F(x_{k+1}) - F^* \mid x_k] \leq \tfrac{1}{n}(H(x_k, T(x_k)) - F^*) + \tfrac{n-1}{n}(F(x_k) - F^*). \qquad (30)$$

Proof.

$$\begin{aligned}
\mathbf{E}[F(x_{k+1}) \mid x_k] &= \sum_{i=1}^{n} \tfrac{1}{n} F(x_k + U_i T^{(i)}(x_k)) \\
&\overset{(16)}{\leq} \tfrac{1}{n} \sum_{i=1}^{n} [f(x_k) + V_i(x_k, T^{(i)}(x_k)) + C_i(x_k)] \\
&\overset{(28)}{=} \tfrac{1}{n} H(x_k, T(x_k)) + \tfrac{n-1}{n} f(x_k) + \tfrac{1}{n} \sum_{i=1}^{n} C_i(x_k) \\
&\overset{(18)}{=} \tfrac{1}{n} H(x_k, T(x_k)) + \tfrac{n-1}{n} f(x_k) + \tfrac{1}{n} \sum_{i=1}^{n} \sum_{j \neq i} \Psi_j(x_k^{(j)}) \\
&= \tfrac{1}{n} H(x_k, T(x_k)) + \tfrac{n-1}{n} F(x_k).
\end{aligned}$$

$\square$

Lemma 3. *For all $x \in \operatorname{dom} F$ we have $H(x, T(x)) \leq \min_{y \in \mathbb{R}^N} \{F(y) + \frac{1 - \mu_f(L)}{2} \|y - x\|_L^2\}$.*

Proof.

$$\begin{aligned}
H(x, T(x)) &\overset{(29)}{=} \min_{T \in \mathbb{R}^N} H(x, T) = \min_{y \in \mathbb{R}^N} H(x, y - x) \\
&\overset{(27)}{=} \min_{y \in \mathbb{R}^N} f(x) + \langle \nabla f(x), y - x \rangle + \Psi(y) + \tfrac{1}{2} \|y - x\|_L^2 \\
&\overset{(9)}{\leq} \min_{y \in \mathbb{R}^N} f(y) - \tfrac{\mu_f(L)}{2} \|y - x\|_L^2 + \Psi(y) + \tfrac{1}{2} \|y - x\|_L^2.
\end{aligned}$$

$\square$

3.1 Convex Objective

In order for Lemma 2 to be useful, we need to estimate $H(x_k, T(x_k)) - F^*$ from above in terms of $F(x_k) - F^*$.

Lemma 4. *Fix $x^* \in X^*$, $x \in \operatorname{dom} \Psi$ and let $R = \|x - x^*\|_L$. Then*

$$H(x, T(x)) - F^* \leq \begin{cases} \left(1 - \frac{F(x) - F^*}{2R^2}\right)(F(x) - F^*), & \text{if } F(x) - F^* \leq R^2, \\ \tfrac{1}{2} R^2 < \tfrac{1}{2}(F(x) - F^*), & \text{otherwise.} \end{cases} \qquad (31)$$

Proof. Since we do not assume strong convexity, $\mu_f(W) = 0$, and hence

$$H(x, T(x)) \overset{\text{Lemma 3}}{\leq} \min_{y \in \mathbb{R}^N} F(y) + \tfrac{1}{2} \|y - x\|_L^2 \leq \min_{\alpha \in [0,1]} F(\alpha x^* + (1 - \alpha)x) + \tfrac{\alpha^2}{2} \|x - x^*\|_L^2$$

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

Minimizing the last expression gives $\alpha^* = \min\left\{1, \frac{1}{R^2}(F(x) - F^*)\right\}$; the result follows. $\square$

We are now ready to estimate the number of iterations needed to push the objective value within ε of the optimal value with high probability. Note that since ρ appears in the logarithm, it is easy to attain high confidence.

Theorem 5. *Choose initial point x_0 and target confidence $0 < \rho < 1$. Further, let the target accuracy $\varepsilon > 0$ and iteration counter k be chosen in any of the following two ways:*

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

$$k \geq \frac{2n\max\{\mathscr{R}_L^2(x_0), F(x_0) - F^*\}}{\varepsilon}\left(1 + \log\frac{1}{\rho}\right) + 2 - \frac{2n\max\{\mathscr{R}_L^2(x_0), F(x_0) - F^*\}}{F(x_0) - F^*}, \tag{32}$$

(ii) $\varepsilon < \min\{\mathscr{R}_L^2(x_0), F(x_0) - F^*\}$ *and*

$$k \geq \frac{2n\mathscr{R}_L^2(x_0)}{\varepsilon}\log\frac{F(x_0) - F^*}{\varepsilon\rho}. \tag{33}$$

If x_k is the random point generated by $UCDC(x_0)$ as applied to the convex function F, then

$$\mathbf{P}(F(x_k) - F^* \leq \varepsilon) \geq 1 - \rho.$$

Proof. Since $F(x_k) \leq F(x_0)$ for all k, we have $\|x_k - x^*\|_L \leq \mathscr{R}_L(x_0)$ for all $x^* \in X^*$. Plugging the inequality (31) (Lemma 2) into (30) (Lemma 4) then gives that the following holds for all k:

$$\mathbf{E}[F(x_{k+1}) - F^* \mid x_k] \leq \frac{1}{n}\max\left\{1 - \frac{F(x_k) - F^*}{2\|x_k - x^*\|_L^2}, \frac{1}{2}\right\}(F(x_k) - F^*) + \frac{n-1}{n}(F(x_k) - F^*)$$

$$= \max\left\{1 - \frac{F(x_k) - F^*}{2n\|x_k - x^*\|_L^2}, 1 - \frac{1}{2n}\right\}(F(x_k) - F^*)$$

$$\leq \max\left\{1 - \frac{F(x_k) - F^*}{2n\mathscr{R}_L^2(x_0)}, 1 - \frac{1}{2n}\right\}(F(x_k) - F^*). \tag{34}$$

Let $\xi_k = F(x_k) - F^*$ and consider case (i). If we let $c_1 = 2n\max\{\mathscr{R}_L^2(x_0), F(x_0) - F^*\}$, then from (34) we obtain

$$\mathbf{E}[\xi_{k+1} \mid x_k] \leq (1 - \frac{\xi_k}{c_1})\xi_k = \xi_k - \frac{\xi_k^2}{c_1}, \qquad k \geq 0.$$

Moreover, $\varepsilon < \xi_0 < c_1$. The result then follows by applying Theorem 1. Consider now case (ii). Letting $c_2 = \frac{2n\mathscr{R}_L^2(x_0)}{\varepsilon} > 1$, notice that if $\xi_k \geq \varepsilon$, inequality (34) implies that

$$\mathbf{E}[\xi_{k+1} \mid x_k] \leq \max\left\{1 - \frac{\varepsilon}{2n\mathscr{R}_L^2(x_0)}, 1 - \frac{1}{2n}\right\}\xi_k = (1 - \frac{1}{c_2})\xi_k.$$

Again, the result follows from Theorem 1. $\square$

3.2 Strongly Convex Objective

The following lemma will be useful in proving linear convergence of the expected value of the objective function to the minimum.

Lemma 6. *If $\mu_f(L) + \mu_\Psi(L) > 0$, then for all $x \in \mathrm{dom}\, F$ we have*

$$H(x, T(x)) - F^* \leq \frac{1 - \mu_f(L)}{1 + \mu_\Psi(L)}(F(x) - F^*). \tag{35}$$

Proof. Letting $\mu_f = \mu_f(L)$, $\mu_\Psi = \mu_\Psi(L)$ and $\alpha^* = (\mu_f + \mu_\Psi)/(1 + \mu_\Psi) \le 1$, we have

$$
H(x, T(x)) \overset{\text{Lem 3}}{\le} \min_{y \in \mathbb{R}^N} F(y) + \frac{1 - \mu_f}{2} \|y - x\|_L^2
$$

$$
\le \min_{\alpha \in [0,1]} F(\alpha x^* + (1 - \alpha)x) + \frac{(1 - \mu_f)\alpha^2}{2} \|x - x^*\|_L^2
$$

$$
\overset{(11)+(10)}{\le} \min_{\alpha \in [0,1]} \alpha F^* + (1 - \alpha)F(x) - \frac{(\mu_f + \mu_\Psi)\alpha(1 - \alpha) - (1 - \mu_f)\alpha^2}{2} \|x - x^*\|_L^2
$$

$$
\le F(x) - \alpha^*(F(x) - F^*).
$$

The last inequality follows from the identity $(\mu_f + \mu_\Psi)(1 - \alpha^*) - (1 - \mu_f)\alpha^* = 0$. $\qquad\square$

A modification of the above lemma (and of the subsequent results using it) is possible where the assumption $\mu_f(L) + \mu_\Psi(L) > 0$ replaced by the slightly weaker assumption $\mu_F(L) > 0$. Indeed, in the third inequality in the proof one can replace $\mu_f + \mu_\Psi$ by μ_F; the estimate (35) gets improved a bit. However, we prefer the current version for reasons of simplicity of exposition.

