Comput. Optim. Appl. manuscript No.
(will be inserted by the editor)

A random coordinate descent algorithm for optimization problems with composite objective function and linear coupled constraints

Ion Necoara · Andrei Patrascu

Received: 25 March 2012 / Accepted: date

Abstract In this paper we propose a variant of the random coordinate descent method for solving linearly constrained convex optimization problems with composite objective functions. If the smooth part of the objective function has Lipschitz continuous gradient, then we prove that our method obtains an ϵ-optimal solution in $\mathcal{O}(N^2/\epsilon)$ iterations, where N is the number of blocks. For the class of problems with cheap coordinate derivatives we show that the new method is faster than methods based on full-gradient information. Analysis for the rate of convergence in probability is also provided. For strongly convex functions our method converges linearly. Extensive numerical tests confirm that on very large problems, our method is much more numerically efficient than methods based on full gradient information.

Keywords Coordinate descent · composite objective function · linearly coupled constraints · randomized algorithms · convergence rate $\mathcal{O}(1/\epsilon)$.

1 Introduction

The basic problem of interest in this paper is the following convex minimization problem with composite objective function:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} F(x) \quad (:= f(x) + h(x)) \\ &\text{s.t.:}\ a^T x = 0, \end{aligned} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth convex function defined by a black-box oracle, $h : \mathbb{R}^n \to \mathbb{R}$ is a general closed convex function and $a \in \mathbb{R}^n$. Further, we assume that function h is coordinatewise separable and simple (by simple we mean that

The research leading to these results has received funding from: the European Union (FP7/2007–2013) under grant agreement no 248940; CNCS (project TE-231, 19/11.08.2010); ANCS (project PN II, 80EU/2010); POSDRU/89/1.5/S/62557.

I. Necoara and A. Patrascu are with the Automation and Systems Engineering Department, University Politehnica Bucharest, 060042 Bucharest, Romania, Tel.: +40-21-4029195,
Fax: +40-21-4029195; E-mail: `ion.necoara@acse.pub.ro, andrei.patrascu@acse.pub.ro`

we can find a closed-form solution for the minimization of h with some simple auxiliary function). Special cases of this model include linearly constrained smooth optimization (where $h \equiv 0$) which was analyzed in [16,30], support vector machines (where h is the indicator function of some box constrained set) [10,14] and composite optimization (where $a = 0$) [23,27–29].

Linearly constrained optimization problems with composite objective function arise in many applications such as compressive sensing [5], image processing [6], truss topology design [17], distributed control [15], support vector machines [28], traffic equilibrium and network flow problems [3] and many other areas. For problems of moderate size there exist many iterative algorithms such as Newton, quasi-Newton or projected gradient methods [8,9,13]. However, the problems that we consider in this paper have the following features: *the dimension of the optimization variables* is very large such that usual methods based on full gradient computations are prohibitive. Moreover, *the incomplete structure of information* that may appear when the data are distributed in space and time, or when there exists lack of physical memory and enormous complexity of the gradient update can also be an obstacle for full gradient computations. In this case, it appears that a reasonable approach to solving problem (1) is to use *(block) coordinate descent methods.* These methods were among the first optimization methods studied in literature [4]. The main differences between all variants of coordinate descent methods consist of the criterion of choosing at each iteration the coordinate over which we minimize our objective function and the complexity of this choice. Two classical criteria, used often in these algorithms, are the cyclic and the greedy (e.g., Gauss-Southwell) coordinate search, which significantly differ by the amount of computations required to choose the appropriate index. The rate of convergence of cyclic coordinate search methods has been determined recently in [1,26]. Also, for coordinate descent methods based on the Gauss-Southwell rule, the convergence rate is given in [27–29]. Another interesting approach is based on random coordinate descent, where the coordinate search is random. Recent complexity results on random coordinate descent methods were obtained by Nesterov in [20] for smooth convex functions. The extension to composite objective functions was given in [23] and for the grouped Lasso problem in [22]. However, all these papers studied optimization models where the constraint set is decoupled (i.e., characterized by Cartesian product). The rate analysis of a random coordinate descent method for linearly coupled constrained optimization problems with smooth objective function was developed in [16].

In this paper we present a random coordinate descent method suited for large scale problems with composite objective function. Moreover, in our paper we focus on linearly coupled constrained optimization problems (i.e., the constraint set is coupled through linear equalities). Note that the model considered in this paper is more general than the one from [16], since we allow composite objective functions. We prove for our method an expected convergence rate of order $\mathcal{O}(\frac{N^2}{k})$, where N is number of blocks and k is the iteration counter. We show that for functions with cheap coordinate derivatives the new method is much faster, either in worst case complexity analysis, or numerical implementation, than schemes based on full gradient information (e.g., coordinate gradient descent method developed in [29]). But our method also offers other important advantages, e.g., due to the randomization, our algorithm is easier to analyze and implement, it leads to more robust output and is adequate for modern computational architectures (e.g, parallel or distributed architectures). Analysis for rate of convergence in probability is also provided. For strongly convex functions we prove that the new method converges linearly. We also provide extensive numerical simulations and compare our algorithm against state-of-the-art methods from the literature on three large-scale applications: support vector machine, the Chebyshev center of a set of points and random generated optimization problems with an ℓ_1-regularization term.

The paper is organized as follows. In order to present our main results, we introduce some notations and assumptions for problem (1) in Section 1.1. In Section 2 we present the new random coordinate descent (RCD) algorithm. The main results of the paper can be found in Section 3, where we derive the rate of convergence in expectation, probability and for the strongly convex case. In Section 4 we generalize the algorithm and extend the previous results to a more general model. We also analyze its complexity and compare it with other methods from the literature, in particular the coordinate descent method of Tseng [29] in Section 5. Finally, we test the practical efficiency of our algorithm through extensive numerical experiments in Section 6.

1.1 Preliminaries

We work in the space $\mathbb{R}^n$ composed of column vectors. For $x, y \in \mathbb{R}^n$ we denote:

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i.$$

We use the same notation $\langle \cdot, \cdot \rangle$ for spaces of different dimensions. If we fix a norm $\|\cdot\|$ in $\mathbb{R}^n$, then its dual norm is defined by:

$$\|y\|^* = \max_{\|x\|=1} \langle y, x \rangle.$$

We assume that the entire space dimension is decomposable into N blocks:

$$n = \sum_{i=1}^{N} n_i.$$

We denote by U_i the blocks of the identity matrix:

$$I_n = [U_1 \ \dots \ U_N],$$

where $U_i \in \mathbb{R}^{n \times n_i}$. For some vector $x \in \mathbb{R}^n$, we use the notation x_i for the ith block in x, $\nabla_i f(x) = U_i^T \nabla f(x)$ is the ith block in the gradient of the function f at x, and $\nabla_{ij} f(x) = \begin{bmatrix} \nabla_i f(x) \\ \nabla_j f(x) \end{bmatrix}$. We denote by $supp(x)$ the number of nonzero coordinates in x. Given a matrix $A \in \mathbb{R}^{m \times n}$, we denote its nullspace by $Null(A)$. In the rest of the paper we consider local Euclidean norms in all spaces $\mathbb{R}^{n_i}$, i.e., $\|x_i\| = \sqrt{(x_i)^T x_i}$ for all $x_i \in \mathbb{R}^{n_i}$ and $i = 1, \dots, N$.
For model (1) we make the following assumptions:

Assumption 1 *The smooth and nonsmooth parts of the objective function in optimization model* (1) *satisfy the following properties:*

(i) Function f is convex and has block-coordinate Lipschitz continuous gradient:

$$\|\nabla_i f(x + U_i h_i) - \nabla_i f(x)\| \le L_i \|h_i\| \quad \forall x \in \mathbb{R}^n, \;\; h_i \in \mathbb{R}^{n_i}, \;\; i = 1, \dots, N.$$

(ii) The nonsmooth function h is convex and coordinatewise separable.

Assumption 1 (i) is typical for composite optimization, see e.g., [19,29]. Assumption 1 (ii) covers many applications as we further exemplify. A special case of coordinatewise separable function that has attracted a lot of attention in the area of signal processing and data mining is the ℓ_1-regularization [22]:

$$h(x) = \lambda \|x\|_1, \tag{2}$$

where $\lambda > 0$. Often, a large λ factor induces sparsity in the solution of optimization problem (1). Note that the function h in (2) belongs to the general class of coordinatewise separable piecewise linear/quadratic functions with $\mathcal{O}(1)$ pieces. Another special case is the box indicator function, i.e.:

$$h(x) = \mathbf{1}_{[l,u]} = \begin{cases} 0, & l \le x \le u \\ \infty, & \text{otherwise.} \end{cases} \tag{3}$$

Adding box constraints to a quadratic objective function f in (1) leads e.g., to support vector machine (SVM) problems [7,28]. The reader can easily find many other examples of function h satisfying Assumption 1 (ii).
Based on Assumption 1 (i), the following inequality can be derived [18]:

$$f(x + U_i h_i) \le f(x) + \langle \nabla_i f(x), h_i \rangle + \frac{L_i}{2} \|h_i\|^2 \quad \forall x \in \mathbb{R}^n, \;\; h_i \in \mathbb{R}^{n_i}. \tag{4}$$

In the sequel, we use the notation:

$$L = \max_{1 \le i \le N} L_i.$$

For $\alpha \in [0, 1]$ we introduce the extended norm on $\mathbb{R}^n$ similar as in [20]:

$$\|x\|_\alpha = \left(\sum_{i=1}^N L_i^\alpha \|x_i\|^2 \right)^{\frac{1}{2}}$$

and its dual norm

$$\|y\|_\alpha^* = \left(\sum_{i=1}^N \frac{1}{L_i^\alpha} \|y_i\|^2 \right)^{\frac{1}{2}}.$$

Note that these norms satisfy the Cauchy-Schwartz inequality:

$$\|x\|_\alpha \|y\|_\alpha^* \ge \langle x, y \rangle \;\; \forall x, y \in \mathbb{R}^n.$$

For a simpler exposition we use a context-dependent notation as follows: let $x \in \mathbb{R}^n$ such that $x = \sum_{i=1}^N U_i x_i$, then $x_{ij} \in \mathbb{R}^{n_i + n_j}$ denotes a two component vector $x_{ij} = \begin{bmatrix} x_i \\ x_j \end{bmatrix}$. Moreover, by addition with a vector in the extended space $y \in \mathbb{R}^n$, i.e., $y + x_{ij}$, we understand $y + U_i x_i + U_j x_j$. Also, by inner product $\langle y, x_{ij} \rangle$ with vectors y from the extended space $\mathbb{R}^n$ we understand $\langle y, x_{ij} \rangle = \langle y_i, x_i \rangle + \langle y_j, x_j \rangle$.
Based on Assumption 1 (ii) we can derive from (4) the following result:

Lemma 1 *Let function f be convex and satisfy Assumption 1. Then, the function f has componentwise Lipschitz continuous gradient w.r.t. every pair (i,j), i.e.:*

$$\left\| \begin{bmatrix} \nabla_i f(x+U_i s_i+U_j s_j) \\ \nabla_j f(x+U_i s_i+U_j s_j) \end{bmatrix} - \begin{bmatrix} \nabla_i f(x) \\ \nabla_j f(x) \end{bmatrix} \right\|_\alpha^* \leq L_{ij}^\alpha \left\| \begin{bmatrix} s_i \\ s_j \end{bmatrix} \right\|_\alpha \quad \forall x \in \mathbb{R}^n,\ s_i \in \mathbb{R}^{n_i},\ s_j \in \mathbb{R}^{n_j},$$

where we define $L_{ij}^\alpha = L_i^{1-\alpha} + L_j^{1-\alpha}$.

