ROBUST BLOCK COORDINATE DESCENT

May 8, 2015

KIMON FOUNTOULAKIS[*] AND RACHAEL TAPPENDEN[†]

Abstract. In this paper we present a novel *randomized* block coordinate descent method for the minimization of a convex composite objective function. The method uses (approximate) partial second-order (curvature) information, so that the algorithm performance is more robust when applied to highly nonseparable or ill conditioned problems. We call the method Robust Coordinate Descent (RCD). At each iteration of RCD, a block of coordinates is sampled randomly, a quadratic model is formed about that block and the model is minimized *approximately/inexactly* to determine the search direction. An inexpensive line search is then employed to ensure a monotonic decrease in the objective function and acceptance of large step sizes. We prove global convergence of the RCD algorithm, and we also present several results on the local convergence of RCD for strongly convex functions. Finally, we present numerical results on large-scale problems to demonstrate the practical performance of the method.

Key words. large scale optimization, second-order methods, curvature information, block coordinate descent, nonsmooth problems

1. Introduction.

In this work we are interested in solving the following convex composite optimization problem

$$\min_{x\in\mathbb{R}^N} F(x) := f(x) + \Psi(x), \tag{1.1}$$

where $f(x)$ is a smooth convex function and $\Psi(x)$ is a (possibly) nonsmooth, block separable, extended real valued convex function (this will be defined precisely in Section 2.5). Problems of the form of (1.1) arise in many important scientific fields, and applications include machine learning [29], regression [27] and compressed sensing [4, 3, 5]. Often the term $f(x)$ is a data fidelity term, and the term $\Psi(x)$ represents some kind of regularization.

Frequently, problems of the form of (1.1) are large-scale problems, i.e., the size of N is of the order of a million or a billion. Large-scale problems impose restrictions on the types of methods that can be employed for the solution of (1.1). In particular, the methods should have low per iteration computational cost, otherwise completing even a single iteration of the method might require unreasonable time. The methods must also rely only on simple operations such as inner products or matrix vector products, and ideally, they should offer fast progress towards optimality.

First order methods, and in particular randomized coordinate descent methods, have found great success in this area because they can take advantage of the underlying problem structure (separability and block structure), and satisfy the requirements of low computational cost and low storage requirements. For example, in [20] the authors show that their randomized coordinate descent method was able to solve sparse problems with millions of variables in a reasonable amount of time.

[*] K. Fountoulakis is with the School of Mathematics and Maxwell Institute, The University of Edinburgh, Edinburgh, Mayfield Road, Edinburgh EH9 3JZ, United Kingdom e-mail: K.Fountoulakis@sms.ed.ac.uk.

[†] R. Tappenden is with the School of Mathematics and Maxwell Institute, The University of Edinburgh, Edinburgh, Mayfield Road, Edinburgh EH9 3JZ, United Kingdom e-mail: r.tappenden@ed.ac.uk.

Unfortunately, randomized coordinate descent methods have two significant drawbacks. First, due to its coordinate nature, it is efficient only on problems with high degree of separability, and performance suffers when there is a high dependency between variables. Second, as a first-order method, coordinate descent methods do not usually capture essential curvature information of the problem and have been shown to struggle on complicated sparse problems [8].

The purpose of this work is to overcome these drawbacks by equipping a randomized block coordinate descent method with approximate partial second-order information. In particular, at every iteration of RCD the direction is obtained by solving *approximately* a block piece-wise quadratic model, where the model includes a matrix representing approximate second order information. Then, a line search is employed in order to guarantee a monotonic decrease of the objective function.

RCD randomly selects a block of coordinates at every iteration, which is inexpensive. Although the per iteration computational cost of the method may be higher than other randomized coordinate descent methods, we show that in practise the method is more robust and the total number of iterations decreases. In particular we show that RCD is able to solve difficult problems, on which other coordinate descent method may struggle. RCD uses an inexact search direction, (the termination condition for the block piecewise quadratic subproblem is inspired by [2]), coupled with a line search to ensure a monotonic decrease in the function values, and we prove global convergence of RCD and study its local convergence properties.

1.1. Literature review. Coordinate descent methods are some of the oldest iterative methods, and they are often better known in the literature under various names such as Jacobi methods, Gauss-Seidel methods, among others. It has been observed that these methods suffer from poor practical performance, particularly on ill-conditioned problems. However, as we enter the era of big data, coordinate descent methods are coming back into favour, because of their ability to provide approximate solutions of some realistic very large/huge scale problems in a reasonable amount of time.

Currently, randomized coordinate descent methods include that of Richtàrik and Takàč [20], where the method can be applied to unconstrained convex composite optimization problems of the form (1.1). The algorithm is supported by theoretical convergence guarantees in the form of high probability iteration complexity results, and [20] also reports very impressive practical performance on highly separable large scale problems. The work has also been extended to the parallel case [21], to include acceleration techniques [7], and to include the use of inexact updates [26].

Other important works on randomized coordinate descent methods include methods for huge-scale problems [16], work in [13] that improves the complexity analysis of [21], coordinate descent methods for group lasso [19, 25] and general regularizers [24, 28] and coordinate descent for constrained optimization problems [14].

Unfortunately, on ill-conditioned problems, or problems that are highly nonseparable, first order methods can display very poor practical performance, and this has prompted the study of methods that employ second order information. To this end, recently there has been a flurry of research on Newton-type methods for problems of the general form (1.1), or a special case where $\Psi(x) = \|x\|_1$. For example, Karimi and Vavasis [10] have developed a proximal quasi-Newton method for l_1-regularized least squares problems, Lee, Sun and Saunders [11, 12] have proposed a family of Newton-type methods for solving problems of the form (1.1) and Scheinberg and Tang [23] present iteration complexity results for a proximal Newton-type method. Moreover, the authors in [2] extended standard inexact Newton-type methods to the case of minimization of a composite objective involving a smooth convex term plus an l_1-regularizer term. Finally, there exists parallel deterministic [6] and sequential active set [22] block coordinate descent methods, where the authors incorporate some block second-order information in the algorithmic process.

We believe that the works in [2, 6] can lead the way to allow general *randomized* coordinate descent to practically incorporate second-order information and in this paper we propose a method which combines the ideas in [2, 6, 20].

1.2. Core ideas and Major Contributions. In this section we list several of the core ideas and major contributions of this work on randomized block coordinate descent methods. The first two points briefly describe the idea of incorporating (approximate) partial second-order (curvature) information (which have been also presented in a similar way in [6]), whilst the last three are contributions of this paper.

1. **Incorporation of some second order information.** RCD uses a quadratic model to determine the search direction, which incorporates a user defined positive definite matrix $H^{(i)}(x_k)$. If $H^{(i)}(x_k)$ approximates the Hessian, then second order information is incorporated into the quadratic model, and the search direction obtained by minimizing the model is an approximate Newton-type direction. We stress that $H^{(i)}(x)$ can change at every iteration and this is an advantage over the method in [20] where the matrix is fixed at the start of the algorithm for each block of coordinates.
2. **Inexact updates.** To ensure that this method is computationally practical, it is imperative that the iterates are inexpensive, and RCD achieves this through the use of *inexact* updates. Any algorithm can be used to approximately minimize the quadratic model. Moreover, the stopping conditions for inner solve are *easy to verify*; they depend upon quantities that are easy/inexpensive to obtain, or may be available as a byproduct of the inner search direction solver.
3. **Blocks can vary throughout iterations.** If $\Psi(x)$ is completely separable into coordinates then we do not restrict ourselves to a fixed block structure; rather we allow the blocks of coordinates to *change at any iteration*. This is important because every element of the Hessian can be accessed (this is discussed further in Section 3.2.2).
4. **Line search.** The algorithm includes a line search step to ensure a monotonic decrease of the objective function as iterates progress. The line search is inexpensive to perform because, at each iteration, *it depends on a single block of coordinates only.* One of the major advantages of incorporating second-order information combined with line search is to allow in practice the selection of *large step sizes* (close to one). This is because unit step sizes can substantially improve the practical efficiency of a method. We prove that if f is strongly convex, then close to the optimal solution unit step sizes are selected. In fact, for all experiments that we performed, unit step sizes were accepted by line search for the majority of the iterations.
5. **Convergence theory.** We provide global convergence results to show that the RCD algorithm is guaranteed to converge in the limit. We also provide local convergence theory for strongly convex functions f. In particular, depending on the choice of stopping condition for the inner search direction solve and the matrix $H^{(i)}(x_k)$, we show that close to the optimal solution RCD has on expectation *block* quadratic or superlinear rate of convergence.

1.3. Format of the paper. The paper is organised as follows. In Section 2 we introduce the notation and definitions that are used throughout this paper, as well as giving several technical results. We also define the quadratic model that is used in the algorithm, prove the equivalence of some stationarity conditions for problem (1.1), and define a continuous measure of the distance of the current point from the set of solutions of (1.1). A thorough description of the RCD algorithm is presented in Section 3, including how the blocks are selected/sampled at each iteration, a description of the search direction and line search, several suggestions for the matrices $H^{(i)}(x_k)$, and we also present several concrete examples.

The second half of the paper is devoted to providing convergence results and numerical experiments. In Sections 4, global convergence results are presented, which do not require f to be convex. Local convergence theory for RCD is presented in Section 5. There we show that, close to optimality line search accepts unit step sizes. Moreover, if both the stopping conditions for the inner search direction solve and the matrix $H^{(i)}(x_k)$ are chosen appropriately, then RCD has on expectation block quadratic or superlinear rate of convergence. Finally, several numerical experiments are presented in Section 6, which show that the algorithm performs very well in practice.

2. Preliminaries.

In this section we introduce the notation and definitions that are used in this paper, and we also present some important technical results. Throughout the paper $\|\cdot\| \equiv \sqrt{\langle \cdot, \cdot \rangle}$ and $\|\cdot\|_A \equiv \sqrt{\langle \cdot, A\cdot \rangle}$, where A is a positive definite matrix. Moreover, $\lambda_{\min}(\cdot)$ and $\lambda_{\max}(\cdot)$ denote the smallest and largest eigenvalue of $\cdot$, respectively.

