A Flexible Coordinate Descent Method for Big Data Applications

Kimon Fountoulakis[*] Rachael Tappenden[†]

January 22, 2016

Abstract

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 Flexible Coordinate Descent (FCD). At each iteration of FCD, 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 present several high probability iteration complexity results to show that convergence of FCD is guaranteed theoretically. Finally, we present numerical results on large-scale problems to demonstrate the practical performance of the method.

Keywords. large scale optimization; second-order methods; curvature information; block coordinate descent; nonsmooth problems; iteration complexity; randomized.

AMS Classification. 49M15; 49M37; 65K05; 90C06; 90C25; 90C53.

1 Introduction

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

$$\min_{x\in\mathbf{R}^N} F(x) := f(x) + \Psi(x), \tag{1}$$

where $f(x)$ is a smooth convex function and $\Psi(x)$ is a (possibly) nonsmooth, (coordinate) separable, real valued convex function (this will be defined precisely in Section 2). Problems of the form (1) arise in many important scientific fields, and applications include machine learning [42], regression [37] and compressed sensing [5, 4, 10]. 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 (1) are large-scale, 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). 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 matrix vector products, and ideally, they should offer fast progress towards optimality.

[*]K. Fountoulakis is with the International Computer Science Institute, Department of Statistics, University of California Berkeley, 1947 Center St, Ste. 600, Berkeley, CA 94704, USA. e-mail: kfount@berkeley.edu.

[†]R. Tappenden (corresponding author) is with the Department of Applied Mathematics and Statistics, Johns Hopkins University, Whitehead Hall, 3400 N Charles St, Baltimore, MD 21218, USA. e-mail: rtappen1@jhu.edu.

First order methods, and in particular randomized coordinate descent methods, have found great success in this area because they 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 [29] 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.

Unfortunately, randomized coordinate descent methods have two significant drawbacks. Firstly, due to their coordinate nature, coordinate descent methods are usually efficient only on problems with a high degree of partial separability,[1] and performance suffers when there is a high dependency between variables. Secondly, as first-order methods, coordinate descent methods do not usually capture essential curvature information and have been shown to struggle on complicated sparse problems [14, 15].

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 FCD a search direction is obtained by solving a local quadratic model *approximately.* The quadratic model is defined by a matrix representing approximate second order information, so that essential curvature information is incorporated. The model need only be minimized *approximately* to give an *inexact* search direction, so that the process is efficient in practice. (The termination condition for the quadratic subproblem is inspired by [3], and we discuss this in more detail later.) A line search is then employed along this inexact direction, in order to guarantee a monotonic decrease in the objective function.

FCD is computationally efficient. At each iteration of FCD, a block of coordinates is randomly selected, which is inexpensive. The method allows the use of inexact search directions, i.e., the quadratic model is minimized approximately. Intuitively, it is expected that this will lead to a reduction in algorithm running time, compared with minimizing the model exactly at every iteration. Also, the line search step depends on a subset of the coordinates only (corresponding to the randomly selected block of coordinates), so it is much cheaper compared with working in the full dimensional space. We note that the per iteration computational cost of the FCD method may be slightly higher than other randomized coordinate descent methods. This is due to the incorporation of the matrix representing partial second order information — the matrix gives rise to a quadratic model that is (in general) nonseparable, so the associated subproblem may not have a closed form solution. However, in the numerical results presented later in this work, we show that the method is more robust (and efficient) in practice, and often FCD will require fewer iterations to reach optimality, compared with other state-of-the-art methods. i.e., we show that FCD is able to solve difficult problems, on which other coordinate descent method may struggle.

The FCD method is supported by theoretical convergence guarantees in the form of high probability iteration complexity results. These iteration complexity results provide an explicit expression for the number of iterations k, which guarantees that, for any given error tolerance $\epsilon > 0$, and confidence level $\rho \in (0, 1)$, the probability that FCD achieves an ϵ-optimal solution, exceeds $1 - \rho$, i.e., $\mathbf{P}(F(x_k) - F^* \leq \epsilon) \geq 1 - \rho$.