Proof Let $f^* = \min_{x \in \mathbb{R}^n} f(x)$. Based on (4) we have for any pair (i,j):

$$\begin{aligned} f(x) - f^* &\geq \max_{l \in \{1,\dots N\}} \frac{1}{2L_l} \|\nabla_l f(x)\|^2 \geq \max_{l \in \{i,j\}} \frac{1}{2L_l} \|\nabla_l f(x)\|^2 \\ &\geq \frac{1}{2\left(L_i^{1-\alpha} + L_j^{1-\alpha}\right)} \left(\frac{1}{L_i^\alpha} \|\nabla_i f(x)\|^2 + \frac{1}{L_j^\alpha} \|\nabla_j f(x)\|^2 \right) \\ &= \frac{1}{2L_{ij}^\alpha} \|\nabla_{ij} f(x)\|_\alpha^{*2}, \end{aligned}$$

where in the third inequality we used that $\alpha a + (1-\alpha) b \leq \max\{a,b\}$ for all $\alpha \in [0,1]$. Now, note that the function $g_1(y_{ij}) = f(x + y_{ij} - x_{ij}) - f(x) - \langle \nabla f(x), y_{ij} - x_{ij} \rangle$ satisfies the Assumption 1 (i). If we apply the above inequality to $g_1(y_{ij})$ we get the following relation:

$$f(x+y_{ij}-x_{ij}) \geq f(x) + \langle \nabla f(x), y_{ij} - x_{ij} \rangle + \frac{1}{2L_{ij}^\alpha} \|\nabla_{ij} f(x + y_{ij} - x_{ij}) - \nabla_{ij} f(x)\|_\alpha^{*2}.$$

On the other hand, applying the same inequality to $g_2(x_{ij}) = f(x) - f(x + y_{ij} - x_{ij}) + \langle \nabla f(x + y_{ij} - x_{ij}), y_{ij} - x_{ij} \rangle$, which also satisfies Assumption 1 (i), we have:

$$\begin{aligned} f(x) \geq f(x + y_{ij} - x_{ij}) &+ \langle \nabla f(x+y_{ij} - x_{ij}), y_{ij} - x_{ij} \rangle + \\ &\frac{1}{2L_{ij}^\alpha} \|\nabla_{ij} f(x + y_{ij} - x_{ij}) - \nabla_{ij} f(x)\|_\alpha^{*2}. \end{aligned}$$

Further, denoting $s_{ij} = y_{ij} - x_{ij}$ and adding up the resulting inequalities we get:

$$\begin{aligned} \frac{1}{L_{ij}^\alpha} \|\nabla_{ij} f(x + s_{ij}) - \nabla_{ij} f(x)\|_\alpha^{*2} &\leq \langle \nabla_{ij} f(x + s_{ij}) - \nabla_{ij} f(x), s_{ij} \rangle \\ &\leq \|\nabla_{ij} f(x + s_{ij}) - \nabla_{ij} f(x)\|_\alpha^* \|s_{ij}\|_\alpha, \end{aligned}$$

for all $x \in \mathbb{R}^n$ and $s_{ij} \in \mathbb{R}^{n_i+n_j}$, which proves the statement of this lemma. □

It is straightforward to see that we can obtain from Lemma 1 the following inequality (see also [18]):

$$f(x + s_{ij}) \leq f(x) + \langle \nabla_{ij} f(x), s_{ij} \rangle + \frac{L_{ij}^\alpha}{2} \|s_{ij}\|_\alpha^2, \tag{5}$$

for all $x \in \mathbb{R}^n, s_{ij} \in \mathbb{R}^{n_i+n_j}$ and $\alpha \in [0,\ 1]$.

2 Random coordinate descent algorithm

In this section we introduce a variant of Random Coordinate Descent (RCD) method for solving problem (1) that performs a minimization step with respect to two block variables at each iteration. The coupling constraint (that is, the weighted sum constraint $a^T x = 0$) prevents the development of an algorithm that performs a minimization with respect to only one variable at each iteration. We will therefore be interested in the restriction of the objective function f on feasible directions consisting of at least two nonzero (block) components.

Let (i, j) be a two dimensional random variable, where $i, j \in \{1, \dots, N\}$ with $i \neq j$ and $p_{i_k j_k} = \Pr((i,j) = (i_k, j_k))$ be its probability distribution. Given a feasible x, two blocks are chosen randomly with respect to a probability distribution p_{ij} and a quadratic model derived from the composite objective function is minimized with respect to these coordinates. Our method has the following iteration: given a feasible initial point x^0, that is $a^T x^0 = 0$, then for all $k \geq 0$

```
Algorithm 1 (RCD)
1. Choose randomly two coordinates $(i_k, j_k)$ with probability $p_{i_k j_k}$
2. Set $x^{k+1} = x^k + U_{i_k} d_{i_k} + U_{j_k} d_{j_k}$,
```

where the directions d_{i_k} and d_{j_k} are chosen as follows: if we use for simplicity the notation (i,j) instead of (i_k, j_k), the direction $d_{ij} = [d_i^T \; d_j^T]^T$ is given by

$$\begin{aligned} d_{ij} = \arg \min_{s_{ij} \in \mathbb{R}^{n_i + n_j}} & f(x^k) + \langle \nabla_{ij} f(x^k), s_{ij} \rangle + \frac{L_{ij}^{\alpha}}{2} \left\| s_{ij} \right\|_{\alpha}^2 + h(x^k + s_{ij}) \\ \text{s.t.:} \quad & a_i^T s_i + a_j^T s_j = 0, \end{aligned} \tag{6}$$

where $a_i \in \mathbb{R}^{n_i}$ and $a_j \in \mathbb{R}^{n_j}$ are the ith and jth blocks of vector a, respectively. Clearly, in our algorithm we maintain feasibility at each iteration, i.e. $a^T x^k = 0$ for all $k \geq 0$.

Remark 1

(i) Note that for the scalar case (i.e., $N = n$) and h given by (2) or (3), the direction d_{ij} in (6) can be computed in closed form. For the block case (i.e., $n_i > 1$ for all i) and if h is a coordinatewise separable, strictly convex and piece-wise linear/quadratic function with $\mathcal{O}(1)$ pieces (e.g., h given by (2)), there are algorithms for solving the above subproblem in linear-time (i.e., $\mathcal{O}(n_i + n_j)$ operations) [29]. Also for h given by (3), there exist in the literature algorithms for solving the subproblem (6) with overall complexity $\mathcal{O}(n_i + n_j)$ [2,12].

(ii) In algorithm (RCD) we consider $(i,j) = (j,i)$ and $i \neq j$. Moreover, we know that the complexity of choosing randomly a pair (i,j) with a uniform probability distribution requires $\mathcal{O}(1)$ operations. □

We assume that random variables $(i_k, j_k)_{k \geq 0}$ are i.i.d. In the sequel, we use notation η^k for the entire history of random pair choices and ϕ^k for the expected value of the objective function w.r.t. η^k, i.e.:

$$\eta^k = \{(i_0, j_0), \dots, (i_{k-1}, j_{k-1})\} \text{ and } \phi^k = E_{\eta^k} F(x^k).$$

2.1 Previous work

We briefly review some well-known methods from the literature for solving the optimization model (1). In [27–29] Tseng studied optimization problems in the form (1) and developed a (block) coordinate gradient descent(CGD) method based on the Gauss-Southwell choice rule. The main requirement for the (CGD) iteration is the solution of the following problem: given a feasible x and a working set of indexes $\mathcal{J}$, the update direction is defined by

$$\begin{aligned} d_H(x;\mathcal{J}) = \arg\min_{s\in\mathbb{R}^n} & f(x) + \langle \nabla f(x), s\rangle + \frac{1}{2}\langle Hs, s\rangle + h(x+s) \\ \text{s.t.: } & a^T s = 0, \quad s_j = 0 \quad \forall j \notin \mathcal{J}, \end{aligned} \tag{7}$$

where $H \in \mathbb{R}^{n\times n}$ is a symmetric matrix chosen at the initial step of the algorithm.

Algorithm (CGD):

```
1. Choose a nonempty set of indices $\mathcal{J}^k \subset \{1,\dots,n\}$ with respect to the
   Gauss-Southwell rule
2. Solve (7) with $x = x^k$, $\mathcal{J} = \mathcal{J}^k$, $H = H_k$ to obtain $d^k = d_{H_k}(x^k;\mathcal{J}^k)$
3. Choose stepsize $\alpha^k > 0$ and set $x^{k+1} = x^k + \alpha^k d^k$.
```

In [29], the authors proved for the particular case when function h is piece-wise linear/quadratic with $\mathcal{O}(1)$ pieces that an ϵ-optimal solution is attained in $\mathcal{O}(\frac{nLR_0^2}{\epsilon})$ iterations, where R_0 denotes the Euclidean distance from the initial point to some optimal solution. Also, in [29] the authors derive estimates of order $\mathcal{O}(n)$ on the computational complexity of each iteration for this choice of h.

Furthermore, for a quadratic function f and a box indicator function h (e.g., support vector machine (SVM) applications) one of the first decomposition approaches developed similar to (RCD) is Sequential Minimal Optimization (SMO) [21]. SMO consists of choosing at each iteration two scalar coordinates with respect to some heuristic rule based on KKT conditions and solving the small QP subproblem obtained through the decomposition process. However, the rate of convergence is not provided for the SMO algorithm. But the numerical experiments show that the method is very efficient in practice due to the closed form solution of the QP subproblem. List and Simon [14] proposed a variant of block coordinate descent method for which an arithmetical complexity $\mathcal{O}(\frac{n^2LR_0^2}{\epsilon})$ is proved on a quadratic model with a box indicator function and generalized linear constraints. Later, Hush et al. [10] presented a more practical decomposition method which attains the same complexity as the previous methods.

A random coordinate descent algorithm for model (1) with $a = 0$ and h being the indicator function for a Cartesian product of sets was analyzed by Nesterov in [20]. The generalization of this algorithm to composite objective functions has been studied in [22, 23]. However, none of these papers studied the application of coordinate descent algorithms to linearly coupled constrained optimization models. A similar random coordinate descent algorithm as the (RCD) method described in the present paper, for optimization problems with smooth objective and linearly coupled constraints, has been developed and analyzed by Necoara et al. in [16].