We now show that the expected value of $F(x_k)$ converges to F^* linearly.

Theorem 7. *Assume $\mu_f(L) + \mu_\Psi(L) > 0$ and choose initial point x_0. If x_k is the random point generated UCDC(x_0), then*

$$
\mathbf{E}[F(x_k) - F^*] \le \left(1 - \frac{1}{n}\frac{\mu_f(L) + \mu_\Psi(L)}{1 + \mu_\Psi(L)}\right)^k (F(x_0) - F^*). \tag{36}
$$

Proof. Follows from Lemma 2 and Lemma 6. $\qquad\square$

The following is an analogue of Theorem 5 in the case of a strongly convex objective. Note that both the accuracy and confidence parameters appear in the logarithm.

Theorem 8. *Assume $\mu_f(L) + \mu_\Psi(L) > 0$. Choose initial point x_0, target accuracy level $0 < \varepsilon < F(x_0) - F^*$, target confidence level $0 < \rho < 1$, and*

$$
k \ge n\frac{1 + \mu_\Psi(L)}{\mu_f(L) + \mu_\Psi(L)} \log\left(\frac{F(x_0) - F^*}{\varepsilon\rho}\right). \tag{37}
$$

If x_k is the random point generated by UCDC(x_0), then

$$
\mathbf{P}(F(x_k) - F^* \le \varepsilon) \ge 1 - \rho.
$$

Proof. Using Markov inequality and Theorem 7, we obtain

$$
\mathbf{P}[F(x_k) - F^* \ge \varepsilon] \le \frac{1}{\varepsilon}\mathbf{E}[F(x_k) - F^*] \overset{(36)}{\le} \frac{1}{\varepsilon}\left(1 - \frac{1}{n}\frac{\mu_f(L) + \mu_\Psi(L)}{1 + \mu_\Psi(L)}\right)^k (F(x_0) - F^*) \overset{(37)}{\le} \rho.
$$

$\qquad\square$

Let us rewrite the *condition number* appearing in the complexity bound (37) in a more natural form:

$$
\frac{1 + \mu_\Psi(L)}{\mu_f(L) + \mu_\Psi(L)} \overset{(15)}{=} \frac{\operatorname{tr}(L)/n + \mu_\Psi(\widetilde{L})}{\mu_f(\widetilde{L}) + \mu_\Psi(\widetilde{L})} \le 1 + \frac{\operatorname{tr}(L)/n}{\mu_f(\widetilde{L}) + \mu_\Psi(\widetilde{L})}. \tag{38}
$$

Hence, it is (up to the constant 1) equal to the ratio of the *average* of the Lipschitz constants L_i and the (strong) convexity parameter of the objective function F with respect to the (normalized) norm $\|\cdot\|_{\widetilde{L}}$.

3.3 A Regularization Technique

In this section we investigate an alternative approach to establishing an iteration complexity result in the case of an objective function that is not strongly convex. The strategy is very simple. We first regularize the objective function by adding a small quadratic term to it, thus making it strongly convex, and then argue that when Algorithm 2 is applied to the regularized objective, we can recover an approximate solution of the original non-regularized problem. This approach was used in [13] to obtain iteration complexity results for a randomized block coordinate descent method applied to a smooth function. Here we use the same idea outlined above with the following differences: i) our proof is different, ii) we get a better complexity result, and iii) our approach works also in the composite setting.

Fix x_0 and $\varepsilon > 0$ and consider a regularized version of the objective function defined by

$$F_\mu(x) \overset{\text{def}}{=} F(x) + \tfrac{\mu}{2}\|x - x_0\|_L^2, \qquad \mu = \tfrac{\varepsilon}{\|x_0 - x^*\|_L^2}. \tag{39}$$

Clearly, F_μ is strongly convex with respect to the norm $\|\cdot\|_L$ with convexity parameter $\mu_{F_\mu}(L) = \mu$. In the rest of this subsection we show that if we apply $\text{UCDC}(x_0)$ to F_μ with target accuracy $\frac{\varepsilon}{2}$, then with high probability we recover an ε-approximate solution of (1). Note that μ is *not known* in advance since x^* is not known. This means that any iteration complexity result obtained by applying our algorithm to the objective F_μ will *not* lead to a *true/valid* iteration complexity bound unless a bound on $\|x_0 - x^*\|_L$ is available.

We first need to establish that an approximate minimizer of F_μ must be an approximate minimizer of F.

Lemma 9. *If x' satisfies $F_\mu(x') \leq \min_{x\in\mathbb{R}^N} F_\mu(x) + \frac{\varepsilon}{2}$, then $F(x') \leq F^* + \varepsilon$.*

Proof. Clearly,

$$F(x) \leq F_\mu(x), \qquad x \in \mathbb{R}^N. \tag{40}$$

If we let $x_\mu^* \overset{\text{def}}{=} \arg\min_{x\in\mathbb{R}^N} F_\mu(x)$ then, by assumption,

$$F_\mu(x') - F_\mu(x_\mu^*) \leq \tfrac{\varepsilon}{2}, \tag{41}$$

$$F_\mu(x_\mu^*) = \min_{x\in\mathbb{R}^N} F(x) + \tfrac{\mu}{2}\|x - x_0\|_L^2 \leq F(x^*) + \tfrac{\mu}{2}\|x^* - x_0\|_L^2 \overset{(39)}{\leq} F(x^*) + \tfrac{\varepsilon}{2}. \tag{42}$$

Putting all these observations together, we get

$$0 \leq F(x') - F(x^*) \overset{(40)}{\leq} F_\mu(x') - F(x^*) \overset{(41)}{\leq} F_\mu(x_\mu^*) + \tfrac{\varepsilon}{2} - F(x^*) \overset{(42)}{\leq} \varepsilon.$$

$\square$

The following theorem is an analogue of Theorem 5. The result we obtain in this way is slightly different to the one given in Theorem 5 in that $2n\mathscr{R}_L^2(x_0)/\varepsilon$ is replaced by $n(1 + \|x_0 - x^*\|_L^2/\varepsilon)$. In some situations, $\|x_0 - x^*\|_L^2$ can be significantly smaller than $\mathscr{R}_L^2(x_0)$.

Theorem 10. *Choose initial point x_0, target accuracy level*

$$0 < \varepsilon \leq 2(F(x_0) - F^*), \tag{43}$$

target confidence level $0 < \rho < 1$, and

$$k \geq n\left(1 + \tfrac{\|x_0 - x^*\|_L^2}{\varepsilon}\right) \log\left(\tfrac{2(F(x_0) - F^*)}{\varepsilon\rho}\right). \tag{44}$$

If x_k is the random point generated by $UCDC(x_0)$ as applied to F_μ, then

$$\mathbf{P}(F(x_k) - F^* \leq \varepsilon) \geq 1 - \rho.$$

Proof. Let us apply Theorem 8 to the problem of minimizing F_μ, composed as $f + \Psi_\mu$, with $\Psi_\mu(x) = \Psi(x) + \frac{\mu}{2}\|x - x_0\|_L^2$, with accuracy level $\frac{\varepsilon}{2}$. Note that $\mu_{\Psi_\mu}(L) = \mu$,

$$F_\mu(x_0) - F_\mu(x_\mu^*) \stackrel{(39)}{=} F(x_0) - F_\mu(x_\mu^*) \stackrel{(40)}{\leq} F(x_0) - F(x_\mu^*) \leq F(x_0) - F^*, \qquad (45)$$

$$n(1 + \tfrac{1}{\mu}) \stackrel{(39)}{=} n\left(1 + \frac{\|x_0 - x^*\|_L^2}{\varepsilon}\right). \qquad (46)$$

Comparing (37) and (44) in view of (45) and (46), Theorem 8 implies that

$$\mathbf{P}(F_\mu(x_k) - F_\mu(x_\mu^*) \leq \tfrac{\varepsilon}{2}) \geq 1 - \rho.$$

It now suffices to apply Lemma 9. □

4 Coordinate Descent for Smooth Functions

In this section we give a much simplified and improved treatment of the smooth case ($\Psi \equiv 0$) as compared to the analysis in Sections 2 and 3 of [13].

As alluded to in the above, we will develop the analysis in the smooth case for arbitrary, possibly non-Euclidean, norms $\|\cdot\|_{(i)}$, $i = 1, 2, \ldots, n$. Let $\|\cdot\|$ be an arbitrary norm in $\mathbb{R}^l$. Then its dual is defined in the usual way:

$$\|s\|^* = \max_{\|t\|=1} \langle s, t \rangle.$$

The following (Lemma 11) is a simple result which is used in [13] without being fully articulated nor proved as it constitutes a straightforward extension of a fact that is trivial in the Euclidean setting to the case of general norms. Since we will also need to use it, and because we think it is perhaps not standard, we believe it deserves to be spelled out explicitly. Note that the main problem which needs to be solved at each iteration of Algorithm 1 in the smooth case is of the form (47), with $s = -\frac{1}{L_i}\nabla_i f(x_k)$ and $\|\cdot\| = \|\cdot\|_{(i)}$.