2.1. Subgradient and subdifferential. For a function $\Phi : \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$ the elements $s \in \mathbb{R}^N$ that satisfy

$$\Phi(y) \geq \Phi(x) + \langle s, y - x \rangle,$$

are called the subgradients of Φ at point x. In words, all elements defining a linear function that supports the function Φ at point x are subgradients. The set of all s at a point x is called the subdifferential of Φ and it is denoted by $\partial \Phi(x)$.

2.2. Convexity. A function $\Phi : \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$ is strongly convex with convexity parameter $\mu_\Phi > 0$ if for all $x, y \in \mathbb{R}^N$, and where $s \in \partial\Phi(x)$,

$$\Phi(y) \geq \Phi(x) + \langle s, y - x \rangle + \tfrac{\mu_\Phi}{2}\|y - x\|^2.$$

If $\mu_\Phi = 0$. then function Φ is said to be convex.

2.3. Convex conjugate and proximal mapping. For a convex function $\Phi : \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$, its convex conjugate is defined as $\Phi^*(y) \equiv := \sup_{u \in \mathbb{R}^N} \langle u, y \rangle - \Phi(u)$. The proximal mapping of a convex function Ψ at x is

$$\mathrm{prox}_\Psi(x) := \arg\min_{y \in \mathbb{R}^N} \Psi(y) + \frac{1}{2}\|y - x\|^2, \tag{2.1}$$

and the proximal mapping of its convex conjugate Ψ^* is

$$\mathrm{prox}_{\Psi^*}(x) := \arg\min_{y \in \mathbb{R}^N} \Psi^*(y) + \frac{1}{2}\|y - x\|^2. \tag{2.2}$$

The following relation holds between the two proximal mappings.

Lemma 2.1 (Chapter 1, (1.4) in [1]). *Let Ψ be a convex function and let Ψ^* denote its convex conjugate. Then, $x = \mathrm{prox}_\Psi(x) + \mathrm{prox}_{\Psi^*}(x)$ for all x.*

From Chapter 1 of [1], we also see that $\mathrm{prox}_\Psi(\cdot)$ and $\mathrm{prox}_{\Psi^*}(\cdot)$ are nonexpansive

$$\|\mathrm{prox}_\Psi(y) - \mathrm{prox}_\Psi(x)\| \leq \|y-x\|, \quad \text{and} \quad \|\mathrm{prox}_{\Psi^*}(y) - \mathrm{prox}_{\Psi^*}(x)\| \leq \|y-x\|. \tag{2.3}$$

Finally, from Chapter 1 of [1] we have

$$\mathrm{prox}_{\Psi^*}(x) \in \partial\Psi(\mathrm{prox}_\Psi(x)). \tag{2.4}$$

2.4. Block decomposition of $\mathbb{R}^N$. 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, where U_i is $N \times N_i$ and $\sum_{i=1}^n N_i = N$. It is clear that any vector $x \in \mathbb{R}^N$ can be written uniquely as $x = \sum_{i=1}^n U_i x^{(i)}$, where $x^{(i)} \in \mathbb{R}^{N_i}$ and block i denotes a subset of $\{1, 2, \ldots, N\}$. Moreover, these vectors are given by

$$x^{(i)} := U_i^T x. \tag{2.5}$$

2.5. Block decomposition of Ψ. The function $\Psi : \mathbb{R}^N \to \mathbb{R}\cup\{+\infty\}$ is assumed to be block separable. That is, we assume that $\Psi(x)$ can be decomposed as:

$$\Psi(x) = \sum_{i=1}^n \Psi_i(x^{(i)}), \tag{2.6}$$

where the functions $\Psi_i : \mathbb{R}^{N_i} \to \mathbb{R} \cup \{+\infty\}$ are convex.

Notice that if $n = N$, $\Psi(x)$ is said to be separable (into coordinates), whereas if $n < N$, then $\Psi(x)$ is said to be *block* separable (separable into blocks of coordinates).

The following relationship will be used repeatedly in this work:

$$\begin{aligned}\Psi(x + U_i t^{(i)}) - \Psi(x) &= \Big(\sum_{j\neq i} \Psi_j(x^{(j)}) + \Psi_i(x^{(i)} + t^{(i)})\Big) - \Big(\sum_{j\neq i} \Psi_j(x^{(j)}) + \Psi_i(x^{(i)})\Big) \\ &= \Psi_i(x^{(i)} + t^{(i)}) - \Psi_i(x^{(i)}). \end{aligned} \tag{2.7}$$

2.6. Block Lipschitz continuity of f. Throughout the paper we assume that the gradient of f is block Lipschitz, uniformly in x. This means that, for all $x \in \mathbb{R}^N$, $i \subseteq \{1, 2, \ldots, n\}$ and $t^{(i)} \in \mathbb{R}^{N_i}$ we have

$$\|\nabla_i f(x + U_i t^{(i)}) - \nabla_i f(x)\| \leq L_i \|t^{(i)}\|, \tag{2.8}$$

where $\nabla_i f(x) \overset{(2.5)}{=} U_i^T \nabla f(x)$. An important consequence of (2.8) is the following standard inequality [15, p.57]:

$$f(x + U_i t^{(i)}) \leq f(x) + \langle \nabla_i f(x), t^{(i)}\rangle + \tfrac{L_i}{2}\|t^{(i)}\|^2. \tag{2.9}$$

2.7. Piecewise Quadratic Model. For fixed $x \in \mathbb{R}^N$, we define a piecewise quadratic approximation of F around the point $(x + t) \in \mathbb{R}^N$ as follows:

$$F(x+t) \approx Q(x;t) := f(x) + \sum_{i=1}^n Q_i(x, t^{(i)}), \tag{2.10}$$

where

$$Q_i(x, t^{(i)}) := \langle \nabla_i f(x), t^{(i)}\rangle + \frac{1}{2}\|t^{(i)}\|^2_{H^{(i)}(x)} + \Psi_i(x^{(i)} + t^{(i)}), \tag{2.11}$$

and $H^{(i)}(x) \in \mathbb{R}^{N_i\times N_i}$ is *any* positive definite matrix, which possibly depends on x. Notice that $Q(x;0) = F(x)$ and that $Q_i(x, t^{(i)})$ is the quadratic model for block i.

2.8. Stationarity conditions. The following theorem gives the equivalence of some stationarity conditions of problem (1.1).

THEOREM 2.2. *The following are equivalent first order optimality conditions of problem* (1.1).

(i) $\nabla f(x) + s = 0$ *and* $s \in \partial\Psi(x)$,

(ii) $-\nabla f(x) \in \partial\Psi(x)$,

(iii) $\nabla f(x) + \frac{1}{\beta}\text{prox}_{(\beta\Psi)^*}(x - \beta\nabla f(x)) = 0$,

(iv) $x = \text{prox}_{\beta\Psi}(x - \beta\nabla f(x))$,

where β *is any positive constant.*

Proof. It is easy to see that (i) are first-order optimality conditions of problem (1.1), which can be obtained by using the definition of subgradient. It is trivial to show that $(i) \iff (ii)$. By Lemma 2.1, we have that $(iii) \iff (iv)$. We now show that $(iii) \iff (ii)$. We rewrite (ii) as

$$0 \in \beta\nabla f(x) + y - x + \beta\partial\Psi(x) \quad \text{and} \quad y = x,$$

which is satisfied if and only if (iv) holds, hence, if and only if (iii) holds. □

Let us define the continuous function

$$g(x;t) := \nabla f(x) + H(x)t + \frac{1}{\beta}\text{prox}_{(\beta\Psi)^*}\big(x + t - \beta(\nabla f(x) + H(x)t)\big), \tag{2.12}$$

where β is a positive constant, which is used in the local convergence analysis (Section 5). By Theorem 2.2, the points that satisfy $g(x;0) = \nabla f(x) + \frac{1}{\beta}\text{prox}_{(\beta\Psi)^*}(x - \beta\nabla f(x)) = 0$ are stationary points for problem (1.1). Hence, $g(x;0)$ is a continuous measure of the distance from the set of stationary points of problem (1.1).

Furthermore, let us define

$$\begin{aligned} g_i(x;t^{(i)}) := {} & \nabla_i f(x) + H^{(i)}(x)t^{(i)} \\ & + \frac{1}{\beta}\text{prox}_{(\beta\Psi)_i^*}\big(x^{(i)} + t^{(i)} - \beta(\nabla_i f(x) + H^{(i)}(x)t^{(i)})\big), \end{aligned} \tag{2.13}$$

which will be used as a continuous measure for the distance from stationarity of the block piecewise quadratic function $Q_i(x_k;t^{(i)})$.

3. The Algorithm.

In this section we present the Robust Coordinate Descent (RCD) algorithm for solving problems of the form (1.1). There are three key steps in the algorithm: (step 4) the coordinates are sampled randomly; (step 5) the quadratic model (2.11) is solved approximately until the stopping conditions (3.2) are satisfied to give a search direction; (step 6) a line search is performed to find a step size that ensures a sufficient reduction in the objective value. Once these key steps have been performed, the current point x_k is updated to give a new point x_{k+1}, and the process is repeated.

The following assumption is used in RCD. The reason this assumption is used will be made clear in Section 3.1.

ASSUMPTION 3.1. *The block decomposition of* $\mathbb{R}^N$ *used within RCD, and the associated probability distribution, adhere to the block structure of* $\Psi(x)$.

We now present pseudocode for the algorithm, while a thorough description of each of the key steps in the algorithm will follow in the rest of this section.