We further extend these results to linearly constrained composite objective function optimization and provide in the sequel the convergence rate analysis for the previously presented variant of the (RCD) method (see Algorithm 1 (RCD)).

3 Convergence results

In the following subsections we derive the convergence rate of Algorithm 1 (RCD) for composite optimization model (1) in expectation, probability and for strongly convex functions.

3.1 Convergence in expectation

In this section we study the rate of convergence in expectation of algorithm (RCD). We consider uniform probability distribution, i.e., the event of choosing a pair (i, j) can occur with probability:

$$p_{ij} = \frac{2}{N(N-1)},$$

since we assume that $(i, j) = (j, i)$ and $i \neq j \in \{1, \ldots, N\}$ (see Remark 1 (ii)). In order to provide the convergence rate of our algorithm, first we have to define the *conformal realization* of a vector introduced in [24,25].

Definition 1 Let $d, d' \in \mathbb{R}^n$, then the vector d' is *conformal* to d if:

$$\textit{supp}(d') \subseteq \textit{supp}(d) \quad \text{and} \quad d'_j d_j \geq 0 \;\; \forall j = 1, \ldots, n.$$

For a given matrix $A \in \mathbb{R}^{m \times n}$, with $m \leq n$, we introduce the notion of elementary vectors defined as:

Definition 2 An elementary vector of $\textit{Null}(A)$ is a vector $d \in \textit{Null}(A)$ for which there is no nonzero vector $d' \in \textit{Null}(A)$ conformal to d and $\textit{supp}(d') \neq \textit{supp}(d)$.

Based on Exercise 10.6 in [25] we state the following lemma:

Lemma 2 *[25] Given $d \in \textit{Null}(A)$, if d is an elementary vector, then $|\textit{supp}(d)| \leq \textit{rank}(A) + 1 \leq m + 1$. Otherwise, d has a conformal realization:*

$$d = d^1 + \cdots + d^s,$$

where $s \geq 1$ and $d^t \in \textit{Null}(A)$ are elementary vectors conformal to d for all $t = 1, \ldots, s$.

For the scalar case, i.e., $N = n$ and $m = 1$, the method provided in [29] finds a conformal realization with dimension $s \leq |\text{supp}(d)| - 1$ within $\mathcal{O}(n)$ operations. We observe that elementary vectors d^t in Lemma 2 for the case $m = 1$ (i.e., $A = a^T$) have at most 2 nonzero components.

Our convergence analysis is based on the following lemma, whose proof can be found in [29, Lemma 6.1]:

Lemma 3 *[29] Let h be coordinatewise separable and convex. For any $x, x+d \in \textit{dom}\, h$, let d be expressed as $d = d^1 + \cdots + d^s$ for some $s \geq 1$ and some nonzero $d^t \in \mathbb{R}^n$ conformal to d for $t = 1, \ldots, s$. Then, we have:*

$$h(x+d) - h(x) \geq \sum_{t=1}^{s} \Big(h(x + d^t) - h(x) \Big).$$

For the simplicity of the analysis we introduce the following linear subspaces:

$$S_{ij} = \left\{ d \in \mathbb{R}^{n_i+n_j} : \ a_{ij}^T d = 0 \right\} \quad \text{and} \quad S = \left\{ d \in \mathbb{R}^n : \ a^T d = 0 \right\}.$$

A simplified update rule of algorithm (RCD) is expressed as:

$$x^+ = x + U_i d_i + U_j d_j.$$

We denote by F^* and X^* the optimal value and the optimal solution set for problem (1), respectively. We also introduce the maximal residual defined in terms of the norm $\|\cdot\|_\alpha$:

$$R_\alpha = \max_x \left\{ \max_{x^* \in X^*} \left\| x - x^* \right\|_\alpha : \ F(x) \leq F(x^0) \right\},$$

which measures the size of the level set of F given by x^0. We assume that this distance is finite for the initial iterate x^0.
Now, we prove the main result of this section:

Theorem 1 *Let F satisfy Assumption 1. Then, the random coordinate descent algorithm (RCD) based on the uniform distribution generates a sequence x^k satisfying the following convergence rate for the expected values of the objective function:*

$$\phi^k - F^* \leq \frac{N^2 L^{1-\alpha} R_\alpha^2}{k + \frac{N^2 L^{1-\alpha} R_\alpha^2}{F(x^0) - F^*}}.$$

Proof For simplicity, we drop the index k and use instead of (i_k, j_k) and x^k the notation (i,j) and x, respectively. Based on (5) we derive:

$$\begin{aligned} F(x^+) &\leq f(x) + \langle \nabla_{ij} f(x), d_{ij} \rangle + \frac{L_{ij}^\alpha}{2} \left\| d_{ij} \right\|_\alpha^2 + h(x + d_{ij}) \\ &\overset{(6)}{=} \min_{s_{ij} \in S_{ij}} f(x) + \langle \nabla_{ij} f(x), s_{ij} \rangle + \frac{L_{ij}^\alpha}{2} \left\| s_{ij} \right\|_\alpha^2 + h(x + s_{ij}). \end{aligned}$$

Taking expectation in both sides w.r.t. random variable (i,j) and recalling that $p_{ij} = \frac{2}{N(N-1)}$, we get:

$$\begin{aligned} E_{ij}\left[F(x^+)\right] &\leq E_{ij}\left[\min_{s_{ij} \in S_{ij}} f(x) + \langle \nabla_{ij} f(x), s_{ij} \rangle + \frac{L_{ij}^\alpha}{2} \left\| s_{ij} \right\|_\alpha^2 + h(x + s_{ij}) \right] \\ &\leq E_{ij}\left[f(x) + \langle \nabla_{ij} f(x), s_{ij} \rangle + \frac{L_{ij}^\alpha}{2} \left\| s_{ij} \right\|_\alpha^2 + h(x + s_{ij}) \right] \\ &= \frac{2}{N(N-1)} \sum_{i,j} \left(f(x) + \langle \nabla_{ij} f(x), s_{ij} \rangle + \frac{L_{ij}^\alpha}{2} \left\| s_{ij} \right\|_\alpha^2 + h(x + s_{ij}) \right) \qquad (8) \\ &= f(x) + \frac{2}{N(N-1)} \left(\langle \nabla f(x), \sum_{i,j} s_{ij} \rangle + \sum_{i,j} \frac{L_{ij}^\alpha}{2} \left\| s_{ij} \right\|_\alpha^2 + \sum_{i,j} h(x + s_{ij}) \right), \end{aligned}$$

for all possible $s_{ij} \in S_{ij}$ and pairs (i,j) with $i \neq j \in \{1, \ldots, N\}$.
Based on Lemma 2 for $m = 1$, it follows that any $d \in S$ has a conformal realization defined by $d = \sum_{t=1}^{s} d^t$, where the vectors $d^t \in S$ are conformal to d and have only

two nonzero components. Thus, for any $t = 1, \dots, s$ there is a pair (i,j) such that $d^t \in S_{ij}$. Therefore, for any $d \in S$ we can choose an appropriate set of pairs (i,j) and vectors $s^d_{ij} \in S_{ij}$ conformal to d such that $d = \sum\limits_{i,j} s^d_{ij}$. As we have seen, the above chain of relations in (8) holds for any set of pairs (i,j) and vectors $s_{ij} \in S_{ij}$. Therefore, it also holds for the set of pairs (i,j) and vectors s^d_{ij} such that $d = \sum\limits_{i,j} s^d_{ij}$. In conclusion, we have from (8) that:

$$E_{ij}\left[F(x^+)\right] \leq f(x) + \frac{2}{N(N-1)} \left(\langle \nabla f(x), \sum_{i,j} s^d_{ij} \rangle + \sum_{i,j} \frac{L^\alpha_{ij}}{2} \left\| s^d_{ij} \right\|^2_\alpha + \sum_{i,j} h(x + s^d_{ij}) \right),$$

for all $d \in S$. Moreover, observing that $L^\alpha_{ij} \leq 2L^{1-\alpha}$ and applying Lemma 3 in the previous inequality for coordinatewise separable functions $\|\cdot\|^2_\alpha$ and $h(\cdot)$, we obtain:

$$\begin{aligned} E_{ij}\left[F(x^+)\right] &\leq f(x) + \frac{2}{N(N-1)}(\langle \nabla f(x), \sum_{i,j} s^d_{ij} \rangle + \sum_{i,j} \frac{L^\alpha_{ij}}{2} \|s^d_{ij}\|^2_\alpha + \sum_{i,j} h(x + s^d_{ij})) \\ &\overset{\text{Lemma 3}}{\leq} f(x) + \frac{2}{N(N-1)}(\langle \nabla f(x), \sum_{i,j} s^d_{ij} \rangle + L^{1-\alpha} \| \sum_{i,j} s^d_{ij} \|^2_\alpha + \\ &\qquad h(x + \sum_{i,j} s^d_{ij}) + (\frac{N(N-1)}{2} - 1) h(x)) \qquad (9) \\ &\overset{d = \sum_{i,j} s^d_{ij}}{=} (1 - \frac{2}{N(N-1)}) F(x) + \frac{2}{N(N-1)} (f(x) + \langle \nabla f(x), d \rangle + \\ &\qquad L^{1-\alpha} \|d\|^2_\alpha + h(x + d)), \end{aligned}$$

for any $d \in S$. Note that (9) holds for every $d \in S$ since (8) holds for any $s_{ij} \in S_{ij}$. Therefore, as (9) holds for every vector from the subspace S, it also holds for the following particular vector $\tilde{d} \in S$ defined as:

$$\tilde{d} = \arg\min_{s \in S} f(x) + \langle \nabla f(x), s \rangle + L^{1-\alpha} \|s\|^2_\alpha + h(x + s).$$

Based on this choice and using similar reasoning as in [19,23] for proving the convergence rate of gradient type methods for composite objective functions, we derive the following:

$$\begin{aligned} & f(x) + \langle \nabla f(x), \tilde{d} \rangle + L^{1-\alpha} \left\| \tilde{d} \right\|^2_\alpha + h(x + \tilde{d}) \\ & = \min_{y \in S} f(x) + \langle \nabla f(x), y - x \rangle + L^{1-\alpha} \|y - x\|^2_\alpha + h(y) \\ & \leq \min_{y \in S} F(y) + L^{1-\alpha} \|y - x\|^2_\alpha \\ & \leq \min_{\beta \in [0,1]} F(\beta x^* + (1-\beta)x) + \beta^2 L^{1-\alpha} \left\| x - x^* \right\|^2_\alpha \\ & \leq \min_{\beta \in [0,1]} F(x) - \beta(F(x) - F^*) + \beta^2 L^{1-\alpha} R^2_\alpha \\ & = F(x) - \frac{(F(x) - F^*)^2}{L^{1-\alpha} R^2_\alpha}, \end{aligned}$$

where in the first inequality we used the convexity of f while in the second and third inequalities we used basic optimization arguments. Therefore, at each iteration k the following inequality holds:

$$\begin{aligned} E_{i_k j_k}\left[F(x^{k+1})\right] \leq & (1-\frac{2}{N(N-1)})F(x^k)+ \\ & \frac{2}{N(N-1)}\left[F(x^k)-\frac{(F(x^k)-F^*)^2}{L^{1-\alpha}R_\alpha^2}\right]. \end{aligned}$$