Lemma 11. *If by $s^{\#}$ we denote an optimal solution of the problem*

$$\min_t \left\{ u(s) \stackrel{def}{=} -\langle s, t \rangle + \tfrac{1}{2}\|t\|^2 \right\}, \qquad (47)$$

then

$$u(s^{\#}) = -\tfrac{1}{2}(\|s\|^*)^2, \qquad \|s^{\#}\| = \|s\|^*, \qquad (\alpha s)^{\#} = \alpha(s^{\#}), \ \alpha \in \mathbb{R}. \qquad (48)$$

Proof. For $\alpha = 0$ the last statement is trivial. If we fix $\alpha \neq 0$, then clearly

$$u((\alpha s)^{\#}) = \min_{\|t\|=1} \min_\beta \{-\langle \alpha s, \beta t \rangle + \tfrac{1}{2}\|\beta t\|^2\}.$$

For fixed t the solution of the inner problem is $\beta = \langle \alpha s, t \rangle$, whence

$$u((\alpha s)^{\#}) = \min_{\|t\|=1} -\tfrac{1}{2}\langle \alpha s, t \rangle^2 = -\tfrac{1}{2}\alpha^2 \left(\max_{\|t\|=1} \langle s, t \rangle \right)^2 = -\tfrac{1}{2}(\|\alpha s\|^*)^2, \qquad (49)$$

proving the first claim. Next, note that optimal $t = t^*$ in (49) maximizes $\langle s, t \rangle$ over $\|t\| = 1$. Therefore, $\langle s, t^* \rangle = \|s\|^*$, which implies that $\|(\alpha s)^{\#}\| = |\beta^*| = |\langle \alpha s, t^* \rangle| = |\alpha||\langle s, t^* \rangle| = |\alpha|\|s\|^* = \|\alpha s\|^*$, giving the second claim. Finally, since t^* depends on s only, we have $(\alpha s)^{\#} = \beta^* t^* = \langle \alpha s, t^* \rangle t^*$ and, in particular, $s^{\#} = \langle s, t^* \rangle t^*$. Therefore, $(\alpha s)^{\#} = \alpha(s^{\#})$. □

We can use Lemma 11 to rewrite the main step of Algorithm 1 in the smooth case into the more explicit form,

$$T^{(i)}(x) = \arg\min_{t\in\mathbb{R}_i} V_i(x,t) \overset{(17)}{=} \arg\min_{t\in\mathbb{R}_i} \langle \nabla_i f(x), t\rangle + \tfrac{L_i}{2}\|t\|_{(i)}^2$$

$$\overset{(47)}{=} \left(-\tfrac{\nabla_i f(x)}{L_i}\right)^{\#} \overset{(48)}{=} -\tfrac{1}{L_i}(\nabla_i f(x))^{\#},$$

leading to Algorithm 3.

Algorithm 3 $\mathrm{RCDS}(p,x_0)$ (**R**andomized **C**oordinate **D**escent for **S**mooth Functions)

for $k = 0,1,2,\dots$ **do**
  Choose $i_k = i \in \{1,2,\dots,n\}$ with probability p_i
  $x_{k+1} = x_k - \tfrac{1}{L_i} U_i (\nabla_i f(x_k))^{\#}$
end for

The main utility of Lemma 11 for the purpose of the subsequent complexity analysis comes from the fact that it enables us to give an *explicit* bound on the decrease in the objective function during one iteration of the method in the same form as in the Euclidean case:

$$f(x) - f(x + U_i T^{(i)}(x)) \overset{(6)}{\geq} -[\langle \nabla_i f(x), T^{(i)}(x)\rangle + \tfrac{L_i}{2}\|T^{(i)}(x)\|_{(i)}^2]$$

$$= -L_i u\left((-\tfrac{\nabla_i f(x)}{L_i})^{\#}\right) \tag{50}$$

$$\overset{(48)}{=} \tfrac{L_i}{2}(\| -\tfrac{\nabla_i f(x)}{L_i}\|_{(i)}^*)^2 = \tfrac{1}{2L_i}(\|\nabla_i f(x)\|_{(i)}^*)^2.$$

4.1 Convex Objective

We are now ready to state the main result of this section.

Theorem 12. *Choose initial point x_0, target accuracy $0 < \varepsilon < \min\{f(x_0) - f^*, 2\mathcal{R}_{LP^{-1}}^2(x_0)\}$, target confidence $0 < \rho < 1$ and*

$$k \geq \frac{2\mathcal{R}_{LP^{-1}}^2(x_0)}{\varepsilon}\left(1 + \log\tfrac{1}{\rho}\right) + 2 - \frac{2\mathcal{R}_{LP^{-1}}^2(x_0)}{f(x_0) - f^*}, \tag{51}$$

or

$$k \geq \frac{2\mathcal{R}_{LP^{-1}}^2(x_0)}{\varepsilon}\left(1 + \log\tfrac{1}{\rho}\right) - 2. \tag{52}$$

If x_k is the random point generated by $\mathrm{RCDS}(p,x_0)$ as applied to convex f, then

$$\mathbf{P}(f(x_k) - f^* \leq \varepsilon) \geq 1 - \rho.$$

Proof. Let us first estimate the expected decrease of the objective function during one iteration of the method:

$$f(x_k) - \mathbf{E}[f(x_{k+1}) \mid x_k] = \sum_{i=1}^{n} p_i[f(x_k) - f(x_k + U_i T^{(i)}(x_k))]$$

$$\overset{(50)}{\geq} \tfrac{1}{2}\sum_{i=1}^{n} p_i \tfrac{1}{L_i}(\|\nabla_i f(x_k)\|_{(i)}^*)^2 = \tfrac{1}{2}(\|\nabla f(x_k)\|_W^*)^2,$$

where $W = LP^{-1}$. Since $f(x_k) \leq f(x_0)$ for all k and because f is convex, we get $f(x_k) - f^* \leq \max_{x^* \in X^*} \langle \nabla f(x_k), x_k - x^* \rangle \leq \|\nabla f(x_k)\|_W^* \mathscr{R}_W(x_0)$, whence

$$f(x_k) - \mathbf{E}[f(x_{k+1}) \mid x_k] \geq \tfrac{1}{2} \left(\tfrac{f(x_k) - f^*}{\mathscr{R}_W(x_0)} \right)^2.$$

By rearranging the terms we obtain

$$\mathbf{E}[f(x_{k+1}) - f^* \mid x_k] \leq f(x_k) - f^* - \tfrac{(f(x_k) - f^*)^2}{2\mathscr{R}_W^2(x_0)}.$$

If we now use Theorem 1 with $\xi_k = f(x_k) - f^*$ and $c_1 = 2\mathscr{R}_W^2(x_0)$, we obtain the result for k given by (51). We now claim that $2 - \frac{c_1}{\xi_0} \leq -2$, from which it follows that the result holds for k given by (52). Indeed, first notice that this inequality is equivalent to

$$f(x_0) - f^* \leq \tfrac{1}{2}\mathscr{R}_W^2(x_0). \tag{53}$$

Now, a straightforward extension of Lemma 2 in [13] to general weights states that ∇f is Lipschitz with respect to the norm $\| \cdot \|_V$ with the constant $\mathrm{tr}(LV^{-1})$. This, in turn, implies the inequality

$$f(x) - f^* \leq \tfrac{1}{2}\mathrm{tr}(LV^{-1})\|x - x^*\|_V^2,$$

from which (53) follows by setting $V = W$ and $x = x_0$. $\qquad\square$

4.2 Strongly Convex Objective

Assume now that f is strongly convex with respect to the norm $\| \cdot \|_{LP^{-1}}$ (see definition (9)) with convexity parameter $\mu_f(LP^{-1}) > 0$. Using (9) with $x = x^*$ and $y = x_k$, we obtain

$$f^* - f(x_k) \geq \langle \nabla f(x_k), h \rangle + \tfrac{\mu_f(LP^{-1})}{2}\|h\|_{LP^{-1}}^2$$
$$= \mu_f(LP^{-1}) \left(\langle \tfrac{1}{\mu_f(LP^{-1})} \nabla f(x_k), h \rangle + \tfrac{1}{2}\|h\|_{LP^{-1}}^2 \right),$$

where $h = x^* - x_k$. Applying Lemma 11 to estimate the right hand side of the above inequality from below, we obtain

$$f^* - f(x_k) \geq -\tfrac{1}{2\mu_f(LP^{-1})}(\|\nabla f(x_k)\|_{LP^{-1}}^*)^2. \tag{54}$$

Let us now write down an efficiency estimate for the case of a strongly convex objective.