Algorithm 1 Robust Coordinate Descent (RCD)

```
1: Input Choose $x_0 \in \mathbb{R}^N$, $\theta \in (0, 1/2)$ and $\beta > 0$.
2: Initialize a decomposition of $\mathbb{R}^N$ and a probability distribution following Assumption 3.1
3: for $k = 1, 2, \cdots$ do
4:     Sample a block of coordinates $i$ with probability $p_i > 0$.
5:     If $g_i(x_k; 0) = 0$ then go to Step 3; else approximately solve
           (3.1)    $t_k^{(i)} := \arg\min_{t^{(i)}} Q_i(x_k; t^{(i)})$,
       until the stopping conditions
           (3.2)    $Q(x_k; U_i t_k^{(i)}) < Q(x_k; 0)$ and $\|g_i(x_k; t_k^{(i)})\| \leq \eta_k^i \|g_i(x_k; 0)\|$,
       are satisfied, (where $\eta_k^i \in [0, 1)$).
6:     Perform a backtracking line search along the direction $t_k^{(i)}$ starting from $\alpha = 1$.
       That is, find $\alpha \in (0, 1]$ such that
           (3.3)    $F(x_k) - F(x_k + \alpha U_i t_k^{(i)}) \geq \theta \left( \ell(x_k; 0) - \ell(x_k; \alpha U_i t_k^{(i)}) \right)$,
       where
           (3.4)    $\ell(x_k; t) := f(x_k) + \langle \nabla f(x_k), t \rangle + \Psi(x_k + t)$.
7:     Update $x_{k+1} = x_k + \alpha U_i t_k^{(i)}$
8: end for
```

3.1. Block structure and selection of coordinates (Steps 2 & 4). One of the crucial ideas of this algorithm is that the block of coordinates to be updated at each iteration is chosen *randomly*. This allows the coordinates to be selected very quickly. In this section we explain in detail, how the blocks are selected/sampled at each iteration. We also give examples of how coordinates can be randomly sampled such that Assumption 3.1 is satisfied.

3.1.1. Ψ is block separable with $n < N$. When Ψ has a fixed block structure (i.e., $n < N$), the block decomposition of $\mathbb{R}^N$ (via the matrix $U = [U_1, \ldots, U_n]$) described in Section 2.4 is fixed at the start of the algorithm to coincide with the block structure of Ψ, and does not change as iterations progress. There are several ways to initialize a sampling scheme to use in RCD that follow Assumption 3.1.

1. Fix the n blocks of coordinates according to the decomposition of $\mathbb{R}^N$ defined by U, and in the algorithm, select each block of coordinates with some probability p_i. (e.g., uniform probabilities $p_i = 1/n > 0$ for all $i = 1, \ldots, n$).
2. Perform (single) coordinate descent, where at each iteration of RCD, the coordinate i is selected with some probability p_i (e.g., uniform probabilities $p_i = 1/N$ for all i).
3. Perform block coordinate descent, where each block of coordinates has cardinality $N_{\min} := \min\{N_1, \ldots, N_n\}$. The restriction is that, at any iteration k, the sampled coordinates forming block i, must all belong to the same block

of N_j coordinates defined by submatrix U_j. (i.e., Assumption 3.1 is satisfied because the decomposition of U is obeyed.) Recall that Ψ is separable into n blocks. Let the total number of subblocks be $l (> n)$, where we assume that each coordinate $1, \ldots, N$ appears in at least one of the l blocks. Then each subblock is selected with probability p_i.

3.1.2. Ψ is separable with $n = N$.

When Ψ is separable into coordinates, we have complete control over the indices that are updated at each iteration.

Let τ denote the block size (number of coordinates that are updated at any iteration k), where $1 \leq \tau \leq N$. Note that there are ${}^N C_\tau$ subsets[1] of τ coordinates that can be made from the set $\{1, \ldots, N\}$. At any iteration k of RCD, a subset of coordinates i_k with $|i_k| = \tau$ is sampled with some probability p_i (e.g., uniform probabilities $p_i = 1/{}^N C_\tau > 0$ for all i). Note that in practice, one never explicitly forms the ${}^N C_\tau$ different blocks in order to randomly pick one with some probability p_i. Instead, τ coordinates are sampled randomly without replacement.

3.2. The search direction and Hessian approximation (Step 5).

In this section we describe how RCD determines the search direction. In particular, RCD forms a quadratic model for block i, and minimizes the model approximately until the stopping conditions (3.2) are satisfied, giving an 'inexact' search direction.

We also describe the importance of the choice of matrix H, which is an approximate second order information term. From now on, we will often use the shorthand $H_k^{(i)} \equiv H^{(i)}(x_k)$.

3.2.1. The search direction.

At each iteration the update/search direction is found as follows. The subproblem (3.1), (where $Q_i(x_k; t^{(i)})$ is defined in (2.11)) is approximately solved, and the search direction $t_k^{(i)}$ is accepted when the stopping conditions (3.2) are satisfied, for some $\eta_k^i \in [0, 1)$. Notice that

$$
\begin{aligned}
Q(x; U_i t^{(i)}) - Q(x; 0) &\overset{(2.10)}{=} \langle \nabla_i f(x), t^{(i)} \rangle + \frac{1}{2}\|t^{(i)}\|_{H^{(i)}}^2 + \Psi(x + U_i t^{(i)}) - \Psi(x) \\
&\overset{(2.7)}{=} \langle \nabla_i f(x), t^{(i)} \rangle + \frac{1}{2}\|t^{(i)}\|_{H^{(i)}}^2 + \Psi_i(x^{(i)} + t^{(i)}) - \Psi_i(x^{(i)}). \qquad (3.5)
\end{aligned}
$$

Hence, from (3.5), the stopping conditions (3.2) depend on block i only, and are therefore inexpensive to verify, meaning that they are *implementable*.

REMARK 3.2.

(i) At some iteration k, it is possible that $g_i(x_k; 0) = 0$. In this case, it is easy to verify that the optimal solution of subproblem (3.1) is $t_k^{(i)} = 0$. Therefore, before calculating $t_k^{(i)}$ we check a-priori if condition $g_i(x_k; 0) = 0$ is satisfied.

(ii) Notice that, unless at optimality (i.e., $g(x_k; 0) = 0$), there will always be blocks i such that $g_i(x_k; 0) \neq 0$, which implies that $t_k^{(i)} \neq 0$. Hence, RCD will not stagnate.

(iii) Following similar arguments as those made in [2, p.4], we prove this in Lemma 4.3 that both conditions are required to ensure that $t_k^{(i)}$ is a descent direction.

3.2.2. The Hessian approximation.

Arguably, them most important feature of this method is that the quadratic model (2.11) incorporates second order information in the form of a positive definite matrix $H_k^{(i)}$. This is key because, depending

[1] Here ${}^N C_\tau$ denotes the usual 'N choose τ'. i.e., ${}^N C_\tau = N!/(\tau!(N - \tau))!$

upon the choice of $H_k^{(i)}$, it makes the method robust. Moreover, at each iteration, the user has complete freedom over the choice of $H_k^{(i)} \succ 0$.

We now provide a few suggestions for the choice of $H_k^{(i)}$. (This list is not intended to be exhaustive.) Notice that in each case there is a trade off between a matrix that is inexpensive to work with, and one that is a more accurate representation of the true block Hessian.

1. Clearly, the simplest option is to set $H_k^{(i)} = I$ for all i and k. In this case *no second order information is employed by the method.*
2. A second option is to let $H_k^{(i)} = \mathrm{diag}(\nabla_i^2 f(x_k))$. In this case $H_k^{(i)}$ and it's inverse are inexpensive to work with. Moreover, if f is quadratic, then $\nabla^2 f(x_k)$ is constant for all k, so $H = \mathrm{diag}(\nabla^2 f(x))$ can be computed and stored at the start of the algorithm and elements can be accessed throughout the algorithm as necessary. This is very effective if $\mathrm{diag}(\nabla^2 f(x))$ is a good approximation to $\nabla^2 f(x)$.
3. A third option is to let $H_k^{(i)} = \nabla_i^2 f(x_k)$ (i.e., $H_k^{(i)}$ is a principal minor of the Hessian). In this case, $H_k^{(i)}$ provides the most accurate second order information, but it is (potentially) more computationally expensive to work with. In practice the matrix $\nabla_i^2 f(x_k)$ is used in a matrix-free way and is not explicitly stored. For example, there may be an analytic formula for performing matrix-vector products with $\nabla_i^2 f(x_k)$, or techniques from automatic differentiation could be employed, see [18, Section 7].
4. Another option is to use a quasi-Newton type approach where $H_k^{(i)}$ is an approximation to $\nabla_i^2 f(x_k)$ based on the limited-memory BFGS update scheme, see [18, Section 8]. This approach might be more suitable in cases that the problem is not very ill-conditioned and additionally performing matrix-vector products with $\nabla_i^2 f(x_k)$ is expensive.

REMARK 3.3. *If any of the matrices above are not positive definite, then they can be altered to make them so. For example, if $H_k^{(i)}$ is diagonal, any zero that appears on the diagonal can be replaced with a positive number. Moreover, if $\nabla_i^2 f(x_k)$ is not positive definite, a multiple of the identity can be added to it.*

An advantage of the RCD algorithm (if Option 3 is used for $H_k^{(i)}$) is that *all elements of the Hessian can be accessed.* This is because the blocks of coordinates can change at every iteration, and so too can matrix $H_k^{(i)}$. This makes RCD extremely *flexible* and is particularly advantageous when there are large off diagonal elements in the Hessian.

3.3. The line search (Step 6). The stopping conditions (3.2) ensure that $t_k^{(i)}$ is a descent direction, but if the full step $x_k + U_i t_k^{(i)}$ is taken, a reduction in the function value (1.1) is not guaranteed. To this end, we include a line search step in our algorithm in order to guarantee monotonic decrease of function F. Essentially, the line search guarantees the sufficient decrease of F at every iteration, where sufficient decrease is measured by the loss function (3.4).

In particular, for fixed $\theta \in (0, 1/2)$, we require that for some $\alpha \in (0, 1]$, (3.3) is satisfied. (In Lemma 4.3 we prove that there exists a subinterval $(0, \tilde{\alpha}]$ of $(0, 1]$ in which (3.3) is satisfied.) Notice that

$$\begin{aligned} \ell(x; U_i t^{(i)}) - \ell(x; 0) &\overset{(3.4)}{=} \langle \nabla_i f(x), t^{(i)} \rangle + \Psi(x + U_i t^{(i)}) - \Psi(x) \\ &\overset{(2.7)}{=} \langle \nabla_i f(x), t^{(i)} \rangle + \Psi_i(x^{(i)} + t^{(i)}) - \Psi_i(x^{(i)}), \qquad (3.6) \end{aligned}$$

which shows that the calculation of the right hand side of (3.3) only depends upon block i, so it is inexpensive. Moreover, the line search condition (Step 5) involves the difference between function values $F(x_k) - F(x_k + \alpha U_i t_k^{(i)})$. Fortunately, while function values can be expensive to compute, the difference in the objective value between iterates need not be (this is discussed in more detail in Section 3.4).