Taking expectation with respect to η_k and using convexity properties we get:

$$\begin{aligned} \phi^{k+1}-F^* \leq & (1-\frac{2}{N(N-1)})(\phi^k-F^*)+ \\ & \frac{2}{N(N-1)}\left[(\phi^k-F^*)-\frac{(\phi^k-F^*)^2}{L^{1-\alpha}R_\alpha^2}\right] \\ \leq & (\phi^k-F^*)-\frac{2}{N(N-1)}\left[\frac{(\phi^k-F^*)^2}{L^{1-\alpha}R_\alpha^2}\right]. \end{aligned} \tag{10}$$

Further, if we denote $\Delta^k=\phi^k-F^*$ and $\gamma=N(N-1)L^{1-\alpha}R_\alpha^2$ we get:

$$\Delta^{k+1} \leq \Delta^k - \frac{(\Delta^k)^2}{\gamma}.$$

Dividing both sides with $\Delta^k\Delta^{k+1}>0$ and using the fact that $\Delta^{k+1}\leq\Delta^k$ we get:

$$\frac{1}{\Delta^{k+1}} \geq \frac{1}{\Delta^k}+\frac{1}{\gamma} \quad \forall k \geq 0.$$

Finally, summing up from $0,\dots,k$ we easily get the above convergence rate. □

Let us analyze the convergence rate of our method for the two most common cases of the extended norm introduced in this section: w.r.t. extended Euclidean norm $\|\cdot\|_0$ (i.e., $\alpha=0$) and norm $\|\cdot\|_1$ (i.e., $\alpha=1$). Recall that the norm $\|\cdot\|_1$ is defined by:

$$\|x\|_1^2=\sum_{i=1}^N L_i\|x_i\|^2.$$

Corrollary 1 *Under the same assumptions of Theorem 1, the algorithm (RCD) generates a sequence x^k such that the expected values of the objective function satisfy the following convergence rates for $\alpha=0$ and $\alpha=1$:*

$$\begin{aligned} \alpha=0: \quad & \phi^k-F^* \leq \frac{N^2LR_0^2}{k+\frac{N^2LR_0^2}{F(x^0)-F^*}}, \\ \alpha=1: \quad & \phi^k-F^* \leq \frac{N^2R_1^2}{k+\frac{N^2R_1^2}{F(x^0)-F^*}}. \end{aligned}$$

Remark 2 We usually have $R_1^2 \leq LR_0^2$ and this shows the advantages that the general norm $\|\cdot\|_\alpha$ has over the Euclidean norm. Indeed, if we denote by $r_i^2 = \max_x\{\max_{x^* \in X^*} \|x_i - x_i^*\|^2 : F(x) \leq F(x^0)\}$, then we can provide upper bounds on $R_1^2 \leq \sum_{i=1}^N L_i r_i^2$ and $R_0^2 \leq \sum_{i=1}^n r_i^2$. Clearly, the following inequality is valid:

$$\sum_{i=1}^N L_i r_i^2 \leq \sum_{i=1}^N L r_i^2,$$

and the inequality holds with equality for $L_i = L$ for all $i = 1, \dots, N$. We recall that $L = \max_i L_i$. Therefore, in the majority of cases the estimate for the rate of convergence based on norm $\|\cdot\|_1$ is much better than that based on norm $\|\cdot\|_0$.

3.2 Convergence for strongly convex functions

Now, we assume that the objective function in (1) is σ_α-strongly convex with respect to norm $\|\cdot\|_\alpha$, i.e.:

$$F(x) \geq F(y) + \langle F'(y), x - y\rangle + \frac{\sigma_\alpha}{2}\|x - y\|_\alpha^2 \quad \forall x, y \in \mathbb{R}^n, \tag{11}$$

where $F'(y)$ denotes some subgradient of F at y. Note that if the function f is σ-strongly convex w.r.t. extended Euclidean norm, then we can remark that it is also σ_α-strongly convex function w.r.t. norm $\|\cdot\|_\alpha$ and the following relation between the strong convexity constants holds:

$$\begin{aligned}\frac{\sigma}{L^\alpha}\sum_{i=1}^N L^\alpha \|x_i - y_i\|^2 &\geq \frac{\sigma}{L^\alpha}\sum_{i=1}^N L_i^\alpha \|x_i - y_i\|^2 \\ &\geq \sigma_\alpha \|x - y\|_\alpha^2,\end{aligned}$$

which leads to

$$\sigma_\alpha \leq \frac{\sigma}{L^\alpha}.$$

Taking $y = x^*$ in (11) and from optimality conditions $\langle F'(x^*), x - x^*\rangle \geq 0$ for all $x \in S$ we obtain:

$$F(x) - F^* \geq \frac{\sigma_\alpha}{2}\left\|x - x^*\right\|_\alpha^2. \tag{12}$$

Next, we state the convergence result of our algorithm (RCD) for solving the problem (1) with σ_α-strongly convex objective w.r.t. norm $\|\cdot\|_\alpha$.

Theorem 2 *Under the assumptions of Theorem 1, let F be also σ_α-strongly convex w.r.t. $\|\cdot\|_\alpha$. For the sequence x^k generated by algorithm (RCD) we have the following rate of convergence of the expected values of the objective function:*

$$\phi^k - F^* \leq \left(1 - \frac{2(1-\gamma)}{N^2}\right)^k (F(x^0) - F^*),$$

where γ is defined by:

$$\gamma = \begin{cases} 1 - \frac{\sigma_\alpha}{8L^{1-\alpha}}, & \textit{if } \sigma_\alpha \leq 4L^{1-\alpha} \\ \frac{2L^{1-\alpha}}{\sigma_\alpha}, & \textit{otherwise.}\end{cases}$$

Proof Based on relation (9) it follows that:

$$E_{i_k j_k}[F(x^{k+1})] \leq (1 - \frac{2}{N(N-1)})F(x^k) + \\ \frac{2}{N(N-1)} \min_{d \in S} \Big(f(x^k) + \langle \nabla f(x^k), d \rangle + L^{1-\alpha} \|d\|_\alpha^2 + h(x^k + d) \Big).$$

Then, using similar derivation as in Theorem 1 we have:

$$\begin{aligned}
&\min_{d \in S} f(x^k) + \langle \nabla f(x^k), d \rangle + L^{1-\alpha} \|d\|_\alpha^2 + h(x^k + d) \\
&\leq \min_{y \in S} F(y) + L^{1-\alpha} \left\| y - x^k \right\|_\alpha^2 \\
&\leq \min_{\beta \in [0,1]} F(\beta x^* + (1-\beta)x^k) + \beta^2 L^{1-\alpha} \left\| x^k - x^* \right\|_\alpha^2 \\
&\leq \min_{\beta \in [0,1]} F(x^k) - \beta(F(x^k) - F^*) + \beta^2 L^{1-\alpha} \left\| x^k - x^* \right\|_\alpha^2 \\
&\leq \min_{\beta \in [0,1]} F(x^k) + \beta \left(\frac{2\beta L^{1-\alpha}}{\sigma_\alpha} - 1 \right) \left(F(x^k) - F^* \right),
\end{aligned}$$

where the last inequality results from (12). The statement of the theorem is obtained by noting that $\beta^* = \min\{1, \frac{\sigma_\alpha}{4L^{1-\alpha}}\}$ and the following subcases:

1. If $\beta^* = \frac{\sigma_\alpha}{4L^{1-\alpha}}$ and we take the expectation w.r.t. η^k we get:

$$\phi^{k+1} - F^* \leq \left(1 - \frac{\sigma_\alpha}{4L^{1-\alpha}N^2}\right)(\phi^k - F^*), \tag{13}$$

2. if $\beta^* = 1$ and we take the expectation w.r.t. η^k we get:

$$\phi^{k+1} - F^* \leq \left[1 - \frac{2(1 - \frac{2L^{1-\alpha}}{\sigma_\alpha})}{N^2}\right](\phi^k - F^*). \tag{14}$$

□

3.3 Convergence in probability

Further, we establish some bounds on the required number of iterations for which the generated sequence x^k attains ϵ-accuracy with prespecified probability. In order to prove this result we use Theorem 1 from [23] and for a clear understanding we present it bellow.

Lemma 4 *[23] Let $\xi^0 > 0$ be a constant, $0 < \epsilon < \xi^0$ and consider a nonnegative nonincreasing sequence of (discrete) random variables $\{\xi^k\}_{k \geq 0}$ with one of the following properties:*

(1) $E[\xi^{k+1}|\xi^k] \leq \xi^k - \frac{(\xi^k)^2}{c}$ for all k, where $c > 0$ is a constant,
(2) $E[\xi^{k+1}|\xi^k] \leq \left(1 - \frac{1}{c}\right)\xi^k$ for all k such that $\xi^k \geq \epsilon$, where $c > 1$ is a constant.

Then, for some confidence level $\rho \in (0,1)$ we have in probability that:

$$Pr(\xi^K \leq \epsilon) \geq 1-\rho,$$

for a number K of iterations which satisfies

$$K \geq \frac{c}{\epsilon}\left(1+\log\frac{1}{\rho}\right)+2-\frac{c}{\xi^0},$$

if property (1) holds, or

$$K \geq c\log\frac{\xi^0}{\epsilon\rho},$$

if property (2) holds.