Theorem 13. *Let F be strongly convex with respect to $\| \cdot \|_{LP^{-1}}$ with convexity parameter $\mu_f(LP^{-1}) > 0$. Choose initial point x_0, target accuracy $0 < \varepsilon < f(x_0) - f^*$, target confidence $0 < \rho < 1$ and*

$$k \geq \tfrac{1}{\mu_f(LP^{-1})} \log \tfrac{f(x_0) - f^*}{\varepsilon\rho}. \tag{55}$$

If x_k is the random point generated by $RCDS(p, x_0)$ as applied to f, then

$$\mathbf{P}(f(x_k) - f^* \leq \varepsilon) \geq 1 - \rho.$$

Proof. The expected decrease of the objective function during one iteration of the method can be estimated as follows:

$$
\begin{aligned}
f(x_k) - \mathbf{E}[f(x_{k+1}) \mid x_k] &= \sum_{i=1}^{n} p_i [f(x_k) - f(x_k + U_i T^{(i)}(x_k))] \\
&\overset{(50)}{\geq} \tfrac{1}{2} \sum_{i=1}^{n} p_i \tfrac{1}{L_i} (\|\nabla_i f(x_k)\|_{(i)}^*)^2 \\
&= \tfrac{1}{2} (\|\nabla f(x_k)\|_{LP^{-1}}^*)^2 \\
&\overset{(54)}{\geq} \mu_f(LP^{-1})(f(x_k) - f^*).
\end{aligned}
$$

After rearranging the terms we obtain $\mathbf{E}[f(x_{k+1}) - f^* \mid x_k] \leq (1 - \mu_f(LP^{-1}))\mathbf{E}[f(x_k) - f^*]$. It now remains to use part (ii) of Theorem 1 with $\xi_k = f(x_k) - f^*$ and $c_2 = \frac{1}{\mu_f(LP^{-1})}$. $\qquad \square$

The leading factor $\frac{1}{\mu_f(LP^{-1})}$ in the complexity bound (55) can in special cases be written in a more natural form; we now give two examples.

1. *Uniform probabilities.* If $p_i = \frac{1}{n}$ for all i, then

$$
\frac{1}{\mu_f(LP^{-1})} = \frac{1}{\mu_f(nL)} \overset{(13)}{=} \frac{n}{\mu_f(L)} \overset{(38)}{=} \frac{\operatorname{tr}(L)}{\mu_f(\tilde{L})} = n \frac{\operatorname{tr}(L)/n}{\mu_f(\tilde{L})}.
$$

2. *Probabilities proportional to the Lipschitz constants.* If $p_i = \frac{L_i}{\operatorname{tr}(L)}$ for all i, then

$$
\frac{1}{\mu_f(LP^{-1})} = \frac{1}{\mu_f(\operatorname{tr}(L)I)} \overset{(13)}{=} \frac{\operatorname{tr}(L)}{\mu_f(I)} = n \frac{\operatorname{tr}(L)/n}{\mu_f(I)}.
$$

In both cases, $\frac{1}{\mu_f(LP^{-1})}$ is equal to n multiplied by a *condition number* of the form $\frac{\operatorname{tr}(L)/n}{\mu_f(W)}$, where the numerator is the average of the Lipschitz constants $L_1, \dots, L_n$, W is a diagonal matrix of weights summing up to n and $\mu_f(W)$ is the (strong) convexity parameter of f with respect to $\|\cdot\|_W$.

5 Comparison of CD Methods with Complexity Guarantees

In this section we compare the results obtained in this paper with existing CD methods endowed with iteration complexity bounds.

5.1 Smooth case ($\Psi = 0$)

In Table 2 we look at the results for unconstrained smooth minimization of Nesterov [13] and contrast these with our approach. For brevity we only include results for the non-strongly convex case. We will now comment on the contents of Table 2 in detail.

1. *Uniform probabilities.* Note that in the uniform case ($p_i = \frac{1}{n}$ for all i) we have

$$
\mathscr{R}_{LP^{-1}}^2(x_0) = n\mathscr{R}_L^2(x_0),
$$

Algorithm	Ψ	p_i	Norms	Complexity	Objective
Nesterov [13] (Theorem 4)	0	$\frac{L_i}{\sum_i L_i}$	Eucl.	$(2n + \frac{8(\sum_i L_i)\mathscr{R}_I^2(x_0)}{\varepsilon}) \log \frac{4(f(x_0)-f^*)}{\varepsilon\rho}$	$f(x) + \frac{\varepsilon\|x-x_0\|_I^2}{8\mathscr{R}_I^2(x_0)}$
Nesterov [13] (Theorem 3)	0	$\frac{1}{n}$	Eucl.	$\frac{8n\mathscr{R}_I^2(x_0)}{\varepsilon} \log \frac{4(f(x_0)-f^*)}{\varepsilon\rho}$	$f(x) + \frac{\varepsilon\|x-x_0\|_I^2}{8\mathscr{R}_L^2(x_0)}$
Algorithm 3 (Theorem 12)	0	> 0	general	$\frac{2\mathscr{R}_{LP^{-1}}^2(x_0)}{\varepsilon}(1 + \log\frac{1}{\rho}) - 2$	$f(x)$
Algorithm 2 (Theorem 5)	separable	$\frac{1}{n}$	Eucl.	$\frac{2n\max\{\mathscr{R}_L^2(x_0),F(x_0)-F^*\}}{\varepsilon}(1+\log\frac{1}{\rho})$ $\frac{2n\mathscr{R}_L^2(x_0)}{\varepsilon}\log\frac{F(x_0)-F^*}{\varepsilon\rho}$	$F(x)$

Table 2 Comparison of our results to the results in [13] in the non-strongly convex case. The complexity is for achieving $\mathbf{P}(F(x_k) - F^* \le \varepsilon) \ge 1 - \rho$.

and hence the leading term (ignoring the logarithmic factor) in the complexity estimate of Theorem 12 (line 3 of Table 2) coincides with the leading term in the complexity estimate of Theorem 5 (line 4 of Table 2; the second result): in both cases it is

$$\frac{2n\mathscr{R}_L^2(x_0)}{\varepsilon}.$$

Note that the leading term of the complexity estimate given in Theorem 3 of [13] (line 2 of Table 2), which covers the uniform case, is worse by a factor of 4.

2. *Probabilities proportional to Lipschitz constants.* If we set $p_i = \frac{L_i}{\text{tr}(L)}$ for all i, then

$$\mathscr{R}_{LP^{-1}}^2(x_0) = \text{tr}(L)\mathscr{R}_I^2(x_0).$$

In this case Theorem 4 in [13] (line 1 of Table 2) gives the complexity bound $2[n + \frac{4\text{tr}(L)\mathscr{R}_I^2(x_0)}{\varepsilon}]$ (ignoring the logarithmic factor), whereas we obtain the bound $\frac{2\text{tr}(L)\mathscr{R}_I^2(x_0)}{\varepsilon}$ (line 3 of Table 2), an improvement by a factor of 4. Note that there is a further additive decrease by the constant $2n$ (*and* the additional constant $\frac{2\mathscr{R}_{LP^{-1}}^2(x_0)}{f(x_0)-f^*} - 2$ if we look at the sharper bound (51)).

3. *General probabilities.* Note that unlike the results in [13], which cover the choice of two probability vectors only (lines 1 and 2 of Table 2)—uniform and proportional to L_i— our result (line 3 of Table 2) covers the case of arbitrary probability vector p. This opens the possibility for fine-tuning the choice of p, in certain situations, so as to minimize $\mathscr{R}_{LP^{-1}}^2(x_0)$.

4. *Logarithmic factor.* Note that in our results we have managed to push ε out of the logarithm.

5. *Norms.* Our results hold for general norms.

6. *No need for regularization.* Our results hold for applying the algorithms to F directly; i.e., there is no need to first regularize the function by adding a small quadratic term to it (in a similar fashion as we have done it in Section 3.3). This is an essential feature as the regularization constants are not known and hence the complexity results obtained that way are not true/valid complexity results.

5.2 Nonsmooth case ($\Psi \neq 0$)

In Table 3 we summarize the main characteristics of known complexity results for coordinate (or block coordinate) descent methods for minimizing composite functions. Note that the methods of Saha & Tewari and Shalev-Shwartz & Tewari cover the ℓ_1 regularized case only, whereas the other methods cover the general block-separable case. However, while the greedy approach of Yun & Tseng requires per-iteration work which grows with increasing problem dimension, our randomized strategy can be implemented cheaply. This gives an important advantage to randomized methods for problems of large enough size.