3.4. Examples. In this section we provide several examples to demonstrate the practicality of the algorithm. These examples demonstrate that the difference of function values $F(x_k) - F(x_k + \alpha U_i t_k^{(i)})$ required by the line search conditions (3.3), can be easy/inexpensive to implement and verify.

3.4.1. Quadratic loss plus regularization example. Suppose that $f(x) = \frac{1}{2}\|Ax - b\|^2$ and $\Psi(x) \neq 0$, where $A \in \mathbb{R}^{m \times N}$, $b \in \mathbb{R}^m$ and $x \in \mathbb{R}^N$. Then

$$\begin{aligned} F(x_k) - F(x_k + \alpha U_i t_k^{(i)}) &\overset{(2.7)}{=} f(x_k) + \Psi_i(x^{(i)}) \\ &\quad - f(x_k + \alpha U_i t_k^{(i)}) - \Psi_i(x_k^{(i)} + \alpha t_k^{(i)}) \\ &= \Psi_i(x^{(i)}) - \alpha \langle \nabla_i f(x), t_k^{(i)} \rangle - \frac{\alpha^2}{2}\|A_i t_k^{(i)}\|_2^2 \\ &\quad - \Psi_i(x_k^{(i)} + \alpha t_k^{(i)}). \end{aligned} \tag{3.7}$$

Notice that calculation of $F(x_k) - F(x_k + \alpha U_i t_k^{(i)})$ as a function of α only depends on block i, hence, it is inexpensive. Moreover, in some cases some of the quantities in (3.7) are already needed in the computation of the search direction t, so regarding the line search step, they essentially come “for free”.

3.4.2. Logistic regression example. Suppose that

$$f(x) \equiv \sum_{j=1}^{m} \log(1 + e^{-b_j a_j^T x}) \quad \text{and} \quad \Psi(x) \neq 0,$$

where a_j^T is the jth row of a matrix $A \in \mathbb{R}^{m \times n}$ and b_j is the jth component of vector $b \in \mathbb{R}^m$. As before, we need to evaluate (3.7). Let us split calculation of $F(x_k) - F(x_k + \alpha U_i t_k^{(i)})$ in parts. The first part $\Psi_i(x^{(i)}) - \Psi_i(x_k^{(i)} + \alpha t_k^{(i)})$ is inexpensive, since it depends only on block i. The second part $f(x_k) - f(x_k + \alpha U_i t_k^{(i)})$ is more expensive because is depends upon the logarithm. In this case, one can calculate $f(x_0)$ *once* at the beginning of the algorithm and then update $f(x_k + \alpha U_i t_k^{(i)})$ $\forall k \geq 1$ less expensively. In particular, let us assume that the following terms:

$$e^{-b_j a_j^T x_0} \quad \forall j \quad \text{and} \quad f(x_0) = \sum_{j=1}^{m} \log(1 + e^{-b_j a_j^T x_0}), \tag{3.8}$$

are calculated once and stored in memory. Then, at iteration 1, the calculation of $f(x_0 + \alpha U_i t_0^{(i)}) = \sum_{j=1}^{m} \log(1 + e^{-b_j a_j^T x_0} e^{-\alpha b_j a_j^T (U_i t_0^{(i)})})$ is required for different values of α by the backtracking line search algorithm. The most demanding task in calculating $f(x_0 + \alpha U_i t_0^{(i)})$ is the calculation of the products $b_j a_j^T (U_i t_0^{(i)})$ $\forall j$ *once*, which is inexpensive since $\forall j$ this operation depends only on block i. Having $b_j a_j^T (U_i t_0^{(i)})$ $\forall j$ and (3.8) calculation of $f(x_0) - f(x_0 + \alpha U_i t_0^{(i)})$ for different values of α is inexpensive. At the end of the process, $f(x_1)$ and $e^{-b_j a_j^T x_1}$ $\forall j$ will be given for free, and the same process can be followed for the calculation of $f(x_1) - f(x_1 + \alpha U_i t_1^{(i)})$ etc.

4. Global convergence theory without convexity of f. In this section we provide global convergence theory for the RCD algorithm. Note that we *do not assume that f is convex.* Throughout this section we denote $H_k^{(i)} \equiv H^{(i)}(x_k)$. The following assumptions are made about $H_k^{(i)}$ and f.

Assumption 4.1. *There exist constants $0 < \lambda_i \leq \Lambda_i$, such that the sequence $\{H_k^{(i)}\}_{k\geq 0}$ satisfies*

$$0 < \lambda_i \leq \lambda_{\min}(H_k^{(i)}) \quad \textit{and} \quad \lambda_{\max}(H_k^{(i)}) \leq \Lambda_i, \quad \textit{for all } i \textit{ and } k. \tag{4.1}$$

Assumption 4.2. *The function f is smooth, bounded below, and satisfies* (2.8) *for all i.*

Assumption 4.1 explains that the Hessian approximation $H_k^{(i)}$ must be positive definite for all blocks i at all iterations k. Assumption 4.2 explains that f must be block Lipschitz for all blocks i and all iterations k.

Before proving global convergence of RCD, we present several technical results. The following lemma shows that if $t_k^{(i)}$ is nonzero, then F is decreased.

Lemma 4.3. *Let Assumptions 4.1 and 4.2 hold. Let $\theta \in (0, 1/2)$ and let x_k and i be generated by RCD. Then Step 6 of RCD will accept a step-size α that satisfies*

$$\alpha \geq \tilde{\alpha} \qquad \textit{where} \qquad \tilde{\alpha} := (1-\theta)\frac{\lambda_i}{2L_i}. \tag{4.2}$$

Furthermore,

$$F(x_k) - F(x_k + \alpha U_i t_k^{(i)}) > \theta(1-\theta)\frac{\lambda_i^2}{4L_i}\|t_k^{(i)}\|^2. \tag{4.3}$$

Proof. The proof closely follows that of [2, Theorem 3.1]. From (3.2),

$$0 > Q(x_k; U_i t_k^{(i)}) - Q(x_k; 0) = \ell(x_k; U_i t_k^{(i)}) - \ell(x_k; 0) + \frac{1}{2}\|t_k^{(i)}\|^2_{H_k^{(i)}}. \tag{4.4}$$

Rearranging gives

$$\ell(x_k; 0) - \ell(x_k; U_i t_k^{(i)}) > \frac{1}{2}\|t_k^{(i)}\|^2_{H_k^{(i)}} \overset{(4.1)}{\geq} \frac{1}{2}\lambda_i\|t_k^{(i)}\|^2. \tag{4.5}$$

By Assumption 4.2, for $\alpha \in (0, 1)$, we have

$$F(x_k + \alpha U_i t_k^{(i)}) \leq f(x_k) + \alpha\langle\nabla_i f(x), t_k^{(i)}\rangle + \frac{L_i}{2}\alpha^2\|t_k^{(i)}\|^2 + \Psi(x_k + \alpha U_i t_k^{(i)}).$$

Adding $\Psi(x_k)$ to both sides of the above and rearranging gives

$$\begin{aligned} F(x_k) - F(x_k + \alpha U_i t_k^{(i)}) \;&\geq\; -\alpha\langle\nabla_i f(x_k), t_k^{(i)}\rangle - \frac{L_i}{2}\alpha^2\|t_k^{(i)}\|^2 \\ &\quad\; -\Psi(x_k + \alpha U_i t_k^{(i)}) + \Psi(x_k) \\ &\overset{(3.6)}{=} \ell(x_k; 0) - \ell(x_k; \alpha U_i t_k^{(i)}) - \frac{L_i}{2}\alpha^2\|t_k^{(i)}\|^2. \end{aligned} \tag{4.6}$$

By convexity of $\Psi(x)$ we have that

$$\ell(x_k; 0) - \ell(x_k; \alpha U_i t_k^{(i)}) \geq \alpha(\ell(x_k; 0) - \ell(x_k; U_i t_k^{(i)})). \tag{4.7}$$

Then

$$\begin{aligned}F(x_k) - F(x_k + \alpha U_i t_k^{(i)}) \;&-\; \theta(\ell(x_k;0) - \ell(x_k;\alpha U_i t_k^{(i)}))\\ &\overset{(4.6)}{\geq} (1-\theta)\big(\ell(x_k;0) - \ell(x_k;\alpha U_i t^{(i)})\big) - \frac{L_i}{2}\alpha^2\|t_k^{(i)}\|^2\\ &\overset{(4.7)}{\geq} \alpha(1-\theta)\big(\ell(x_k;0) - \ell_i(x_k;U_i t_k^{(i)})\big) - \frac{L_i}{2}\alpha^2\|t_k^{(i)}\|^2\\ &\overset{(4.5)}{>} \frac{1}{2}\big(\alpha(1-\theta)\lambda_i\|t_k^{(i)}\|^2 - L_i\alpha^2\|t_k^{(i)}\|^2\big)\\ &= \frac{\alpha}{2}\big((1-\theta)\lambda_i - L_i\alpha\big)\|t_k^{(i)}\|^2.\end{aligned}$$

From the previous observe that if α satisfies $0 \leq \alpha \leq (1-\theta)\frac{\lambda_i}{L_i}$, then α also satisfies the backtracking line search step of RCD. Suppose that any α that is rejected by the line search is halved for the next line search trial. Then, it is guaranteed that the α that is accepted satisfies (4.2).

Since the line search condition (3.3) is guaranteed to be satisfied for some step size α, from (3.3) and convexity of ℓ we obtain $F(x_k) - F(x_k + \alpha U_i t_k^{(i)}) \geq \theta\alpha(\ell(x_k;0) - \ell(x_k;U_i t_k^{(i)}))$. Using (4.5) and (4.2) in the last inequality we obtain (4.3). ∎

The following lemma bounds the norm of the direction $t_k^{(i)}$ in terms of $g_i(x_k,0)$.