Based on this lemma we can state the following rate of convergence in probability:

Theorem 3 *Let F be a σ_α-strongly convex function satisfying Assumption 1 and $\rho > 0$ be the confidence level. Then, the sequence x^k generated by algorithm (RCD) using uniform distribution satisfies the following rate of convergence in probability of the expected values of the objective function:*

$$Pr(\phi^K - F^* \leq \epsilon) \geq 1-\rho,$$

with K satisfying

$$K \geq \begin{cases} \frac{2N^2L^{1-\alpha}R_\alpha^2}{\epsilon}\left(1+\log\frac{1}{\rho}\right)+2-\frac{2N^2L^{1-\alpha}R_\alpha^2}{F(x^0)-F^*}, & \sigma_\alpha = 0 \\ \frac{N^2}{2(1-\gamma)}\log\frac{F(x^0)-F^*}{\epsilon\rho}, & \sigma_\alpha > 0, \end{cases}$$

where $\gamma = \begin{cases} 1-\frac{\sigma_\alpha}{8L^{1-\alpha}}, & \textit{if } \sigma_\alpha \leq 4L^{1-\alpha} \\ \frac{2L^{1-\alpha}}{\sigma_\alpha}, & \textit{otherwise.} \end{cases}$

Proof Based on relation (10), we note that taking ξ^k as $\xi^k = \phi^k - F^*$, the property (1) of Lemma 4 holds and thus we get the first part of our result. Relations (13) and (14) in the strongly convex case are similar instances of property (2) in Theorem 4 from which we get the second part of the result. □

4 Generalization

In this section we study the optimization problem (1), but with general linearly coupling constraints:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \; & F(x) \quad (:= f(x)+h(x)) \\ \text{s.t.:} \; & Ax = 0, \end{aligned} \tag{15}$$

where the functions f and h have the same properties as in Assumption 1 and $A \in \mathbb{R}^{m\times n}$ is a matrix with $1 < m \leq n$. There are very few attempts to solve this problem through coordinate descent strategies and up to our knowledge the only complexity result can be found in [29].
For the simplicity of the exposition, we work in this section with the standard Euclidean norm, denoted by $\|\cdot\|_0$, on the extended space $\mathbb{R}^n$. We consider the set

of all $(m+1)$-tuples of the form $\mathcal{N} = (i^1, \dots, i^{m+1})$, where $i^p \in \{1, \dots, N\}$ for all $p = 1, \dots, m+1$. Also, we define $p_{\mathcal{N}}$ as the probability distribution associated with $(m+1)$-tuples of the form $\mathcal{N}$. Given this probability distribution $p_{\mathcal{N}}$, for this general optimization problem (15) we propose the following random coordinate descent algorithm:

```
Algorithm 2 (RCD)$_{\mathcal{N}}$
1. Choose randomly a set of $(m+1)$-tuple $\mathcal{N}_k = (i_k^1, \dots, i_k^{m+1})$
   with probability $p_{\mathcal{N}_k}$
2. Set $x^{k+1} = x^k + d_{\mathcal{N}_k}$,
```

where the direction $d_{\mathcal{N}_k}$ is chosen as follows:

$$d_{\mathcal{N}_k} = \arg\min_{s \in \mathbb{R}^n} f(x^k) + \langle \nabla f(x^k), s \rangle + \frac{L_{\mathcal{N}_k}}{2} \|s\|_0^2 + h(x^k + s)$$
$$\text{s.t.: } As = 0, \quad s_i = 0 \ \ \forall i \notin \mathcal{N}_k.$$

We can easily see that the linearly coupling constraints $Ax = 0$ prevent the development of an algorithm that performs at each iteration a minimization with respect to less than $m+1$ coordinates. Therefore we are interested in the class of iteration updates which restricts the objective function on feasible directions that consist of at least $m+1$ (block) components.

Further, we redefine the subspace S as $S = \{s \in \mathbb{R}^n : \ As = 0\}$ and additionally we denote the local subspace $S_{\mathcal{N}} = \{s \in \mathbb{R}^n : \ \ As = 0, \ \ s_i = 0 \ \ \forall i \in \mathcal{N}\}$. Note that we still consider an ordered $(m+1)$-tuple $\mathcal{N}_k = (i_k^1, \dots, i_k^{m+1})$ such that $i_k^p \neq i_k^l$ for all $p \neq l$. We observe that for a general matrix A, the previous subproblem does not necessarily have a closed form solution. However, when h is coordinatewise separable, strictly convex and piece-wise linear/quadratic with $\mathcal{O}(1)$ pieces (e.g., h given by (2)) there are efficient algorithms for solving the previous subproblem in linear-time [29]. Moreover, when h is the box indicator function (i.e., h given by (3)) we have the following: in the scalar case (i.e., $N = n$) the subproblem has a closed form solution; for the block case (i.e., $N < n$) there exist linear-time algorithms for solving the subproblem within $\mathcal{O}(\sum_{i \in \mathcal{N}_k} n_i)$ operations [2]. Through a similar reasoning as in Lemma 1 we can derive that given a set of indices $\mathcal{N} = (i^1, \dots, i^p)$, with $p \geq 2$, the following relation holds:

$$f(x + d_{\mathcal{N}}) \leq f(x) + \langle \nabla f(x), d_{\mathcal{N}} \rangle + \frac{L_{\mathcal{N}}}{2} \|d_{\mathcal{N}}\|_0^2, \tag{16}$$

for all $x \in \mathbb{R}^n$ and $d_{\mathcal{N}} \in \mathbb{R}^n$ with nonzero entries only on the blocks $i^1, \dots, i^p$. Here, $L_{\mathcal{N}} = L_{i^1} + \dots + L_{i^p}$. Moreover, based on Lemma 2 it follows that any $d \in S$ has a conformal realization defined by $d = \sum_{t=1}^s d^t$, where the elementary vectors $d^t \in S$ are conformal to d and have at most $m+1$ nonzeros. Therefore, any vector $d \in S$ can be generated by $d = \sum_{\mathcal{N}} s_{\mathcal{N}}$, where $s_{\mathcal{N}} \in S_{\mathcal{N}}$ and their extensions in $\mathbb{R}^n$ have at most $m+1$ nonzero blocks and are conformal to d. We now present the main convergence result for this method.

Theorem 4 *Let F satisfy Assumption 1. Then, the random coordinate descent algorithm (RCD)$_{\mathcal{N}}$ that chooses uniformly at each iteration $m+1$ blocks generates a* *sequence x^k satisfying the following rate of convergence for the expected values of the objective function:*

$$\phi^k - F^* \leq \frac{N^{m+1} L R_0^2}{k + \frac{N^{m+1} L R_0^2}{F(x^0) - F^*}}.$$

Proof The proof is similar to that of Theorem 1 and we omit it here for brevity.

5 Complexity analysis

In this section we analyze the total complexity (arithmetic complexity [18]) of algorithm (RCD) based on extended Euclidean norm for optimization problem (1) and compare it with other complexity estimates. Tseng presented in [29] the first complexity bounds for the (CGD) method applied to our optimization problem (1). Up to our knowledge there are no other complexity results for coordinate descent methods on the general optimization model (1).

Note that the algorithm (RCD) has an overall complexity w.r.t. extended Euclidean norm given by:

$$\mathcal{O}\left(\frac{N^2 L R_0^2}{\epsilon}\right) \mathcal{O}(i_{RCD}),$$

where $\mathcal{O}(i_{RCD})$ is the complexity per iteration of algorithm (RCD). On the other hand, algorithm (CGD) has the following complexity estimate:

$$\mathcal{O}\left(\frac{n L R_0^2}{\epsilon}\right) \mathcal{O}(i_{CGD}),$$

where $\mathcal{O}(i_{CGD})$ is the iteration complexity of algorithm (CGD). Based on the particularities and computational effort of each method, we will show in the sequel that for some optimization models arising in real-world applications the arithmetic complexity of (RCD) method is lower than that of (CGD) method. For certain instances of problem (1) we have that the computation of the coordinate directional derivative of the smooth component of the objective function is much more simpler than the function evaluation or directional derivative along an arbitrary direction. Note that the iteration of algorithm (RCD) uses only a small number of coordinate directional derivatives of the smooth part of the objective, in contrast with the (CGD) iteration which requires the full gradient. Thus, we estimate the arithmetic complexity of these two methods applied to a class of optimization problems containing instances for which the directional derivative of objective function can be computed cheaply. We recall that the process of choosing a uniformly random pair (i, j) in our method requires $\mathcal{O}(1)$ operations.

Let us structure a general coordinate descent iteration in two phases:
Phase 1: Gather first-order information to form a quadratic approximation of the original optimization problem.
Phase 2: Solve a quadratic optimization problem using data acquired at Phase 1 and update the current vector.
Both algorithms (RCD) and (CGD) share this structure but, as we will see, there is a gap between computational complexities. We analyze the following example:

$$f(x) = \frac{1}{2} x^T Z^T Z x + q^T x, \tag{17}$$

where $Z = [z_1 \; \dots \; z_n] \in \mathbb{R}^{m \times n}$ has sparse columns, with an average $p << n$ nonzero entries on each column z_i for all $i = 1, \dots, n$. A particular case of this class of functions is $f(x) = \frac{1}{2}\left\|Zx - q\right\|^2$, which has been considered for numerical experiments in [20] and [23]. The problem (1), with the aforementioned structure (17) of the smooth part of the objective function, arises in many applications: e.g., linear SVM [28], truss topology [17], internet (Google problem) [20], Chebyshev center problems [31], etc. The reader can easily find many other examples of optimization problems with cheap coordinate directional derivatives.

Further, we estimate the iteration complexity of the algorithms (RCD) and (CGD). Given a feasible x, from the expression

$$\nabla_i f(x) = \langle z_i, Zx \rangle + q^i,$$

we note that if the residual $r(x) = Zx$ is already known, then the computation of $\nabla_i f(x)$ requires $\mathcal{O}(p)$ operations. We consider that the dimension n_i of each block is of order $\mathcal{O}(\frac{n}{N})$. Thus, the (RCD) method updates the current point x on $\mathcal{O}(\frac{n}{N})$ coordinates and summing up with the computation of the new residual $r(x^+) = Zx^+$, which in this case requires $\mathcal{O}(\frac{pn}{N})$ operations, we conclude that up to this stage, the iteration of (RCD) method has numerical complexity $\mathcal{O}(\frac{pn}{N})$. However, the (CGD) method requires the computation of the full gradient for which are necessary $\mathcal{O}(np)$ operations. As a preliminary conclusion, Phase 1 has the following complexity regarding the two algorithms:

$$\begin{aligned} &\textit{(RCD). Phase 1}: \quad \mathcal{O}(\frac{np}{N}) \\ &\textit{(CGD). Phase 1}: \quad \mathcal{O}(np) \end{aligned}$$

Suppose now that for a given x, the blocks $(\nabla_i f(x), \nabla_j f(x))$ are known for (RCD) method or the entire gradient vector $\nabla f(x)$ is available for (CGD) method within previous computed complexities, then the second phase requires the finding of an update direction with respect to each method. For the general linearly constrained model (1), evaluating the iteration complexity of both algorithms can be a difficult task. Since in [29] Tseng provided an explicit total computational complexity for the cases when the nonsmooth part of the objective function h is separable and piece-wise linear/quadratic with $\mathcal{O}(1)$ pieces, for clarity of the comparison we also analyze the particular setting when h is a box indicator function as given in equation (3). For algorithm (RCD) with $\alpha = 0$, at each iteration, we require the solution of the following problem (see (3)):

$$\begin{aligned} &\min_{s_{ij} \in \mathbb{R}^{n_i + n_j}} \langle \nabla_{ij} f(x), s_{ij} \rangle + \frac{L_{ij}^0}{2} \left\| s_{ij} \right\|_0^2 \\ &\text{s.t.:} \;\; a_i^T s_i + a_j^T s_j = 0, \quad (l - x)_{ij} \le s_{ij} \le (u - x)_{ij}. \end{aligned} \tag{18}$$

It is shown in [12] that problem (18) can be solved in $\mathcal{O}(n_i + n_j)$ operations. However, in the scalar case (i.e., $N = n$) problem (18) can solved in closed form. Therefore, Phase 2 of algorithm (RCD) requires $\mathcal{O}(\frac{n}{N})$ operations. Finally, we estimate for algorithm (RCD) the total arithmetic complexity in terms of the number of blocks N as:

$$\mathcal{O}\left(\frac{N^2 L R_0^2}{\epsilon}\right) \mathcal{O}(\frac{pn}{N}).$$

On the other hand, due to the Gauss-Southwell rule, the (CGD) method requires at each iteration the solution of a quadratic knapsack problem of dimension n. It is argued in [12] that for solving the quadratic knapsack problem we need $\mathcal{O}(n)$ operations. In conclusion, the Gauss-Southwell procedure in algorithm (CGD) requires the conformal realization of the solution of a continuous knapsack problem and the selection of a "good" set of blocks $\mathcal{J}$. This last process has a different cost depending on m. Overall, we estimate the total complexity of algorithm (CGD) for one equality constraint, $m = 1$, as:

$$\mathcal{O}\left(\frac{nLR_0^2}{\epsilon}\right)\mathcal{O}(pn)$$

Table 1: Comparison of arithmetic complexities for alg. (RCD), (CGD) and [10,14] for $m = 1$.