Algorithm	Lipschitz constant(s)	Ψ	block	Choice of coordinate	Work per 1 iteration
Yun & Tseng [22]	$L(\nabla f)$	separable	Yes	greedy	expensive
Saha & Tewari [17]	$L(\nabla f)$	$\|\cdot\|_1$	No	cyclic	cheap
Shalev-Shwartz & Tewari [18]	$\beta = \max_i L_i$	$\|\cdot\|_1$	No	$\frac{1}{n}$	cheap
This paper (Algorithm 2)	L_i	separable	Yes	$\frac{1}{n}$	cheap

Table 3 Comparison of CD approaches for minimizing composite functions (for which iteration complexity results are provided).

Algorithm	complexity	complexity (expanded)
Yun & Tseng [22]	$O(\frac{nL(\nabla f)\|x^*-x_0\|_2^2}{\varepsilon})$	$O(\frac{n}{\varepsilon}\sum_i L(\nabla f)(u^{(i)})^2)$
Saha & Tewari [17]	$O(\frac{nL(\nabla f)\|x^*-x_0\|_2^2}{\varepsilon})$	$O(\frac{n}{\varepsilon}\sum_i L(\nabla f)(u^{(i)})^2)$
Shwartz & Tewari [18]	$O(\frac{n\beta\|x^*-x_0\|_2^2}{\varepsilon})$	$O(\frac{n}{\varepsilon}\sum_i (\max_i L_i)(u^{(i)})^2)$
This paper (Algorithm 2)	$O(\frac{n\|x^*-x_0\|_L^2}{\varepsilon})$	$O(\frac{n}{\varepsilon}\sum_i L_i(u^{(i)})^2)$

Table 4 Comparison of iteration complexities of the methods listed in Table 3. The complexity in the case of the randomized methods gives iteration counter k for which $\mathbf{E}(F(x_k) - F^*) \leq \varepsilon$.

The methods of Yun & Tseng and Saha & Tewari use one Lipschitz constant only, the Lipschitz constant $L(\nabla f)$ of the gradient of f with respect to the standard Euclidean norm. Note that $\max_i L_i \leq L(\nabla f) \leq \sum_i L_i$. If n is large, this constant is typically much larger than

the (block) coordinate constants L_i. Shalev-Shwartz & Tewari use coordinate Lipschitz constants, but assume that all of them are the same. This is suboptimal as in many applications the constants $\{L_i\}$ will have a large variation and hence if one chooses $\beta = \max_i L_i$ for the common Lipschitz constant, steplengths will necessarily be small (see Figure 2 in Section 6).

Let us now compare the impact of the Lipschitz constants on the complexity estimates. For simplicity assume $N = n$ and let $u = x^* - x_0$. The estimates are listed in Table 4. It is clear from the last column that the the approach with individual constants L_i for each coordinate gives the best complexity.

6 Numerical Experiments

In this section we study the numerical behavior of RCDC on synthetic and real problem instances of two problem classes: Sparse Regression / Lasso [20] (Section 6.1) and Linear Support Vector Machines (Section 6.2). As an important concern in Section 6.1 is to demonstrate that our methods scale well with size, our algorithms were written in C and all experiments were run on a PC with 480GB RAM.

6.1 Sparse Regression / Lasso

Consider the problem
$$\min_{x \in \mathbb{R}^n} \tfrac{1}{2} \|Ax - b\|_2^2 + \lambda \|x\|_1, \tag{56}$$

where $A = [a_1, \ldots, a_n] \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $\lambda \geq 0$. The parameter λ is used to induce sparsity in the resulting solution. Note that (56) is of the form (1), with $f(x) = \tfrac{1}{2}\|Ax - b\|_2^2$ and $\Psi(x) = \lambda \|x\|_1$. Moreover, if we let $N = n$ and $U_i = e_i$ for all i, then the Lipschitz constants L_i can be computed explicitly:

$$L_i = \|a_i\|_2^2.$$

Computation of $t = T^{(i)}(x)$ reduces to the "soft-thresholding" operator [28]. In some of the experiments in this section we will allow the probability vector p to change throughout the iterations even though we do not give a theoretical justification for this. With this modification, a direct specialization of RCDC to (56) takes the form of Algorithm 4. If uniform probabilities are used throughout, we refer to the method as UCDC.

Algorithm 4 RCDC for Sparse Regression

Choose $x_0 \in \mathbb{R}^n$ and set $g_0 = Ax_0 - b = -b$
for $k = 0, 1, 2, \ldots$ **do**
    Choose $i_k = i \in \{1, 2, \ldots, n\}$ with probability $p_k^{(i)}$
    $\alpha = a_i^T g_k$
$$t = \begin{cases} -\frac{\alpha + \lambda}{\|a_i\|_2^2}, & \text{if } x_k^{(i)} - \frac{\alpha + \lambda}{\|a_i\|_2^2} > 0 \\ -\frac{\alpha - \lambda}{\|a_i\|_2^2}, & \text{if } x_k^{(i)} - \frac{\alpha - \lambda}{\|a_i\|_2^2} < 0 \\ -x_k^{(i)}, & \text{otherwise} \end{cases}$$
    $x_{k+1} = x_k + t e_i, \quad g_{k+1} = g_k + t a_i$
end for

Instance generator

In order to be able to test Algorithm 4 under controlled conditions we use a (variant of the) instance generator proposed in Section 6 of [12] (the generator was presented for $\lambda = 1$ but can be easily extended to any $\lambda > 0$). In it, one chooses the sparsity level of A and the optimal solution x^*; after that A, b, x^* and $F^* = F(x^*)$ are generated. For details we refer the reader to the aforementioned paper.

In what follows we use the notation $\|A\|_0$ and $\|x\|_0$ to denote the number of nonzero elements of matrix A and of vector x, respectively.

Speed versus sparsity

In the first experiment we investigate, on problems of size $m = 10^7$ and $n = 10^6$, the dependence of the time it takes for UCDC to complete a block of n iterations (the measurements were done by running the method for $10 \times n$ iterations and then dividing by 10) on the sparsity levels of A and x^*. Looking at Table 5, we see that the speed of UCDC depends roughly linearly on the sparsity level of A (and does not depend on $\|x^*\|_0$ at all). Indeed, as $\|A\|_0$ increases from 10^7 through 10^8 to 10^9, the time it takes for the method to complete n iterations increases from about 0.9s through 4–6s to about 46 seconds. This is to be expected since the amount of work per iteration of the method in which coordinate i is chosen is proportional to $\|a_i\|_0$ (computation of α, $\|a_i\|_2^2$ and g_{k+1}).

$\|x^*\|_0$	$\|A\|_0 = 10^7$	$\|A\|_0 = 10^8$	$\|A\|_0 = 10^9$
16×10^2	0.89	5.89	46.23
16×10^3	0.85	5.83	46.07
16×10^4	0.86	4.28	46.93

Table 5 The time it takes for UCDC to complete a block of n iterations increases linearly with $\|A\|_0$ and does not depend on $\|x^*\|_0$.

Efficiency on huge-scale problems

Tables 6 and 7 present typical results of the performance of UCDC, started from $x_0 = 0$, on synthetic sparse regression instances of big/huge size. The instance in the first table is of size $m = 2 \times 10^7$ and $n = 10^6$, with A having 5×10^7 nonzeros and the support of x^* being of size $160,000$.

In both tables the first column corresponds to the "full-pass" iteration counter k/n. That is, after $k = n$ coordinate iterations the value of this counter is 1, reflecting a single "pass" through the coordinates. The remaining columns correspond to, respectively, the size of the current residual $F(x_k) - F^*$ relative to the initial residual $F(x_0) - F^*$, size $\|x_k\|_0$ of the support of the current iterate x_k, and time (in seconds). A row is added whenever the residual initial residual is decreased by an additional factor of 10.

Let us first look at the smaller of the two problems (Table 6). After $35 \times n$ coordinate iterations, UCDC decreases the initial residual by a factor of 10^{18}, and this takes about a minute and a half. Note that the number of nonzeros of x_k has stabilized at this point at $160,000$, the support size of the optima solution. The method has managed to identify the support. After 139.4 seconds the residual is decreased by a factor of 10^{29}. This surprising

$A \in \mathbb{R}^{2 \cdot 10^7 \times 10^6}$, $\|A\|_0 = 5 \cdot 10^7$							
k/n	$\frac{F(x_k)-F^*}{F(x_0)-F^*}$	$\|x_k\|_0$	time [sec]	k/n	$\frac{F(x_k)-F^*}{F(x_0)-F^*}$	$\|x_k\|_0$	time [sec]
0.00	10^0	0	0.0	30.94	10^{-15}	160,139	82.0
2.12	10^{-1}	880,056	5.6	32.75	10^{-16}	160,021	86.6
4.64	10^{-2}	990,166	12.3	34.17	10^{-17}	160,003	90.1
5.63	10^{-3}	996,121	15.1	35.26	10^{-18}	160,000	93.0
7.93	10^{-4}	998,981	20.7	36.55	10^{-19}	160,000	96.6
10.39	10^{-5}	997,394	27.4	38.52	10^{-20}	160,000	101.4
12.11	10^{-6}	993,569	32.3	39.99	10^{-21}	160,000	105.3
14.46	10^{-7}	977,260	38.3	40.98	10^{-22}	160,000	108.1
18.07	10^{-8}	847,156	48.1	43.14	10^{-23}	160,000	113.7
19.52	10^{-9}	701,449	51.7	47.28	10^{-24}	160,000	124.8
21.47	10^{-10}	413,163	56.4	47.28	10^{-25}	160,000	124.8
23.92	10^{-11}	210,624	63.1	47.96	10^{-26}	160,000	126.4
25.18	10^{-12}	179,355	66.6	49.58	10^{-27}	160,000	130.3
27.38	10^{-13}	163,048	72.4	52.31	10^{-28}	160,000	136.8
29.96	10^{-14}	160,311	79.3	53.43	10^{-29}	160,000	139.4

Table 6 Performance of UCDC on a sparse regression instance with a million variables.