Lemma 4.4. *Let Assumptions 4.1 and 4.2 hold. For $\beta > 0$, and x_k and i generated by RCD, we have*

$$\|t_k^{(i)}\| \geq \gamma_i\|g_i(x_k;0)\|, \quad \textit{where} \quad \gamma_i := \frac{1-\eta_k^i}{\frac{1}{\beta} + 2\Lambda_i}, \tag{4.8}$$

where η_k^i is defined in (3.2)*. Moreover,*

$$F(x_k) - F(x_k + \alpha U_i t_k^{(i)}) > \theta(1-\theta)\frac{\lambda_i^2}{4L_i}\gamma_i^2\|g_i(x_k;0)\|^2. \tag{4.9}$$

Proof. This proof closely follows that of [2, Theorem 3.1]. Using the reverse triangular inequality and the fact that $\text{prox}_{\Psi_i^*}(\cdot)$ is nonexpansive, we have that

$$\begin{aligned}(1-\eta_k^i)\|g_i(x_k;0)\| &\overset{(3.2)}{\leq} \|g_i(x_k;0)\| - \|g_i(x_k;t_k^{(i)})\|\\ &\leq \|g_i(x_k;t_k^{(i)}) - g_i(x_k;0)\|\\ &\overset{(2.13)}{=} \|H_k^{(i)}t_k^{(i)} + \tfrac{1}{\beta}\text{prox}_{(\beta\Psi)_i^*}\big(x_k^{(i)} + t_k^{(i)} - \beta(\nabla_i f(x_k) + H_k^{(i)}t_k^{(i)})\big)\\ &\quad -\tfrac{1}{\beta}\text{prox}_{(\beta\Psi)_i^*}\big(x_k^{(i)} - \beta\nabla_i f(x_k)\big)\|\\ &\overset{(2.3)}{\leq} \|H_k^{(i)}t_k^{(i)}\| + \tfrac{1}{\beta}\|(I - \beta H_k^{(i)})t_k^{(i)}\|\\ &\leq (\tfrac{1}{\beta} + 2\|H_k^{(i)}\|)\|t_k^{(i)}\|\\ &\leq (\tfrac{1}{\beta} + 2\Lambda_i)\|t_k^{(i)}\|.\end{aligned}$$

Rearranging gives (4.8), and combining (4.3) and (4.8) gives (4.9).
∎

We now have all the tools to prove global convergence of RCD.

Theorem 4.5. *Let Assumptions 4.1 and 4.2 hold. Then the following hold for RCD:*

$$\lim_{k\to\infty} t_k^{(i)} = 0 \quad \forall i \quad \textit{and} \quad \lim_{k\to\infty} g(x_k;0) = 0.$$

with probability one.

Proof. By Lemma 4.3, for $t_k^{(i)} \neq 0$ we have that $F(x_k) > F(x_k + \alpha U_i t_k^{(i)})$. Since F is bounded from below and every block (i) has positive probability to be selected, then for $k \to \infty$ the following holds with probability one:

$$\lim_{k\to\infty} F(x_k) - F(x_k + \alpha U_i t_k^{(i)}) = 0. \tag{4.10}$$

Using (4.3) in combination with (4.10) we get that for $k \to \infty$ with probability one $\|t_k^{(i)}\| \to 0$, which proves the first part. Using (4.8) and $\|t_k^{(i)}\| \to 0$ for $k \to \infty$ with probability one we also get that $\|g_i(x_k;0)\| \to 0$ with probability one for $k \to \infty$ Since $g_i(x_k;0)$ is a block of $g(x_k;0)$, i.e., $g_i(x_k;0) \overset{(2.5)}{=} U_i^T g(x_k;0)$ and since every block $g_i(x_k;0)$ tends to zero as $k \to \infty$ with probability one, then we have that $g(x_k;0) \to 0$ with probability one. □

5. Local convergence theory. In this section we present local convergence theory for RCD. First we discuss some common assumptions that are needed. Throughout the section we set $H_k^{(i)} = \nabla_i^2 f(x_k)\ \forall i$, where $\nabla_i^2 f(x)$ denotes the principal minor of the Hessian $\nabla^2 f(x)$ with row (equivalently column) indices in the subset i.

Assumption 5.1. *Function f is strongly convex with strong convexity parameter $\mu_f > 0$.*

By continuity of f we have that $\nabla^2 f(x)$ is symmetric, and by strong convexity of f we have that $\mu_f I \preceq \nabla^2 f(x)$.

The next theorem explains that, if the Hessian $H \equiv \nabla^2 f(x)$ is positive definite, then every principal submatrix of it is also positive definite.

Theorem 5.2 (Theorem 4.3.15 in [9]). *Let $A \in \mathbb{R}^{N\times N}$ be a Hermitian matrix, let r be an integer with $1 \le r \le N$, and let A_r denote any $r \times r$ principal submatrix of A (obtained by deleting $N - r$ rows and the corresponding columns from A). For each integer k such that $1 \le k \le r$ we have $\lambda_k(A) \le \lambda_k(A_r) \le \lambda_{k+N-r}(A)$, where $\lambda_k(\cdot)$ denotes the kth eigenvalue of matrix $\cdot$, and the eigenvalues are ordered $\lambda_1 \le \dots, \lambda_N$.*

Corollary 5.3. *If f is strongly convex with strong convexity parameter $\mu_f > 0$, then*

$$\mu_f I \preceq \nabla_i^2 f(x) \qquad \textit{for all } i \subseteq \{1,\dots,N\}. \tag{5.1}$$

Assumption 5.4. *We assume that the blocks of the Hessian of f are Lipschitz continuous. This means that for all $x \in \mathbb{R}^N$, $i \subseteq \{1,2,\dots,n\}$ and $t^{(i)} \in \mathbb{R}^{N_i}$ we have*

$$\|\nabla_i^2 f(x + U_i t^{(i)}) - \nabla_i^2 f(x)\| \le M_i \|t^{(i)}\|. \tag{5.2}$$

The following theorem is used to show that the backtracking line search accepts units step sizes close to optimality for any block i. Similarly to [2], in order to prove the previous statement we have to impose sufficient decrease of the quadratic model

(3.1) at every iteration. This means that the inexactness conditions in (3.2) are replaced by

$$\begin{aligned}\xi(\ell(x_k;0)-\ell(x_k;U_it_k^{(i)})) &\le Q(x_k;0)-Q(x_k;U_it_k^{(i)}),\\ \text{and } \|g_i(x_k;t_k^{(i)})\| &\le \eta_k^i\|g_i(x_k;0)\|,\end{aligned} \tag{5.3}$$

where $\xi \in (\theta, 1/2)$ and $\eta_k^i \in [0,1)$.

Notice that, substituting the equality in (4.4) into (5.3), gives

$$\frac{1}{2}\|t_k^{(i)}\|^2_{H_k^{(i)}} \le (1-\xi)\big(\ell(x_k;0)-\ell(x_k;U_it_k^{(i)})\big). \tag{5.4}$$

THEOREM 5.5. *Let Assumptions 5.1 and 5.4 hold. Let x_k and i be generated by RCD. Moreover, let subproblem* (3.1) *of RCD be solved inexactly until the inexactness conditions* (5.3) *are satisfied. If $\|t_k^{(i)}\| \le 3\lambda_i(\xi-\theta)/M_i$, where $\theta \in (0,1/2)$, $\xi \in (\theta,1/2)$ and $\lambda_i \equiv \lambda_{\min}(H_k^{(i)}) \ge \mu_f > 0$ $\forall i$ (by Assumption 5.1), then the backtracking line search step in RCD accepts step sizes $\alpha = 1$.*

Proof. The proof closely follows that of [12, Lemma 3.3]. Using Lipschitz continuity of $H_k^{(i)}$ we have that

$$f(x_k+U_it_k^{(i)}) \le f(x_k) + \langle \nabla_i f(x_k), t_k^{(i)}\rangle + \tfrac{1}{2}\|t_k^{(i)}\|^2_{H_k^{(i)}} + \tfrac{M_i}{6}\|t_k^{(i)}\|^3.$$

Adding $\Psi(x_k+U_it_k^{(i)})$ to both sides gives

$$\begin{aligned}F(x_k+U_it_k^{(i)}) &\le f(x_k) + \langle \nabla_i f(x_k), t_k^{(i)}\rangle + \tfrac{1}{2}\|t_k^{(i)}\|^2_{H_k^{(i)}} + \tfrac{M_i}{6}\|t_k^{(i)}\|^3 + \Psi(x_k+U_it_k^{(i)})\\ &= F(x_k) + \langle \nabla_i f(x_k), t_k^{(i)}\rangle + \tfrac{1}{2}\|t_k^{(i)}\|^2_{H_k^{(i)}} + \tfrac{M_i}{6}\|t_k^{(i)}\|^3\\ &\quad +\Psi(x_k+U_it_k^{(i)}) - \Psi(x_k)\\ &= F(x_k) - (\ell(x_k;0)-\ell(x_k;U_it_k^{(i)})) + \tfrac{1}{2}\|t_k^{(i)}\|^2_{H_k^{(i)}} + \tfrac{M_i}{6}\|t_k^{(i)}\|^3\\ &\overset{(5.4)}{\le} F(x_k) - \xi(\ell(x_k;0)-\ell(x_k;U_it_k^{(i)})) + \tfrac{M_i}{6}\|t_k^{(i)}\|^3. \end{aligned} \tag{5.5}$$

Clearly, $\lambda_i \equiv \lambda_{\min}(H_k^{(i)}) \le \|t_k^{(i)}\|^2_{H_k^{(i)}}/\|t_k^{(i)}\|^2$. Using this with (5.4) in (5.5) we get

$$\begin{aligned}F(x_k+U_it_k^{(i)}) &\le F(x_k) - \xi(\ell(x_k;0)-\ell(x_k;U_it_k^{(i)}))\\ &\quad + \tfrac{M_i}{3\lambda_i}\|t_k^{(i)}\|(\ell(x_k;0)-\ell(x_k;U_it_k^{(i)}))\\ &= F(x_k) - (\xi - \tfrac{M_i}{3\lambda_i}\|t_k^{(i)}\|)(\ell(x_k;0)-\ell(x_k;U_it_k^{(i)})).\end{aligned}$$

If $\|t_k^{(i)}\| \le 3\lambda_i(\xi-\theta)/M_i$, $\theta \in (0,\frac{1}{2})$ and $\xi \in (\theta, 1/2)$ then $F(x_k) - F(x_k+U_it_k^{(i)}) \ge \theta(\ell(x_k;0)-\ell(x_k;U_it_k^{(i)}))$, which implies that RCD accepts a step $\alpha = 1$. □

COROLLARY 5.6. *By Theorem 4.5, $\|t_k^{(i)}\| \to 0$ $\forall i$ as $k \to \infty$. Thus, there will be a region close to the optimal solution x_* such that $\|t_k^{(i)}\| \le 3\lambda_i(\xi-\theta)/M_i$ for all i and for all k. Hence, by Theorem 5.5, in this region, the backtracking line search step in RCD accepts unit step sizes for any i.*

The following assumption is mild since it is guaranteed to be satisfied by Corollary 5.6.