Algorithm / $m = 1$	$h(x)$	Probabilities	Complexity
(RCD)	separable	$\frac{1}{N^2}$	$\mathcal{O}(\frac{pNnLR_0^2}{\epsilon})$
(CGD)	separable	greedy	$\mathcal{O}(\frac{pn^2LR_0^2}{\epsilon})$
Hush [10], List [14]	box indicator	greedy	$\mathcal{O}(\frac{pn^2LR_0^2}{\epsilon})$

First, we note that in the case $m = 1$ and $N << n$ (i.e., the block case) algorithm (RCD) has better arithmetic complexity than algorithm (CGD) and previously mentioned block-coordinate methods [10,14] (see Table 1). When $m = 1$ and $N = n$ (i.e., the scalar case), by substitution in the above expressions from Table 1, we have a total complexity for algorithm (RCD) comparable to the complexity of algorithm (CGD) and the algorithms from [10,14].

On the other hand, the complexity of choosing a random pair (i, j) in algorithm (RCD) is very low, i.e., we need $\mathcal{O}(1)$ operations. Thus, choosing the working pair (i, j) in our algorithm (RCD) is much simpler than choosing the working set $\mathcal{J}$ within the Gauss-Southwell rule for algorithm (CGD) which assumes the following steps: first, compute the projected gradient direction and second, find the conformal realization of computed direction; the overall complexity of these two steps being $\mathcal{O}(n)$. In conclusion, the algorithm (RCD) has a very simple implementation due to simplicity of the random choice for the working pair and a low complexity per iteration.

For the case $m = 2$ the algorithm (RCD) needs in Phase 1 to compute coordinate directional derivatives with complexity $\mathcal{O}(\frac{pn}{N})$ and in Phase 2 to find the solution of a 3-block dimensional problem of the same structure as (18) with complexity $\mathcal{O}(\frac{n}{N})$. Therefore, the iteration complexity of the (RCD) method in this case is still $\mathcal{O}(\frac{pn}{N})$. On the other hand, the iteration complexity of the algorithm (CGD) for $m = 2$ is given by $\mathcal{O}(pn + n\log n)$ [29].

For $m > 2$, the complexity of Phase 1 at each iteration of our method still requires $\mathcal{O}(\frac{pn}{N})$ operations and the complexity of Phase 2 is $\mathcal{O}(\frac{mn}{N})$, while in the (CGD) method the iteration complexity is $\mathcal{O}(m^3n^2)$ [29].

For the case $m > 1$, a comparison between arithmetic complexities of algorithms (RCD) and (CGD) is provided in Table 2. We see from this table that depending on the values of n, m and N, the arithmetic complexity of (RCD) method can be better or worse than that of the (CGD) method.

Table 2: Comparison of arithmetic complexities for algorithms (RCD) and (CGD) for $m \geq 2$.

Algorithm	$m = 2$	$m > 2$
(RCD)	$\frac{pN^2 nLR_0^2}{\epsilon}$	$\frac{(p+m)N^m nLR_0^2}{\epsilon}$
(CGD)	$\frac{(p+\log n)n^2 LR_0^2}{\epsilon}$	$\frac{m^3 n^3 LR_0^2}{\epsilon}$

We conclude from the rate of convergence and the previous complexity analysis that algorithm (RCD) is easier to be implemented and analyzed due to the randomization and the typically very simple iteration. Moreover, on certain classes of problems with sparsity structure, that appear frequently in many large-scale real applications, the arithmetic complexity of (RCD) method is better than that of some well-known methods from the literature. All these arguments make the algorithm (RCD) to be competitive in the composite optimization framework. Moreover, the (RCD) method is suited for recently developed computational architectures (e.g., distributed or parallel architectures).

6 Numerical Experiments

In this section we present extensive numerical simulations, where we compare our algorithm (RCD) with some recently developed state-of-the-art algorithms from the literature for solving the optimization problem (1): coordinate gradient descent (CGD) [29], projected gradient method for composite optimization (GM) [19] and LIBSVM [7]. We test the four methods on large-scale optimization problems ranging from $n = 10^3$ to $n = 10^7$ arising in various applications such as: support vector machine (Section 6.1), Chebyshev center of a set of points (Section 6.2) and random generated problems with an ℓ_1-regularization term (Section 6.3). Firstly, for the SVM application we compare algorithm (RCD) against (CGD) and LIBSVM and we remark that our algorithm has the best performance on large-scale problem instances with sparse data. Secondly, we also observe a more robust behavior for algorithm (RCD) in comparison with algorithms (CGD) and (GM) when using different initial points on Chebyshev center problem instances. In the final part, we test our algorithm on randomly generated problems, where the nonsmooth part of the objective function contains an ℓ_1-norm term, i.e., $\lambda \sum_{i=1}^n |x_i|$ for some $\lambda > 0$, and we compare our method with algorithms (CGD) and (GM).

We have implemented all the algorithms in C-code and the experiments were run on a PC with Intel Xeon E5410 CPU and 8GB RAM memory. In all algorithms we considered the scalar case, i.e., $N = n$ and we worked with the extended Euclidean norm ($\alpha = 0$). In our applications the smooth part f of the composite objective function is of the form (17). The coordinate directional derivative at the current point for algorithm (RCD) $\nabla_i f(x) = \langle z_i, Zx \rangle + q_i$ is computed efficiently by knowing at each iteration the residual $r(x) = Zx$. For (CGD) method the working set is chosen accordingly to Section 6 in [28]. Therefore, the entire gradient at current point $\nabla f(x) = Z^T Zx + q$ is required, which is computed efficiently using the residual $r(x) = Zx$. For gradient and residual computations we use an efficient sparse matrix-vector multiplication procedure. We coded the standard

(CGD) method presented in [29] and we have not used any heuristics recommended by Tseng in [28], e.g., the "3-pair" heuristic technique. The direction d_{ij} at the current point from subproblem (6) for algorithm (RCD) is computed in closed form for all three applications considered in this section. For computing the direction $d_H(x; \mathcal{J})$ at the current point from subproblem (7) in the (CGD) method for the first two applications, we coded the algorithm from [12] for solving quadratic knapsack problems of the form (18) that has linear time complexity. For the second application, the direction at the current point for algorithm (GM) is computed using a linear time simplex projection algorithm introduced in [11]. For the third application, we used the equivalent formulation of the subproblem (7) given in [29], obtaining for both algorithms (CGD) and (GM), an iteration which requires the solution of some double size quadratic knapsack problem of the form (18).

In the following tables, for each algorithm we present the final objective function value (obj), the number of iterations (iter) and the necessary CPU time for our computer to execute all the iterations. As the algorithms (CGD), LIBSVM and (GM) use the whole gradient information to obtain the working set and to find the direction at the current point, we also report for the algorithm (RCD) the equivalent number of full-iterations which means the total number of iterations divided by $\frac{n}{2}$ (i.e., the number of iterations groups $x^0, x^{n/2}, \dots, x^{kn/2}$).

6.1 Support vector machine

In order to better understand the practical performance of our method, we have tested the algorithms (RCD), (CGD) and LIBSVM on two-class data classification problems with linear kernel, which is a well-known real-world application that can be posed as a large-scale optimization problem in the form (1) with a sparsity structure. In this section, we describe our implementation of algorithms (RCD), (CGD) [28] and LIBSVM [7] and report the numerical results on different test problems. Note that linear SVM is a technique mainly used for text classification, which can be formulated as the following optimization problem:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \ & \frac{1}{2} x^T Z^T Z x - e^T x + \mathbf{1}_{[0,C]}(x) \\ \text{s.t.:} \ & a^T x = 0, \end{aligned} \tag{19}$$

where $\mathbf{1}_{[0,C]}$ is the indicator function for the box constrained set $[0, C]^n$, $Z \in \mathbb{R}^{m \times n}$ is the instance matrix with an average sparsity degree p (i.e., on average, Z has p nonzero entries on each column), $a \in \mathbb{R}^n$ is the label vector of instances, C is the penalty parameter and $e = [1 \dots 1]^T \in \mathbb{R}^n$. Clearly, this model fits the aforementioned class of functions (17). We set the primal penalty parameter $C = 1$ in all SVM test problems. As in [28], we initialize all the algorithms with $x^0 = 0$. The stopping criterion used in the algorithm (RCD) is: $f(x^{k-j}) - f(x^{k-j+1}) \leq \epsilon$, where $j = 0, \dots, 10$, while for the algorithm (CGD) we use the stopping criterion $f(x^k) - f(x^{k+1}) \leq \epsilon$, where $\epsilon = 10^{-5}$.

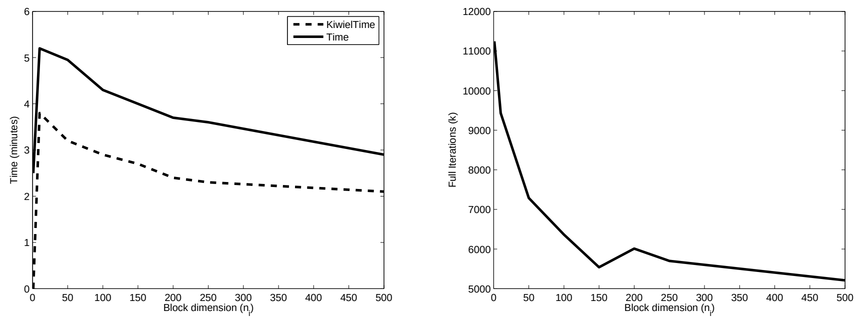
We report in Table 3 the results for algorithms (RCD), (CGD) and LIBSVM implemented in the scalar case, i.e., $N = n$. The data used for the experiments can be found on the LIBSVM webpage (http://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/). For problems with very large dimensions, we generated the data randomly (see "test1" and "test2") such that the nonzero elements of Z fit into the

Table 3: Comparison of algorithms (RCD), (CGD) and library LIBSVM on SVM problems.