$A \in \mathbb{R}^{10^{10} \times 10^9}$, $\|A\|_0 = 2 \times 10^{10}$			
k/n	$\frac{F(x_k)-F^*}{F(x_0)-F^*}$	$\|x_k\|_0$	time [hours]
0	10^0	0	0.00
1	10^{-1}	14,923,993	1.43
3	10^{-2}	22,688,665	4.25
16	10^{-3}	24,090,068	22.65

Table 7 Performance of UCDC on a sparse regression instance with a billion variables and 20 billion nonzeros in matrix A.

convergence speed and ability to find solutions of high accuracy can in part be explained by the fact that for random instances with $m > n$, f will typically be strongly convex, in which case UCDC converges linearly (Theorem 8). It should also be noted that decrease factors this high ($10^{18} - 10^{29}$) would rarely be needed in practice. However, it is nevertheless interesting to see that a simple algorithm can achieve such high levels of accuracy on certain problem instances in huge dimensions.

UCDC has a very similar behavior on the larger problem as well (Table 7). Note that A has 20 billion nonzeros. In $1 \times n$ iterations the initial residual is decreased by a factor of 10, and this takes less than an hour and a half. After less than a day, the residual is decreased by a factor of 1000. Note that it is very unusual for convex optimization methods equipped with iteration complexity guarantees to be able to solve problems of these sizes.

Performance on fat matrices ($m < n$)

When $m < n$, then f is not strongly convex and UCDC has the complexity $O(\frac{n}{\varepsilon} \log \frac{1}{\rho})$ (Theorem 5). In Table 8 we illustrate the behavior of the method on such an instance; we have chosen $m = 10^4$, $n = 10^5$, $\|A\|_0 = 10^7$ and $\|x^*\|_0 = 1,600$. Note that after the first $5,010 \times n$ iterations UCDC decreases the residual by a factor of 10+ only; this takes less than 19 minutes. However, the decrease from 10^2 to 10^{-3} is done in $15 \times n$ iterations and takes 3 seconds only, suggesting very fast local convergence.

k/n	$F(x_k) - F^*$	$\|x_k\|_0$	time [s]
1	$> 10^7$	63,106	0.21
5,010	$< 10^6$	33,182	1,092.59
18,286	$< 10^5$	17,073	3,811.67
21,092	$< 10^4$	15,077	4,341.52
21,416	$< 10^3$	11,469	4,402.77
21,454	$< 10^2$	5,316	4,410.09
21,459	$< 10^1$	1,856	4,411.04
21,462	$< 10^0$	1,609	4,411.63
21,465	$< 10^{-1}$	1,600	4,412.21
21,468	$< 10^{-2}$	1,600	4,412.79
21,471	$< 10^{-3}$	1,600	4,413.38

Table 8 UCDC needs many more iterations when $m < n$, but local convergence is still fast.

Comparing different probability vectors

Nesterov [13] considers only probabilities proportional to a power of the Lipschitz constants:

$$p_i = \frac{L_i^\alpha}{\sum_{i=1}^n L_i^\alpha}, \qquad 0 \le \alpha \le 1. \tag{57}$$

In Figure 1 we compare the behavior of RCDC, with the probability vector chosen according to the power law (57), for three different values of α (0, 0.5 and 1). All variants of RCDC were compared on a single instance with $m = 1,000$, $n = 2,000$ and $\|x^*\|_0 = 300$ (different instances produced by the generator yield similar results) and with $\lambda \in \{0, 1\}$. The plot on the left corresponds to $\lambda = 0$, the plot on the right to $\lambda = 1$.
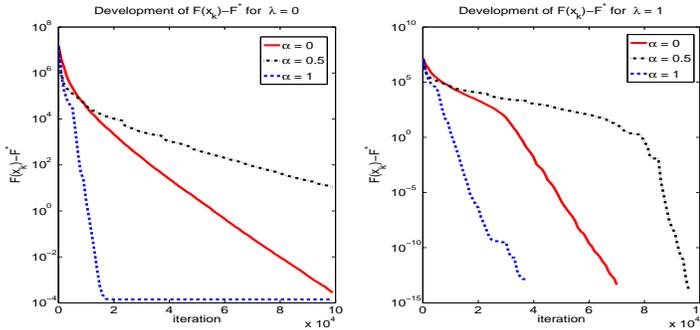

Fig. 1 Development of $F(x_k) - F^*$ for sparse regression problem with $\lambda = 0$ (left) and $\lambda = 1$ (right).

Note that in both cases the choice $\alpha = 1$ is the best. In other words, coordinates with large L_i have a tendency to decrease the objective function the most. However, looking at the $\lambda = 0$ case, we see that the method with $\alpha = 1$ stalls after about 20,000 iterations. The reason for this is that now the coordinates with small L_i should be chosen to further decrease the objective value. However, they are chosen with very small probability and hence the slowdown. A solution to this could be to start the method with $\alpha = 1$ and then switch to $\alpha = 0$ later on. On the problem with $\lambda = 1$ this effect is less pronounced. This is to be expected as now the objective function is a combination of f and Ψ, with Ψ exerting its influence and mitigating the effect of the Lipschitz constants.

Coordinate descent vs. a full-gradient method

In Figure 2 we compare the performance of RCDC with the full gradient (FG) algorithm [12] (with the Lipschitz constant $L_{FG} = \lambda_{\max}(A^T A)$ for four different distributions of the Lipschitz constants L_i. Note that $\max_i L_i \leq L_{FG} \leq \sum_i L_i$. Since the work performed during one iteration of FG is comparable with the work performed by UCDC during n coordinate iterations[3], for FG we multiply the iteration count by n. In all four tests we solve instances with $A \in \mathbb{R}^{2,000 \times 1,000}$.
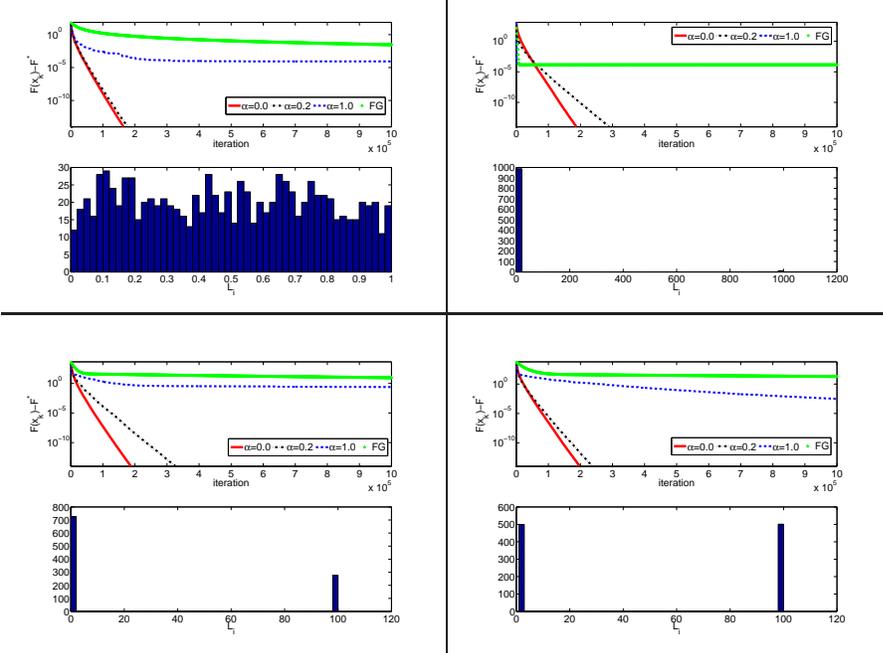

Fig. 2 Comparison of UCDC with different choices of α with a full-gradient method (essentially UCDC with one block: $n = 1$) for four different distributions of the Lipschitz constants L_i.