Assumption 5.7. *Iteration x_k is close to the optimal solution x_* of* (1.1) *such that unit step sizes are accepted by the backtracking line search algorithm of RCD.*

The next lemma is a technical result that will be used in Theorem 5.9.

Lemma 5.8. *Let Assumptions 4.2 and 5.1 hold. Let $L_{\max} = \max_i\{L_1, \ldots, L_n\}$. Let $\beta < 1/L_{\max}$ in* (2.13). *Then $g_i(x; t^{(i)})$ inherits strong monotonicity of $\nabla_i f(x)$ $\forall i$:*

$$(u - v)^T (g_i(x; u) - g_i(x; v)) \geq \frac{\mu_f}{2} \|u - v\|^2 \quad \forall u, v \in \mathbb{R}^{N_i}.$$

Proof. The proof is the same as [12, Lemma 3.9], but restricted to the ith block, so is omitted. □

In the following theorem we demonstrate that RCD has on expectation *block* quadratic or superlinear local rate of convergence.

Theorem 5.9. *Let Assumptions 4.2, 5.1, 5.4 and 5.7 hold. Let $x_{k+1} = x_k + t_k^{(i)}$, $\eta_k^i = \min\{1/2, \|g_i(x_k; 0)\|\}$, and $\beta < 1/L_{\max}$ in* (2.13). *Then, $\|g_i(x_k; 0)\|$ has a quadratic rate of convergence in expectation:*

$$\lim_{k\to\infty} \mathbf{E}\left[\frac{\|g_i(x_{k+1}; 0)\|}{\|g_i(x_k; 0)\|^2} \mid x_k\right] = c,$$

where c is a positive constant. If $\eta_k^i \to 0$ for $k \to \infty$, then $\|g_i(x_k; 0)\|$ has a superlinear rate of convergence in expectation:

$$\lim_{k\to\infty} \mathbf{E}\left[\frac{\|g_i(x_{k+1}; 0)\|}{\|g_i(x_k; 0)\|} \mid x_k\right] = 0.$$

Proof. For a given x_k we define

$$x(\delta) := x_k + \delta U_i t_k^{(i)}, \quad \text{and} \quad x(\sigma) := x_k + \sigma U_i t_k^{(i)}. \tag{5.6}$$

Using the Fundamental Theorem of Calculus (F.T.o.C.), we have

$$\begin{aligned} g_i(x(\delta); 0) = \nabla_i f(x_k) + \delta \nabla_i^2 f(x_k) t_k^{(i)} + \int_0^\delta \int_0^u \nabla_i^3 f(x(\sigma))[t_k^{(i)}, t_k^{(i)}] d\sigma du \\ + \frac{1}{\beta} \text{prox}_{(\beta\Psi)_i^*}(x^{(i)}(\delta) - \beta \nabla_i f(x(\delta))). \end{aligned} \tag{5.7}$$

Also, from the definition of a derivative we have

$$\begin{aligned} &\int_0^\delta \int_0^u \|\nabla_i^3 f(x(\sigma))[t_k^{(i)}, t_k^{(i)}]\| d\sigma du \\ &\quad = \int_0^\delta \int_0^u \lim_{\sigma\to 0} \|\frac{1}{\sigma}(t_k^{(i)})^T (\nabla_i^2 f(x(\sigma)) - \nabla_i^2 f(x_k))\| d\sigma du \\ &\quad \leq \|t_k^{(i)}\| \int_0^\delta \int_0^u \lim_{\sigma\to 0} \|\frac{1}{\sigma}(\nabla_i^2 f(x(\sigma)) - \nabla_i^2 f(x_k))\| d\sigma du \\ &\quad \overset{(5.2)}{\leq} M_i \|t_k^{(i)}\|^2 \int_0^\delta \int_0^u 1 \, d\sigma du = \frac{\delta^2}{2} M_i \|t_k^{(i)}\|^2. \end{aligned} \tag{5.8}$$

Now, adding and subtracting $\frac{1}{\beta}\text{prox}_{(\beta\Psi)_i^*}(x_k^{(i)} + t_k^{(i)} - \beta(\nabla_i f(x_k) + H_k^{(i)} t_k^{(i)}))$ from (5.7), followed by taking norms, applying the triangle inequality, and using (5.8), gives

$$\begin{aligned}\|g_i(x(\delta);0)\| \leq \|\nabla_i f(x_k) + \delta\nabla_i^2 f(x_k) t_k^{(i)} \\ &+ \frac{1}{\beta}\text{prox}_{(\beta\Psi)_i^*}(x_k^{(i)} + t_k^{(i)} - \beta(\nabla_i f(x_k) + H_k^{(i)} t_k^{(i)}))\| \\ &+ \frac{1}{\beta}\|\text{prox}_{(\beta\Psi)_i^*}(x^{(i)}(\delta) - \beta\nabla_i f(x(\delta))) \\ &- \text{prox}_{(\beta\Psi)_i^*}(x_k^{(i)} + t_k^{(i)} - \beta(\nabla_i f(x_k) + H_k^{(i)} t_k^{(i)}))\| + \frac{\delta^2}{2} M_i \|t_k^{(i)}\|^2.\end{aligned} \tag{5.9}$$

By Assumption 5.7, RCD accepts unit step sizes. Hence, setting $\delta = 1$ in (5.9) gives

$$\begin{aligned}\|g_i(x_{k+1};0)\| &\leq \|g_i(x_k;t_k^{(i)})\| + \frac{1}{2} M_i \|t_k^{(i)}\|^2 \\ &\quad + \frac{1}{\beta}\|\text{prox}_{(\beta\Psi)_i^*}(x_{k+1}^{(i)} - \beta\nabla_i f(x_{k+1})) \\ &\quad - \text{prox}_{(\beta\Psi)_i^*}(x_{k+1}^{(i)} - \beta(\nabla_i f(x_k) + H_k^{(i)} t_k^{(i)}))\| \\ &\leq \|g_i(x_k;t_k^{(i)})\| + \|\nabla_i f(x_k) + H_k^{(i)} t_k^{(i)} - \nabla_i f(x_{k+1})\| + \frac{1}{2} M_i \|t_k^{(i)}\|^2.\end{aligned}$$

Using the same trick as before with the F.T.o.C. we have the bound $\|\nabla_i f(x_k) + H_k^{(i)} t_k^{(i)} - \nabla_i f(x_{k+1})\| \leq \frac{1}{2} M_i \|t_k^{(i)}\|^2$, so that

$$\begin{aligned}\|g_i(x_{k+1};0)\| &\leq \|g_i(x_k;t_k^{(i)})\| + M_i\|t_k^{(i)}\|^2 \\ &\overset{(5.3)}{\leq} \eta_k^i \|g_i(x_k;0)\| + M_i \|t_k^{(i)}\|^2.\end{aligned} \tag{5.10}$$

Setting $u = t_k^{(i)}$ and $v = 0$ in Lemma 5.8 gives $(g_i(x_k;t_k^{(i)}) - g_i(x_k;0))^T t_k^{(i)} \geq \frac{\mu_f}{2}\|t_k^{(i)}\|^2$. Then, using Cauchy-Schwarz we have

$$\|g_i(x_k;t_k^{(i)}) - g_i(x_k;0)\| \geq \frac{\mu_f}{2}\|t_k^{(i)}\|.$$

By the triangular inequality and stopping conditions (3.2) we have

$$(1+\eta_k^i)\|g_i(x_k;0)\| \geq \frac{\mu_f}{2}\|t_k^{(i)}\|. \tag{5.11}$$

Replacing (5.11) in (5.10) we have

$$\|g_i(x_{k+1};0)\| \leq \eta_k^i \|g_i(x_k;0)\| + \frac{4M_i(1+\eta_k^i)^2}{\mu_f^2}\|g_i(x_k;0)\|^2. \tag{5.12}$$

Moreover, by setting $\eta_k^i = \min\{1/2, \|g_i(x_k;0)\|\}$ we obtain

$$\|g_i(x_{k+1};0)\| \leq \left(1 + \frac{9M_i}{\mu_f^2}\right)\|g_i(x_k;0)\|^2. \tag{5.13}$$

Rearranging and taking expectation gives

$$\lim_{k\to\infty} \mathbf{E}\left[\frac{\|g_i(x_{k+1};0)\|}{\|g_i(x_k;0)\|^2} \mid x_k\right] \leq \lim_{k\to\infty} \mathbf{E}\left[1 + \frac{9M_i}{\mu_F^2} \mid x_k\right]. \tag{5.14}$$

The right hand side of (5.14) is constant for all i, which implies quadratic convergence of $\|g_i(x_k; 0)\|$ in expectation.

Moreover, if $\eta_k^i \to 0$ as $k \to \infty$, then from (5.12), Theorem 4.5 and Lemma 4.4,

$$\lim_{k\to\infty} \mathbf{E}\left[\frac{\|g_i(x_{k+1}; 0)\|}{\|g_i(x_k; 0)\|} \mid x_k\right] = 0,$$

which implies superlinear convergence of $\|g_i(x_k; 0)\|$ in expectation. □

6. Numerical Experiments. In this section we examine the performance of two versions of RCD and two versions of a Uniform Coordinate Descent method (UCDC) [20] on two common optimization problems. The first problem is an ℓ_1-regularized least squares problem of the form (1.1) with

$$f(x) = \tfrac{1}{2}\|Ax - b\|^2 \quad \text{and} \quad \Psi(x) = c\|x\|_1, \tag{6.1}$$

where $c > 0$, $x \in \mathbb{R}^N$, $A \in \mathbb{R}^{m\times N}$ and $b \in \mathbb{R}^m$. The second problem is an ℓ_1-regularized logistic regression problems of the form (1.1) with

$$f(x) = \sum_{j=1}^{m} \log(1 + e^{-b_j x^T a_j}) \quad \text{and} \quad \Psi(x) = c\|x\|_1, \tag{6.2}$$

where $c > 0$, $a_j \in \mathbb{R}^N$ $\forall j = 1, 2, \dots, m$ are the training samples and $b_j \in \{-1, +1\}$ are the corresponding labels.