Data set	n/m	(RCD)	(CGD)	LIBSVM
		full-iter/obj/time(min)	iter/obj/time(min)	iter/obj/time(min)
a7a	16100/122 ($p = 14$)	11242/-5698.02/2.5	23800/-5698.25/21.5	63889/-5699.25/0.46
a8a	22696/123 ($p = 14$)	22278/-8061.9/18.1	37428/-8061.9/27.8	94877/-8062.4/1.42
a9a	32561/123 ($p = 14$)	15355/-11431.47/7.01	45000/-11431.58/89	78244/-11433.0/2.33
w8a	49749/300 ($p = 12$)	15380/-1486.3/26.3	19421/-1486.3/27.2	130294/-1486.8/42.9
ijcnn1	49990/22 ($p = 13$)	7601/-8589.05/6.01	9000/-8589.52/16.5	15696/-8590.15/1.0
web	350000/254 ($p = 85$)	1428/-69471.21/29.95	13600/-27200.68/748	59760/-69449.56/467
covtyp	581012/54 ($p = 12$)	1722/-337798.34/38.5	12000/-24000/480	466209/-337953.02/566.5
test1	$2.2 \cdot 10^6/10^6$ ($p = 50$)	228/-1654.72/51	4600/-473.93/568	*
test2	$10^7/5 \cdot 10^3$ ($p = 10$)	350/-508.06/112.65	502/-507.59/516.66	*

available memory of our computer. For each algorithm we present the final objective function value (obj), the number of iterations (iter) and the necessary CPU time (in minutes) for our computer to execute all the iterations. For the algorithm (RCD) we report the equivalent number of full-iterations, that is the number of iterations groups $x^0, x^{n/2}, \ldots, x^{kn/2}$. On small test problems we observe that LIBSVM outperforms algorithms (RCD) and (CGD), but we still have that the CPU time for algorithm (RCD) does not exceed 30 min, while algorithm (CGD) performs much worse. On the other hand, on large-scale problems the algorithm (RCD) has the best behavior among the three tested algorithms (within a factor of 10). For very large problems ($n \geq 10^6$), LIBSVM has not returned any result within 10 hours.

Fig. 1: Performance of algorithm (RCD) for different block dimensions.



For the block case (i.e., $N \leq n$), we have plotted for algorithm (RCD) on the test problem "a7a" the CPU time and total time (in minutes) to solve knapsack problems (left) and the number of full-iterations (right) for different dimensions of the blocks n_i. We see that the number of iterations decreases with the increasing dimension of the blocks, while the CPU time increases w.r.t. the scalar case due to the fact that for $n_i > 1$ the direction d_{ij} cannot be computed in closed form as in the scalar case (i.e., $n_i = 1$), but requires solving a quadratic knapsack problem (18) whose solution can be computed in $\mathcal{O}(n_i + n_j)$ operations [12].

6.2 Chebyshev center of a set of points

Many real applications such as location planning of shared facilities, pattern recognition, protein analysis, mechanical engineering and computer graphics (see e.g., [31] for more details and appropriate references) can be formulated as finding the Chebyshev center of a given set of points. The Chebyshev center problem involves the following: given a set of points $z_1, \dots, z_n \in \mathbb{R}^m$, find the center z_c and radius r of the smallest enclosing ball of the given points. This geometric problem can be formulated as the following optimization problem:

$$\begin{aligned} &\min_{r, z_c} \; r \\ &\text{s.t.:} \quad \|z_i - z_c\|^2 \leq r \quad \forall i = 1, \dots, n, \end{aligned}$$

where r is the radius and z_c is the center of the enclosing ball. It can be immediately seen that the dual formulation of this problem is a particular case of our linearly constrained optimization model (1):

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} \|Zx\|^2 - \sum_{i=1}^{n} \|z_i\|^2 x_i + \mathbf{1}_{[0,\infty)}(x) \qquad (20) \\ &\text{s.t.:} \quad e^T x = 1, \end{aligned}$$

where Z is the matrix containing the given points z_i as columns. Once an optimal solution x^* for the dual formulation is found, a primal solution can be recovered as follows:

$$r* = \left(- \left\|Zx^*\right\|^2 + \sum_{i=1}^{n} \|z_i\|^2 x_i^* \right)^{1/2}, \qquad z_c^* = Zx^*. \qquad (21)$$

The direction d_{ij} at the current point in the algorithm (RCD) is computed in closed form. For computing the direction in the (CGD) method we need to solve a quadratic knapsack problem that has linear time complexity [12]. The direction at the current point for algorithm (GM) is computed using a linear time simplex projection algorithm introduced in [11]. We compare algorithms (RCD), (CGD) and (GM) for a set of large-scale problem instances generated randomly with a uniform distribution. We recover a suboptimal radius and Chebyshev center using the same set of relations (21) evaluated at the final iteration point x^k for all three algorithms.

In Fig. 2 we present the performance of the three algorithms (RCD), (GM) and (CGD) on a randomly generated matrix $Z \in \mathbb{R}^{2 \times 1000}$ for 50 full-iterations

Fig. 2: Performance of algorithms (RCD), (GM) and (CGD) for 50 full-iterations and initial point e_1 (top) and $\frac{e}{n}$ (bottom) on a randomly generated matrix $Z \in \mathbb{R}^{2\times 1000}$.

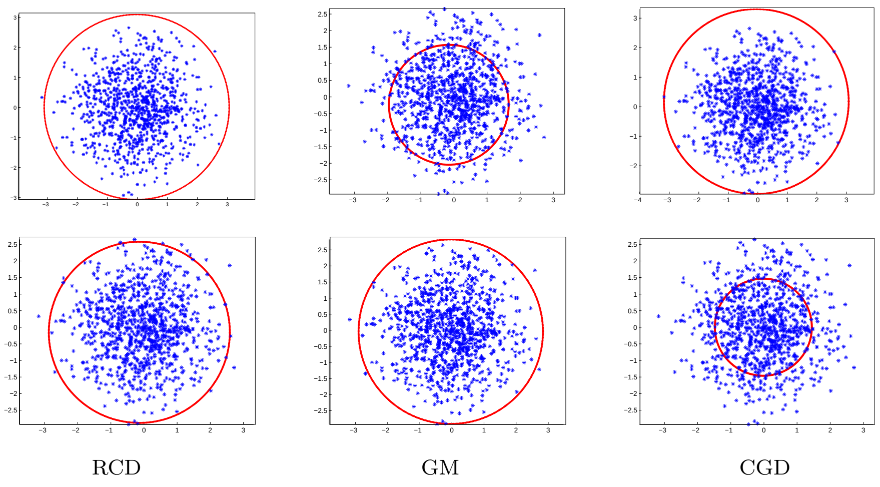
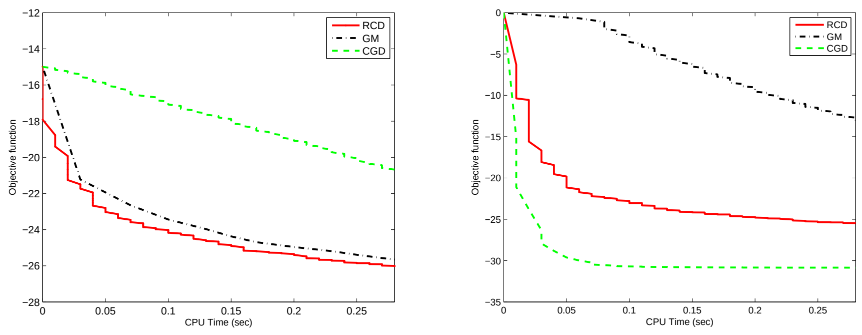


Fig. 3: Time performance of algorithms (RCD), (GM) and (CGD) for initial point $\frac{e}{n}$ (left) and e_1(right) on a randomly generated matrix $Z \in \mathbb{R}^{30\times 1000}$.



with two different initial points: $x^0 = e_1$ (the vector with the first entry 1 and the rest of the entries zeros) and $x^0 = \frac{e}{n}$. Note that for the initial point $x^0 = e_1$, the algorithm (GM) is outperformed by the other two methods: (RCD) and (CGD). Also, if all three algorithms are initialized with $x^0 = \frac{e}{n}$, the algorithm (CGD) has the worst performance among all three. We observe that our algorithm (RCD) is very robust against the initial point choice.

In Fig. 3 we plot the objective function evaluation over time (in seconds) for the three algorithms (RCD), (GM) and (CGD) on a matrix $Z \in \mathbb{R}^{30\times 1000}$. We observe that the algorithm (RCD) has a comparable performance with algorithm (GM) and a much better performance than (CGD) when the initial point is taken $\frac{e}{n}$. On the other hand, the algorithm (GM) has the worst behavior among all three methods when sparse initializations are used. However, the behavior of our algorithm (RCD) is not dependent on the sparsity of the initial point.

Table 4: Comparison of algorithms (RCD), (CGD) and (GM) on Chebyshev center problems.

x^0	n m	(RCD)	(CGD)	GM
		full-iter/obj/time(sec)	iter/obj/time(sec)	iter/obj/time(sec)
$\frac{e}{n}$	$5 \cdot 10^3$ 10	2064/-79.80/0.76	4620/-79.80/5.3	17156/-79.82/5.6
	10^4 10	6370/-84.71/4.75	9604/-84.7/23.2	42495/-84.71/28.01
	$3 \cdot 10^4$ 10	13213/-87.12/31.15	27287/-86.09/206.52	55499/-86.09/111.81
	$5 \cdot 10^3$ 30	4269/-205.94/2.75	823/-132.08/0.6	19610/-204.94/13.94
	10^4 30	5684/-211.95/7.51	9552/-211.94/33.42	28102/-210.94/40.18
	$3 \cdot 10^4$ 30	23744/-215.66/150.86	156929/-214.66/1729.1	126272/-214.66/937.33
e_1	$5 \cdot 10^3$ 10	2392/-79.81/0.88	611/-80.8/0.77	29374/-79.8/9.6
	10^4 10	9429/-84.71/7.05	350/-85.2/0.86	60777/-84.7/40.1
	$3 \cdot 10^4$ 10	13007/-87.1/30.64	615/-88.09/6.20	129221/-86.09/258.88
	$5 \cdot 10^3$ 30	2682/-205.94/1.73	806/-206.94/1.13	35777/-204.94/25.29
	10^4 30	4382/-211.94/5.77	594/-212.94/2.14	59825/-210.94/85.52
	$3 \cdot 10^4$ 30	16601/-215.67/102.11	707 /-216.66/8.02	191303/-214.66/1421

In Table 4, for a number of $n = 5 \cdot 10^3, 10^4$ and $3 \cdot 10^4$ points generated randomly using uniform distribution in $\mathbb{R}^{10}$ and $\mathbb{R}^{30}$, we compare all three algorithms (RCD), (CGD) and (GM) with two different initial points: $x^0 = e_1$ and $x^0 = \frac{e}{n}$. Firstly, we have computed f^* with the algorithm (CGD) using $x^0 = e_1$ and imposing the termination criterion $f(x^k) - f(x^{k+1}) \leq \epsilon$, where $\epsilon = 10^{-5}$. Secondly, we have used the precomputed optimal value f^* to test the other algorithms with termination criterion $f(x^k) - f^* \leq 1$ or 2. We clearly see that our algorithm (RCD) has superior performance over the (GM) method and is comparable with (CGD) method when we start from $x^0 = e_1$. When we start from $x^0 = \frac{e}{n}$ our algorithm provides better performance in terms of objective function and CPU time (in seconds) than the (CGD) and (GM) methods (at least 6 times faster). We also observe that our algorithm is not sensitive w.r.t. the initial point.