In the 1-1 plot of Figure 2 (plot in the 1-1 position, i.e., in the upper-left corner), the Lipschitz constants L_i were generated uniformly at random in the interval $(0, 1)$. We see that the RCDC variants with $\alpha = 0$ and $\alpha = 0.2$ exhibit virtually the same behavior, whereas $\alpha = 1$ and FG struggle finding a solution with error tolerance below 10^{-5} and 10^{-2}, respectively. The $\alpha = 1$ method does start off a bit faster, but then stalls due to the fact that the coordinates with small Lipschitz constants are chosen with extremely small probabilities. For a more accurate solution one needs to be updating these coordinates as well.

In order to zoom in on this phenomenon, in the 1-2 plot we construct an instance with an extreme distribution of Lipschitz constants: 98% of the constants have the value 10^{-6}, whereas the remaining 2% have the value 10^3. Note that while the FG and $\alpha = 1$ methods are able to quickly decrease the objective function within 10^{-4} of the optimum, they get stuck afterwards since they effectively never update the coordinates with $L_i = 10^{-6}$. On the

---

[3] This will not be the case for certain types of matrices, such as those arising from wavelet bases or FFT.

other hand, the $\alpha = 0$ method starts off slowly, but does not stop and manages to solve the problem eventually, in about 2×10^5 iterations.

In the 2-1 (resp. 2-2) plot we choose 70% (resp. 50%) of the Lipschitz constants L_i to be 1, and the remaining 30% (resp. 50%) equal to 100. Again, the $\alpha = 0$ and $\alpha = 0.2$ methods give the best long-term performance.

In summary, if fast convergence to a solution with a moderate accuracy us needed, then $\alpha = 1$ is the best choice (and is always better than FG). If one desires a solution of higher accuracy, it is recommended to switch to $\alpha = 0$. In fact, it turns out that we can do much better than this using a "shrinking" heuristic.

Speedup by adaptive change of probability vectors

It is well-known that increasing values of λ encourage increased sparsity in the solution of (56). In the experimental setup of this section we observe that from certain iteration onwards, the sparsity pattern of the iterates of RCDC is a very good predictor of the sparsity pattern of the optimal solution x^* the iterates converge to. More specifically, we often observe in numerical experiments that for large enough k the following holds:

$$(x_k^{(i)} = 0) \quad \Rightarrow \quad (\forall l \geq k \quad x_l^{(i)} = (x^*)^{(i)} = 0).\tag{58}$$

In words, for large enough k, zeros in x_k typically stay zeros in all subsequent iterates[4] and correspond to zeros in x^*. Note that RCDC is *not* able to take advantage of this. Indeed, RCDC, as presented in the theoretical sections of this paper, uses the fixed probability vector p to randomly pick a single coordinate i to be updated in each iteration. Hence, eventually, $\sum_{i:x_k^{(i)}=0} p_i$ proportion of time will be spent on vacuous updates.

Looking at the data in Table 6 one can see that after approximately $35 \times n$ iterations, x_k has the same number of non-zeros as x^* (160,000). What is not visible in the table is that, in fact, the relation (58) holds for this instance much sooner. In Figure 3 we illustrate this phenomenon in more detail on an instance with $m = 500$, $n = 1,000$ and $\|x^*\|_0 = 100$.
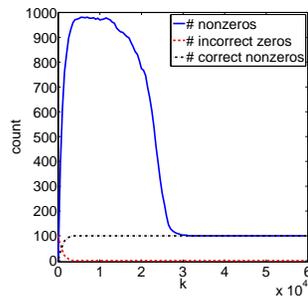

Fig. 3 Development of non-zero elements in x_k through iterations.

First, note that the *number of nonzeros* (solid blue line) in the current iterate, $\#\{i : x_k^{(i)} \neq 0\}$, is first growing from zero (since we start with $x_0 = 0$) to just below n in about 0.6×10^4

---

[4] There are various theoretical results on the *identification of active manifolds* explaining numerical observations of this type; see [7] and the references therein. See also [28].

iterations. This value than starts to decrease starting from about $k \approx 15n$ and reaches the optimal number of nonzeros at iteration $k \approx 30n$ and stays there afterwards. Note that the number of *correct nonzeros*,

$$cn_k = \#\{i : x_k^{(i)} \neq 0 \ \& \ (x^*)^{(i)} \neq 0\},$$

is increasing (for this particular instance) and reaches the optimal level $\|x^*\|_0$ very quickly (at around $k \approx 3n$). An alternative, and perhaps a more natural, way to look at the same thing is via the *number of incorrect zeros*,

$$iz_k = \#\{i : x_k^{(i)} = 0 \ \& \ (x^*)^{(i)} \neq 0\}.$$

Indeed, we have $cn_k + iz_k = \|x^*\|_0$. Note that for our problem $iz_k \approx 0$ for $k \geq k_0 \approx 3n$.

The above discussion suggests that an *iterate-dependent* policy for updating of the probability vectors p_k in Algorithm 4 might help to accelerate the method. Let us now introduce a simple *q-shrinking* strategy for adaptively changing the probabilities as follows: at iteration $k \geq k_0$, where k_0 is large enough, set

$$p_k^{(i)} = \hat{p}_k^{(i)}(q) \overset{\text{def}}{=} \begin{cases} \frac{1-q}{n}, & \text{if } x_k^{(i)} = 0, \\ \frac{1-q}{n} + \frac{q}{\|x_k\|_0}, & \text{otherwise.} \end{cases}$$

This is equivalent to choosing i_k uniformly from the set $\{1, 2, \ldots, n\}$ with probability $1 - q$ and uniformly from the support set of x_k with probability q. Clearly, different variants of this can be implemented, such as fixing a new probability vector for $k \geq k_0$ (as opposed to changing it for every k) ; and some may be more effective and/or efficient than others in a particular context. In Figure 4 we illustrate the effectiveness of q-shrinking on an instance of size $m = 500$, $n = 1,000$ with $\|x^*\|_0 = 50$. We apply to this problem a modified version of RCDC started from the origin ($x_0 = 0$) in which uniform probabilities are used in iterations $0, \ldots, k_0 - 1$, and q-shrinking is introduced as of iteration k_0:

$$p_k^{(i)} = \begin{cases} \frac{1}{n}, & \text{for } k = 0, 1, \ldots, k_0 - 1, \\ \hat{p}_k^{(i)}(q), & \text{for } k \geq k_0. \end{cases}$$
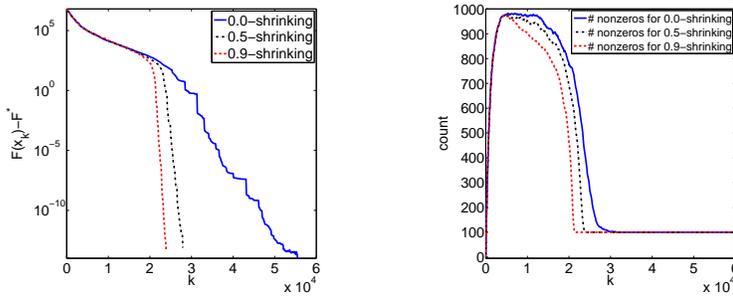
We have used $k_0 = 5 \times n$.

Fig. 4 Comparison of different shrinking strategies.

Notice that as the number of nonzero elements of x_k decreases, the time savings from q-shrinking grow. Indeed, 0.9-shrinking introduces a saving of nearly 70% when compared

to 0-shrinking to obtain x_k satisfying $F(x_k) - F^* \leq 10^{-14}$. We have repeated this experiment with two modifications: a) a random point was used as the initial iterate (scaled so that $\|x_0\|_0 = n$) and b) $k_0 = 0$. The corresponding plots are very similar to Figure 4 with the exception that the lines in the second plot start from $\|x_0\|_0 = n$.

6.2 Linear Support Vector Machines

Consider the problem of training a linear classifier with training examples $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, where x_i are the feature vectors and $y_i \in \{-1, +1\}$ the corresponding labels (classes). This problem is usually cast as an optimization problem of the form (1),

$$\min_{w \in \mathbb{R}^n} F(w) = f(w) + \Psi(w), \tag{59}$$

where

$$f(w) = \gamma \sum_{i=1}^{m} \mathscr{L}(w; x_i, y_i),$$

$\mathscr{L}$ is a nonnegative convex loss function and $\Psi(\cdot) = \|\cdot\|_1$ for L1-regularized and $\Psi(\cdot) = \|\cdot\|_2$ for L2-regularized linear classifier. Some popular loss functions are listed in Table 9. For more details we refer the reader to [28] and the references therein; for a survey of recent advances in large-scale linear classification see [29].

$\mathscr{L}(w; x_i, y_i)$	name	property
$\max\{0, 1 - y_j w^T x_j\}$	L1-SVM loss (L1-SVM)	C^0 continuous
$\max\{0, 1 - y_j w^T x_j\}^2$	L2-SVM loss (L2-SVM)	C^1 continuous
$\log(1 + e^{-y_j w^T x_j})$	logistic loss (LG)	C^2 continuous

Table 9 A list of a few popular loss functions.