For (6.1) a synthetic sparse large scale experiment is performed and for (6.2) we compare the methods on two real world large scale problems from machine learning. Notice that for both (6.1) and (6.2), $\Psi(x) = c\|x\|_1$, which is fully separable into coordinates. This means that, for RCD, we have complete control over the block decomposition, and the indices making up each block can change at every iteration.

All algorithms are coded in MATLAB, and for fairness, MATLAB is limited to a single computational thread for each test run. All experiments are performed on a Dell PowerEdge R920 running Redhat Enterprise Linux with four Intel Xeon E7-4830 v2 2.2GHz, 20M Cache, 7.2 GT/s QPI, Turbo (4x10Cores).

6.1. Implementations of RCD and UCDC. In this section we discuss some details of the implementations of methods RCD and UCDC.

6.1.1. RCD. For the RCD method, we fix the size of blocks $\tau > 1$ (to be given in the numerical experiments subsections), and at every iteration of RCD, τ coordinates are sampled uniformly at random without replacement.

We implement two versions of RCD, which we denote by RCD v.1 and RCD v.2. The two versions only differ in how matrix $H_k^{(i)}$ is chosen. In particular, for RCD v.1 we set $H_k^{(i)} := \operatorname{diag}(\nabla_i^2 f(x_k))$ for all i and k. In this case subproblem (3.1) is separable and it has a closed form solution

$$t_k^{(i)} = \mathcal{S}(x_k^{(i)} - (H_k^{(i)})^{-1}\nabla_i f(x_k^{(i)}), c\,\operatorname{diag}((H_k^{(i)})^{-1})),$$

where

$$\mathcal{S}(u, v) = \operatorname{sign}(u)\max(|u| - v, 0) \tag{6.3}$$

is the well-known soft-thresholding operator which is applied component wise when u and v are vectors. Notice that since the subproblem is solved exactly there is no need to verify the stopping conditions (3.2).

For RCD v.2, we set

$$H_k^{(i)} := \nabla_i^2 f(x_k) + \rho I_{N_i}, \text{ for all } i \text{ and } k, \tag{6.4}$$

where $\rho > 0$ guarantees that $H_k^{(i)}$ is positive definite for all i, k. Hence, the subproblem (3.1) is well defined. The larger ρ is the smaller the condition number of matrix H_k becomes, hence, the faster subproblem (3.1) will be solved by an iterative solver. However, we do not want ρ to dominate matrix H_k because the essential second order information from $\nabla^2 f(x_k)$ will be lost.

In this setting of matrix $H_k^{(i)}$ we solve subproblems (3.1) iteratively using an Orthant Wise Limited-memory Quasi-Newton (OWL) method, which can be downloaded from `http://www.di.ens.fr/~mschmidt/Software/L1General.html`. We chose OWL because it has been shown in [2] to result in a robust and efficient deterministic version of RCD, i.e. $\tau = N$ (one block of size N). Note that we never explicitly form matrix $H_k^{(i)}$, we only perform matrix-vector products with it in a matrix-free manner.

6.1.2. UCDC. We also implement two versions of a uniform coordinate descent method as it is described in Algorithm 2 in [20]. For both versions the size τ of the blocks and the decomposition of $\mathbb{R}^N$ into $\lceil N/\tau \rceil$ blocks are *fixed a-priori* and all blocks are selected by UCDC with uniform probability. We compare two versions of UCDC, denoted by UCDC v.1 and UCDC v.2 respectively. For UCDC v.1 we set $\tau = 1$ and for UCDC v.2 we set $\tau > 1$ (the exact τ is given later).

One of the key ingredients of UCDC are the block Lipschitz constants, which are explicitly required in the algorithm. For single coordinate blocks, the Lipschitz constants can be computed with relative ease. However, for blocks of size greater than 1, the block Lipschitz constants can be far more expensive to compute. (For example, for problem (6.1), the block Lipschitz constants correspond to the maximum eigenvalue of $A_i^T A_i$, where $A_i := AU_i$.) For this reason, we do not compute the actual block Lipschitz constants, rather, we use an overapproximation.

To this end, let $L_j > 0 \, \forall j = 1, 2, \cdots, N$ denote the coordinate Lipschitz constants of function f. Then the direction $t^{(i)}$ at every iteration is obtained by solving exactly subproblem (3.1) with

$$H_k^{(i)} := \Big(\sum_{j \in i} L_j\Big) I_\tau, \tag{6.5}$$

using operator (6.3). Notice that for problem (6.1), (6.5) is equivalent to $H_k^{(i)} = \text{trace}(A_i^T A_i) I_\tau$, where $\text{trace}(A_i^T A_i)$ is an overapproximation of the maximum eigenvalue of $A_i^T A_i$.

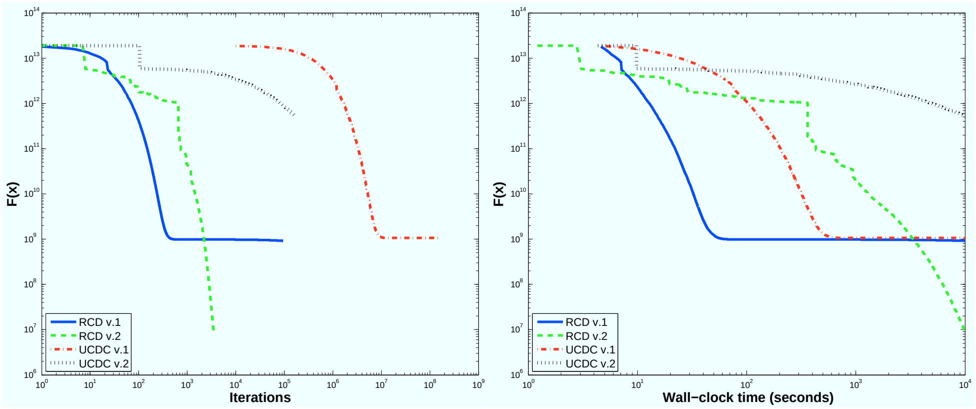
Moreover, notice that Algorithm 2 in [20] is a special case of RCD where the subproblem (3.1) is solved exactly and line search is unnecessary. This is because, by setting $H_k^{(i)}$ as in (6.5), then subproblem (3.1) is an over estimator of function F along block coordinate direction $t^{(i)}$ (for details we refer the reader to [20]).

6.2. Termination Criteria and Parameter Tuning. The only termination criteria that RCD and UCDC should have are maximum number of iterations or maximum running time. This is because using subgradients as a measure of distance from optimality or any other operation of similar cost are considered as expensive tasks for large scale problems. In our experiments RCD and UCDC are terminated when their running time exceeds the maximum allowed running time. Furthermore, for RCD we set parameter $\eta_k^{(i)}$ in (3.2) equal to 0.9 $\forall i, k$ and $\rho = 10^{-6}$ in (6.4). The maximum number of backtracking line search iterations is set to 10 and $\theta = 10^{-3}$. For UCDC the coordinate Lipschitz constants L_j $\forall j$ are calculated once at the beginning of the algorithm and this task is included in the overall running time. Finally, all methods are initialized with the zero solution.

6.3. ℓ_1-Regularized Least Squares. In this subsection we present the performance of RCD and UCDC on the ℓ_1-regularized least squares problem (6.1). For this problem the data A and b were synthetically constructed using a generator proposed in [17, Section 6], and we set $c = 1$. The advantage of this generator is that it produces data A and b with a known minimizer x_*. We slightly modified the generator so that we could control the density of A.

The dimensions of the problem are $N = 2^{21}$ and $m = N/4$ and the generated matrix A is full rank (with at least one non zero component per column) and a density of $\approx 10^{-4}mN$. The optimal solution is set to have $\lceil 0.01N \rceil$ non zero components with values uniformly at random in the interval $[-1, 1]$. For UCDC, the coordinate Lipschitz constants are $L_j := \|A_j\|_2^2 \quad j = 1, 2, \cdots, N$, and for RCD v.1, RCD v.2 and UCDC v.2, we set $\tau = \lceil 0.01N \rceil$.

The result of this experiment is shown in Figure 6.1. In this figure notice that all methods were terminated after 10^4 seconds. For practical purposes, for UCDC v.1 results are shown every 10^4 iterations. For all other methods results are shown after the first iteration takes place and then at every iteration. Observe in sub Figure 6.1a that block methods RCD v.1, RCD v.2 and UCDC v.2 performed fewer iterations compared to the single coordinate UCDC v.1. This is due to much larger per iteration



(a) Objective function $F(x)$ against iterations

(b) Objective function $F(x)$ against time

Fig. 6.1: Performance of all four methods RCD v.1 and v.2 and UCDC v.1 and v.2 on a sparse large scale ℓ_1-regularized least squares problem. For practical purposes, for UCDC v.1 results are printed every ten thousand iterations. *Calculation of $F(x)$ is not included in running time of the methods.* **Fig.1a** shows how the objective function $F(x)$ decreases as a function of the number of iterations. **Fig.1b** shows how the objective function $F(x)$ decreases as a function of wall-clock time measured in seconds.

computational complexity of the former methods compared to the latter. RCD v.2 despite its larger per iteration computational complexity among all methods it was the only one that solved the problem to higher accuracy within the required maximum time. Moreover, observe in sub Figure 6.1b that for purely practical purposes it might be better to have a combination of methods RCD v.1 and v.2. The former could be used at the beginning of the process while the latter could be used at later stages in order to guarantee robustness and speed closer to the optimal solution. *Finally, it is important to mention that on this problem for both RCD versions unit step sizes α were accepted by the backtracking line search for a major part of the process. Hence, backtracking line search was inexpensive.*

6.4. ℓ_1-Regularized Logistic Regression. In this section we present the performance of RCD and UCDC on the ℓ_1-regularized logistic regression problems (6.2).