6.3 Random generated problems with ℓ_1-regularization term

In this section we compare our algorithm (RCD) with the methods (CGD) and (GM) on problems with composite objective function, where the nonsmooth part contains an ℓ_1-regularization term $\lambda \sum_{i=1}^{n} |x_i|$. Many applications from signal processing and data mining can be formulated into the following optimization problem

[5,22]:

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}x^T Z^T Z x + q^T x + \left(\lambda \sum_{i=1}^{n} |x_i| + \mathbf{1}_{[l,u]}(x)\right) \qquad (22)$$
$$\text{s.t.:} \quad a^T x = b,$$

where $Z \in \mathbb{R}^{m\times n}$ and the penalty parameter $\lambda > 0$. Further, the rest of the parameters are chosen as follows: $a = e$, $b = 1$ and $-l = u = 1$. The direction d_{ij} at the current point in the algorithm (RCD) is computed in closed form. For computing the direction in the (CGD) and (GM) methods we need to solve a double size quadratic knapsack problem of the form (18) that has linear time complexity [12].

Table 5: Comparison of algorithms (RCD), (CGD) and (GM) on ℓ_1-regularization problems.

x^0	λ	n	(RCD)	(CGD)	(GM)
			full-iter/obj/time(sec)	iter/obj/time(sec)	iter/obj/time(sec)
$\frac{e}{n}$	0.1	10^4	905/-6.66/0.87	10/-6.67/0.11	9044/-6.66/122.42
		$5 \cdot 10^4$	1561/-0.79/12.32	8/-0.80/0.686	4242/-0.75/373.99
		10^5	513/-4.12/10.45	58/-4.22/7.55	253/-4.12/45.06
		$5 \cdot 10^5$	245/-2.40/29.03	13/-2.45/9.20	785/-2.35/714.93
		$2 \cdot 10^6$	101/-10.42/61.27	6/-10.43/22.79	1906/-9.43/6582.5
		10^7	29/-2.32/108.58	7/-2.33/140.4	138/-2.21/2471.2
	10	10^4	316/11.51/0.29	5858/11.51/35.67	22863/11.60/150.61
		$5 \cdot 10^4$	296/23.31/17.65	1261/23.31/256.6	1261/23.40/154.6
		10^5	169/22.43/12.18	46/22.34/15.99	1467/22.43/423.4
		$5 \cdot 10^5$	411/21.06/50.82	37/21.02/22.46	849/22.01/702.73
		$2 \cdot 10^6$	592/11.84/334.30	74/11.55/182.44	664/12.04/2293.1
		10^7	296/20.9/5270.2	76/20.42/1071.5	1646/20.91/29289.1
e_1	0.1	10^4	536/-6.66/0.51	4/-6.68/0.05	3408/-6.66/35.26
		$5 \cdot 10^4$	475/-0.79/24.30	84564/-0.70/7251.4	54325/-0.70/4970.7
		10^5	1158/-4.07/21.43	24/-4.17/4.83	6699/-3.97/1718.2
		$5 \cdot 10^5$	226/-2.25/28.81	8/-2.35/9.03	2047/-2.25/2907.5
		$2 \cdot 10^6$	70/-10.42/40.4	166/-10.41/632	428/-10.33/1728.3
		10^7	30/-2.32/100.1	*	376/-2.22/6731
	10	10^4	1110/11.51/1.03	17/11.52/0.14	184655/11.52/1416.8
		$5 \cdot 10^4$	237/23.39/1.22	21001/23.41/4263.5	44392/23.1/5421.4
		10^5	29/22.33/2.47	*	*
		$5 \cdot 10^5$	29/21.01/3.1	*	*
		$2 \cdot 10^6$	5/11.56/2.85	*	*
		10^7	2/20.42/4.51	*	*

In Table 5, for dimensions ranging from $n = 10^4$ to $n = 10^7$ and $m = 10$, we have generated randomly the matrix $Z \in \mathbb{R}^{m\times n}$ and $q \in \mathbb{R}^n$ using uniform distribution. We compare all three algorithms (RCD), (CGD) and (GM) with two different initial points $x^0 = e_1$ and $x^0 = \frac{e}{n}$ and two different values of the penalty parameter $\lambda = 0.1$ and $\lambda = 10$. Firstly, we have computed f^* with the algorithm (CGD) using $x^0 = \frac{e}{n}$ and imposing the termination criterion $f(x^k) - f(x^{k+1}) \leq \epsilon$, where $\epsilon = 10^{-5}$. Secondly, we have used the precomputed optimal value f^* to test the other algorithms with termination criterion $f(x^k) - f^* \leq 0.1$ or 1. For the penalty parameter $\lambda = 10$ and initial point e_1 the algorithms (CGD) and (GM) have not returned any result within 5 hours. We can clearly see from Table 5 that for most of the tests with the initialization $x^0 = e_1$ our algorithm (RCD) performs up to 100 times faster than the other two methods. Also, note that when we start from $x^0 = \frac{e}{n}$ our algorithm provides a comparable performance in terms of objective function and CPU time (in seconds) with algorithm (CGD). Finally, we observe that algorithm (RCD) is the most robust w.r.t. the initial point among all three tested methods.

References

1. A. Beck and L. Tetruashvilli, *On the convergence of block-coordinate descent type methods*, submitted to Mathematical Programming, 2012.
2. P. Berman, N. Kovoor and P.M. Pardalos, *Algorithms for least distance problem*, Complexity in Numerical Optimization, P.M. Pardalos ed., World Scientific, 33–56, 1993.
3. D.P. Bertsekas, *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, 2003.
4. D.P. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1999.
5. E. Candes, J. Romberg and T. Tao, *Robust Uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Transactions on Information Theory, 52, 489–509, 2006.
6. S. Chen, D. Donoho and M. Saunders, *Atomic decomposition by basis pursuit*, SIAM Review, 43, 129–159, 2001.
7. C.C. Chang and C.J. Lin, *LIBSVM: a library for support vector machines*, ACM Transactions on Intelligent Systems and Technology, 27, 1–27, 2011.
8. Y.H. Dai and R. Fletcher, *New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds*, Mathematical Programming, 106 (3), 403–421, 2006.
9. M.C. Ferris and T.S. Munson, *Interior-point methods for massive support vector machines*, SIAM Journal of Optimization, 13 (3), 783–804, 2003.
10. D. Hush, P. Kelly, C. Scovel and I. Steinwart, *QP algorithms with guaranteed accuracy and run time for support vector machines*, Journal of Machine Learning Research, 7, 733–769, 2006.
11. J. Judice, M. Raydan, S. Rosa and S. Santos, *On the solution of the symmetric eigenvalue complementarity problem by the spectral projected gradient algorithm*, Numerical Algorithms, 47, 391–407, 2008.
12. K.C. Kiwiel, *On linear-time algorithms for the continuous quadratic knapsack problem*, Journal of Optimization Theory and Applications, 134, 549–554, 2007.
13. C.J. Lin, S. Lucidi, L. Palagi, A. Risi and M. Sciandrone, *A decomposition algorithm model for singly linearly constrained problems subject to lower and upper bounds*, Journal of Optimization Theory and Applications, 141, 107–126, 2009.
14. N. List and H.U. Simon, *General polynomial time decomposition algorithms*, Lecture Notes in Computer Science, vol. 3559, Springer, 308–322, 2005.
15. I. Necoara, V. Nedelcu and I. Dumitrache, *Parallel and distributed optimization methods for estimation and control in networks*, Journal of Process Control, 21(5), 756–766, 2011.
16. I. Necoara, Y Nesterov and F. Glineur, *A random coordinate descent method on large optimization problems with linear constraints*, Technical Report, University Politehnica Bucharest, 2011 (http://acse.pub.ro/person/ion-necoara).

17. Yu. Nesterov and S. Shpirko, *Primal-dual subgradient method for huge-scale linear conic problems*, Optimization Online, 2012.
18. Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Kluwer, 2004.
19. Y. Nesterov, *Gradient methods for minimizing composite objective functions*, Core discussion paper, 76/2007, Universite Catholique de Louvain, 2007.
20. Y. Nesterov, *Efficiency of coordinate descent methods on huge-scale optimization problems*, SIAM Journal on Optimization 22(2), 341–362, 2012.
21. J.C. Platt, *Fast training of support vector machines using sequential minimal optimization*, Advances in Kernel Methods: Support Vector Learning, MIT Press, 1999.
22. Z. Qin, K. Scheinberg and D. Goldfarb, *Efficient Block-coordinate Descent Algorithms for the Group Lasso*, submitted, 2010.
23. P. Richtarik and M. Takac, *Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function*, Mathematical Programming, 2012.
24. R.T. Rockafeller, *The elementary vectors of a subspace in* $\mathbb{R}^N$, Combinatorial Mathematics and its Applications, Proceedings of the Chapel Hill Conference 1967, R.C. Bose and T.A. Downling eds., Univ. North Carolina Press, 104–127, 1969.
25. R.T. Rockafeller, *Network flows and Monotropic Optimization*, Wiley-Interscience, 1984.
26. A. Saha and A. Tewari, *On the non-asymptotic convergence of cyclic coordinate descent methods*, submitted to SIAM Journal on Optimization, 2010.
27. P. Tseng and S. Yun, *A Coordinate Gradient Descent Method for Nonsmooth Separable Minimization*, Mathematical Programming, 117, 387–423, 2009.
28. P. Tseng and S. Yun, *A Coordinate Gradient Descent Method for Linearly Constrained Smooth Optimization and Support Vector Machines Training*, Computational Optimization and Applications, 47, 179–206, 2010.
29. P. Tseng and S. Yun, *A Block-Coordinate Gradient Descent Method for Linearly Constrained Nonsmooth Separable Optimization*, Journal of Optimization Theory and Applications, 140, 513–535, 2009.
30. L. Xiao and S. Boyd, *Optimal Scaling of a gradient method for distributed resource allocation*, Journal of Optimization Theory and Applications, 129, 2006.
31. S. Xu, M. Freund R and J. Sun, *Solution methodologies for the smallest enclosing circle problem*, Computational Optimization and Applications, 25(1-3), 283–292, 2003.