Because our setup requires f to be at least C^1 continuous, we will consider the L2-SVM and LG loss functions only. In the experiments below we consider the L1 regularized setup.

A few implementation remarks

The Lipschitz constants and coordinate derivatives of f for the L2-SVM and LG loss functions are listed in Table 10.

Loss function	L_i	$\nabla_i f(w)$
L2-SVM	$2\gamma \sum_{j=1}^{m} (y_j x_j^{(i)})^2$	$-2\gamma \cdot \sum_{j : -y_j w^T x_j > -1} y_j x_j^{(i)} (1 - y_j w^T x_j)$
LG	$\frac{\gamma}{4} \sum_{j=1}^{m} (y_j x_j^{(i)})^2$	$-\gamma \cdot \sum_{j=1}^{m} y_j x_j^{(i)} \dfrac{e^{-y_j w^T x_j}}{1 + e^{-y_j w^T x_j}}$

Table 10 Lipschitz constants and coordinate derivatives for SVM.

For an efficient implementation of UCDC we need to be able to cheaply update the partial derivatives after each step of the method. If at step k coordinate i gets updated, via $w_{k+1} = w_k + te_i$, and we let $r_k^{(j)} \overset{\text{def}}{=} -y_j w^T x_j$ for $j = 1, \dots, m$, then

$$r_{k+1}^{(j)} = r_k^{(j)} - ty_j x_j^{(i)}, \quad j = 1, \dots, m. \tag{60}$$

Let o_i be the number of observations feature i appears in, i.e., $o_i = \#\{j : x_j^{(i)} \neq 0\}$. Then the update (60), and consequently the update of the partial derivative (see Table 10), requires $O(o_i)$ operations. In particular, in *feature-sparse problems* where $\frac{1}{n} \sum_{i=1}^{n} o_i \ll m$, an average iteration of UCDC will be very cheap.

Small scale test

We perform only preliminary results on the dataset `rcv1.binary`[5]. This dataset has 47,236 features and 20,242 training and 677,399 testing instances. We train the classifier on 90% of training instances (18,217); the rest we used for cross-validation for the selection of the parameter γ. In Table 11 we list cross-validation accuracy (CV-A) for various choices of γ and testing accuracy (TA) on 677,399 instances. The best constant γ is 1 for both loss functions in cross-validation.

Loss function	γ	CV-A	TA	γ	CV-A	TA
L2-SVM	0.0625	94.1%	93.2%	2	97.0%	95.6%
	0.1250	95.5%	94.5%	4	97.0%	95.4%
	0.2500	96.5%	95.4%	8	96.9%	95.1%
	0.5000	97.0%	95.8%	16	96.7%	95.0%
	1.0000	**97.0%**	**95.8%**	32	96.4%	94.9%
LG	0.5000	0.0%	0.0%	8	40.7%	37.0%
	1.0000	**96.4%**	**95.2%**	16	37.7%	36.0%
	2.0000	43.2%	39.4%	32	37.6%	33.4%
	4.0000	39.3%	36.5%	64	36.9%	34.1%

Table 11 Cross validation accuracy (CV-A) and testing accuracy (TA) for various choices of γ.

In Figure 5 we present dependence of TA on the number of iterations we run UCDC for (we measure this number in multiples of n). As you can observe, UCDC finds good solution after $10 \times n$ iterations, which for this data means less then half a second. Let us remark that we did not include bias term or any scaling of the data.

Large scale test

We have used the dataset `kdd2010` (bridge to algebra)[6], which has 29,890,095 features and 19,264,097 training and 748,401 testing instances. Training the classifier on the entire training set required approximately 70 seconds in the case of L2-SVM loss and 112 seconds in the case of LG loss. We have run UCDC for n iterations.
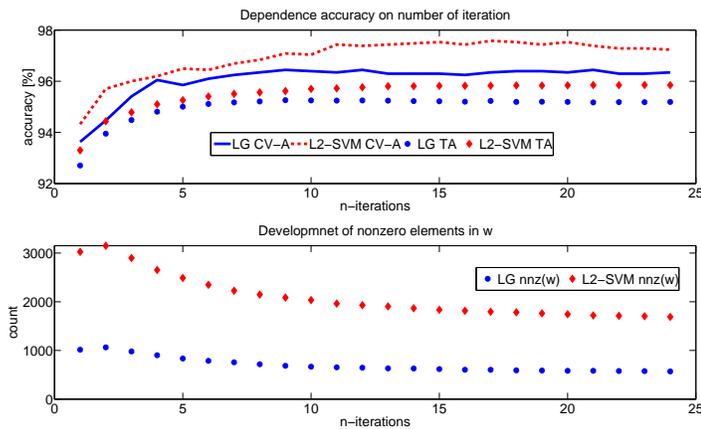
---

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

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

Fig. 5 Dependence of tested accuracy (TA) on the number of full passes through the coordinates.

Acknowledgements

References

1. Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, September 1999.
2. Adrian A. Canutescu and Roland L. Dunbrack. Cyclic coordinate descent: A robotics algorithm for protein loop closure. *Protein Science*, 12:963–972, 2003.
3. Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. Coordinate descent method for large-scale l_2-loss linear support vector machines. *Journal of Machine Learning Research*, 9:1369–1398, 2008.
4. Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A note on the group lasso and a sparse group lasso. Technical report, 2010.
5. Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S Sathiya Keerthi, and S Sundararajan. A dual coordinate descent method for large-scale linear svm. In *ICML 2008*, pages 408–415, 2008.
6. Dennis Leventhal and Adrian S. Lewis. Randomized methods for linear constraints: Convergence rates and conditioning. *Mathematics of Operations Research*, 35(3):641–654, 2010.
7. Adrian S. Lewis and Stephen J. Wright. A proximal method for composite minimization. Technical report, 2008.
8. Yingying Li and Stanley Osher. Coordinate descent optimization for l_1 minimization with application to compressed sensing; a greedy algorithm. *Inverse Problems and Imaging*, 3:487–503, August 2009.
9. Z. Q. Luo and Paul Tseng. A coordinate gradient descent method for nonsmooth separable minimization. *Journal of Optimization Theory and Applications*, 72(1), January 2002.
10. Lukas Meier, Sara Van De Geer, and Peter Buhlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society B*, 70:53–71, 2008.
11. Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer Netherlands, 1st edition, 2004.
12. Yurii Nesterov. Gradient methods for minimizing composite objective function. Core discussion paper #2007/76, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), September 2007.
13. Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J Optimization*, (22):341–362, 2012.
14. Zhiwei (Tony) Qin, Katya Scheinberg, and Donald Goldfarb. Efficient block-coordinate descent algorithms for the group lasso. Technical report, 2010.

15. Peter Richtárik and Martin Takáč. 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*, 2011.

16. Peter Richtárik and Martin Takáč. Efficient serial and parallel coordinate descent method for huge-scale truss topology design. In *Operations Research Proceedings 2011*, pages 27–32. Springer, 2012.

17. Ankan Saha and Ambuj Tewari. On the finite time convergence of cyclic coordinate descent methods. *CoRR*, abs/1005.2146, 2010.

18. Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l_1 regularized loss minimization. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

19. Thomas Strohmer and Roman Vershynin. A randomized kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15:262–278, 2009.

20. Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, 58:268–288, 1996.

21. Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109:475–494, June 2001.

22. Paul Tseng and Sangwoon Yun. Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. *Journal of Optimization Theory and Applications*, 140:513–535, 2009. 10.1007/s10957-008-9458-3.

23. Paul Tseng and Sangwoon Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programmming, Ser. B*, 117:387–423, 2009.

24. Zaiwen Wen, Donald Goldfarb, and Katya Scheinberg. Block coordinate descent methods for semidefinite programming. In Miguel F. Anjos and Jean B. Lasserre, editors, *Handbook on Semidefinite, Cone and Polynomial Optimization: Theory, Algorithms, Software and Applications*. Springer, forthcoming.

25. Stephen J. Wright. Accelerated block-coordinate relaxation for regularized optimization. Technical report, University of Wisconsin, 2010.

26. Stephen J. Wright, Robert D. Nowak, and Mário A. T. Figueiredo. Sparse reconstruction by separable approximation. *Trans. Sig. Proc.*, 57:2479–2493, July 2009.

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

28. Guo-Xun Yuan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. A comparison of optimization methods and software for large-scale l_1-regularized linear classification. *Journal of Machine Learning Research*, 11(1):3183–3234, 2010.

29. Guo-Xun Yuan, Ho Chia-Hua, and Chih-Jen Lin. Recent advances of large-scale linear classification. Technical report, 2012.

30. Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society B*, 68:49–67, 2006.

31. Sangwoon Yun and Kim-Chuan Toh. A coordinate gradient descent method for l_1-regularized convex minimization. *Computational Optimization and Applications*, 48:273–307, 2011.

32. Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, 67:301–320, 2005.