Such problems are important in machine learning and are used for training a linear classifier $x \in \mathbb{R}^N$ that separates input data into two distinct clusters, for example, see [29] for further details.

We present the performance of the methods on two sparse large scale data sets. Problem details are given in Table 6.1, where $A \in \mathbb{R}^{m\times N}$ is a matrix whose rows are training samples.

Table 6.1: Properties of two ℓ_1-regularized logistic regression problems. The second and third columns show the number of training samples and features, respectively. The fourth column shows the sparsity of matrix A.

Problem	$\boldsymbol{m}$	$\boldsymbol{N}$	**nnz(A)/(mN)**
webspam	$350,000$	$16,609,143$	$2.24e$-4
kdd2010 (algebra)	$8,407,752$	$20,216,830$	$1.79e$-6

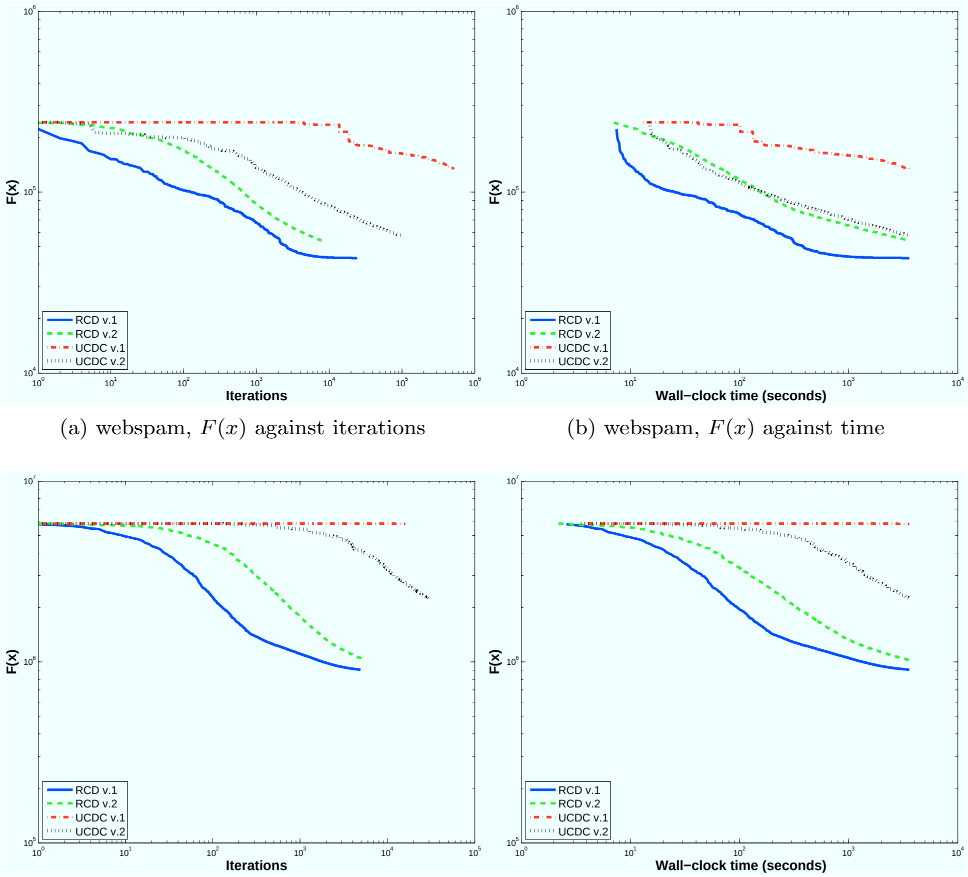
The data sets can be downloaded from the collection of LSVM problems in `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`. For both experiments we set $c = 10$, which resulted in more than 99% classification accuracy of the used data sets.

By [20, Table 10], the coordinate Lipschitz constants for UCDC are

$$L_j := \frac{1}{4}\sum_{q=1}^{m}(A_{qj}y_q)^2 \quad \forall j = 1, 2, \cdots, N,$$

where A_{qj} is the component of matrix A at qth row and jth column. For block versions RCD v.1, RCD v.2 and UCDC v.2, we set $\tau = \lceil 0.001N \rceil$.

The result of this experiment is shown in Figure 6.2. In this experiment all methods were terminated after one hour of running time. Notice that RCD versions were more efficient than both UCDC versions, with RCD v.1 being the fastest among all. An interesting observation in Figures 6.2a and 6.2c is that RCD versions had similar per iteration computational complexity since they performed similar number of iterations within the maximum allowed running time. However, for RCD v.1, it seems that diagonal information from the second order derivatives of f was enough to decrease faster the objective function for all iterations compared to RCD v.2. Finally, in this experiment we observed that both RCD versions accepted unit step sizes for a major part of the process.



(a) webspam, $F(x)$ against iterations

(b) webspam, $F(x)$ against time

(c) kdda, $F(x)$ against iterations

(d) kdda, $F(x)$ against time

Fig. 6.2: Performance of all four methods RCD v.1 and v.2 and UCDC v.1 and v.2 on two large scale ℓ_1-regularized logistic regression problems. The first and second rows of figures show the results for problems *webspam* and *kdda*, respectively. *Calculation of $F(x)$ is not included in running time of the methods.*

7. Conclusion. We presented a robust randomized block coordinate descent method for composite function problems (1.1), which we name *Robust Coordinate Descent* (RCD), that can properly handle second-order (curvature) information. The proposed method can vary from first- to second-order; depending on how large the block updates are set, how accurate second-order information are used and how inexactly the arising subproblems are solved. Although the per iteration computational complexity might be higher for RCD, we present synthetic and real world large scale examples where the number of iterations substantially decreases, as well as the overall time.

From the theoretical point of view, we prove global convergence of RCD and under standard assumptions we show that RCD exhibits on expectation *block* quadratic or superlinear rate of convergence.

REFERENCES

[1] P. Alart, O. Maisonneuve, and R. T. Rockafellar. *Nonsmooth Mechanics and Analysis: Theoretical and Numerical Advances.* Springer US, 2006.

[2] R. H. Byrd, J. Nocedal, and F. Oztoprak. An inexact successive quadratic approximation method for convex l-1 regularized optimization. Technical report, Northwestern University, September 2013. arXiv:1309.3529v1 [math.OC].

[3] E. Candès. Compressive sampling. In *International Congress of Mathematics*, volume 3, pages 1433–1452, Madrid, Spain, 2006.

[4] E. J. Candés, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inf. Theory*, 52(2):489–509, 2006.

[5] D. Donoho. Compressed sensing. *IEEE Trans. on Information Theory*, 52(4):1289–1306, April 2006.

[6] F. Facchinei, S. Sagratella, and G. Scutari. Parallel algorithms for big data optimization. Technical report, March 2014. arXiv:1402.5521v3 [cs.DC].

[7] O. Fercoq and P. Richtárik. Accelerated, parallel and proximal coordinate descent. Technical report, University of Edinburgh, December 2013. arXiv:1312.5799v2 [math.OC].

[8] K. Fountoulakis and J. Gondzio. A second-order method for strongly convex ℓ_1-regularization problems. Technical report, University of Edinburgh, April 2014. arXiv:1306.5386v4 [math.OC].

[9] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, 1985.

[10] S. Karimi and S. Vavasis. IMRO: a proximal quasi-Newton method for solving l_1-regularized least square problem. Technical report, University of Waterloo, January 2014. arXiv:1401.4220v1 [math.OC].

[11] J. D. Lee, Y. Sun, and M. A. Saunders. Proximal Newton-type methods for convex optimization. *Advances in Neural Information Processing Systems*, pages 836–844, 2012.

[12] J. D. Lee, Y. Sun, and M. A. Saunders. Proximal Newton-type methods for minimizing composite functions. Technical report, Stanford University, December 2013.

[13] Z. Lu and L. Xiao. On the complexity analysis of randomized block-coordinate descent methods. Technical report, Simon Fraser University, May 2013. arXiv:1305.4723v1 [math.OC].

[14] I. Necoara and A. Patrascu. A random coordinate descent algorithm for optimization problems with composite objective function and linear coupled constraints. *Computational Optimization and Applications*, 57(2):307–337, 2014.

[15] Yu. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course.* Applied Optimization. Kluwer Academic Publishers, 2004.

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

[17] Yu. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[18] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer Series in Operations Research. Springer-Verlag, New York, 1999.

[19] Z. Qin, K. Scheinberg, and D. Goldfarb. Efficient block-coordinate descent algorithms for the group lasso. *Mathematical Programming Computation*, 5(2):143–169, 2013.

[20] P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 2012.

[21] P. Richtárik and M. Takáč. Parallel coordinate descent methods for big data optimization. Technical report, University of Edinburgh, December 2013. arXiv:1212.0873v2 [math.OC].

[22] M. De Santis, S. Lucidi, and F. Rinaldi. A fast active set block coordinate descent algorithm for ℓ_1-regularized least squares. Technical report, March 2014. arXiv:1403.1738v2 [math.OC].

[23] K. Scheinberg and X. Tang. Practical inexact proximal quasi-Newton method with global complexity analysis. Technical report, Lehigh University, November 2013. arXiv:1311.6547v3 [cs.LG].

[24] S. Shalev-Schwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14:567–599, 2013.

[25] N. Simon and R. Tibshirani. Standardization and the group lasso penalty. *Statistica Sinica*, 22(3):983, 2012.

[26] R. Tappenden, P. Richtárik, and J. Gondzio. Inexact coordinate descent: Complexity and preconditioning. Technical report, University of Edinburgh, April 2013. arXiv:1304.5530v1 [math.OC].

[27] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Roy. Statist. Soc.*, 58(1):267–288, 1996.

[28] S. J. Wright. Accelerated block-coordinate relaxation for regularized optimization. *SIAM Journal of Optimization*, 22(1):159–186, 2012.

[29] G. X. Yuan, C. H. Ho, and C. J. Lin. Recent advances of large-scale linear classification. *Proceedings of the IEEE*, 100(9):2584–2603, 2012.