1.1 Literature review

Coordinate descent methods are some of the oldest and simplest iterative methods, and they are often better known in the literature under various names such as Jacobi methods, or Gauss-Seidel methods, among others. It has been observed that these methods may suffer from poor practical performance, particularly on ill-conditioned problems, and until recently, often higher

[1]See [30, Equation (2)] or [36, Definition 13] for a precise definition of 'partial separability'.

order methods have been favoured by the optimization community. 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 to real world instances of very large/huge scale problems in a reasonable amount of time.

Currently, state-of-the-art randomized coordinate descent methods include that of Richtàrik and Takàč [29], where the method can be applied to unconstrained convex composite optimization problems of the form (1). The algorithm is supported by theoretical convergence guarantees in the form of high probability iteration complexity results, and [29] also reports very impressive practical performance on highly separable large scale problems. Their work has also been extended to the parallel case [30], to include acceleration techniques [13], and to include the use of inexact updates [35].

Other important works on coordinate descent methods include that of Tseng and Yun in [38, 39, 40], Nesterov [23] for huge-scale problems, work in [20] and [36] that improves the complexity analysis of [29] and [28] respectively, coordinate descent methods for group lasso problems [26, 34] or for problems with general regularizers [33, 41], a coordinate descent method that uses a sequential active set strategy [31], and coordinate descent for constrained optimization problems [21].

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), or in the special case where $\Psi(x) = \|x\|_1$. For example, Karimi and Vavasis [16] have developed a proximal quasi-Newton method for l_1-regularized least squares problems, Lee, Sun and Saunders [18, 19] have proposed a family of Newton-type methods for solving problems of the form (1) and Scheinberg and Tang [32] present iteration complexity results for a proximal Newton-type method. Moreover, the authors in [3] 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.

Facchinei and coauthors in [7, 11, 12] have also made a significant contribution to the literature on randomized coordinate descent methods. Two new algorithms are introduced in those works, namely FLEXA [11, 12] and HyFLEXA [7]. Both methods can be applied to problems of the form (1) where f is allowed to be nonconvex, and for HyFLEXA, Ψ is allowed to be nonseparable. We stress that nonconvexity of f, and nonseparability of Ψ are *outside the scope of this work*. Nevertheless, these methods are pertinent to the current work, so we discuss them now.

For FLEXA and HyFLEXA, (as is standard in the current literature), the block structure of the problem is fixed throughout the algorithm: the data vector $x \in \mathbf{R}^N$ is partitioned into $n(\leq N)$ blocks, $x = (x^1, \ldots, x^n) \in \mathbf{R}^N$. At each iteration, one block, all blocks, or some number in between, are selected in such a way that some specified error bound is satisfied. Next, a local *convex* model is formed and that model is approximately minimized to find a search direction. Finally, a convex combination of the current point, and the search direction is taken, to give a new point, and the process is repeated. The model they propose is *block separable*, so the subproblems for each block are independent. (See (3) in [12] for the local/block quadratic model, or F1–F3 in [7] for a description of the strongly convex local/block model). This has the advantage that the subproblem for each block can be solved/minimized in parallel, but has the disadvantage that no interaction between different blocks is captured by either algorithm. We also note that, while these algorithms are equipped with global convergence guarantees, iteration complexity results have *not been established* for either method, which is a significant drawback.

Another important method that has recently been proposed is a randomized coordinate decent method by Qu et al. [27]. One of the most notable features of their algorithm is that the block structure is *not fixed throughout the algorithm.* That is, at each iteration of their algorithm, a random subset of coordinates S_k is selected from the set of coordinates $\{1, \ldots, N\}$. Then, a quadratic model is minimized exactly to obtain an update, and a new point is generated by applying the update to the current point. Unfortunately, this algorithm is only applicable to strongly convex smooth functions, or strongly convex empirical risk minimization problems, but is *not suitable for general convex problems of the form* (1). Moreover, the matrices used to define their quadratic model must obey several (strong) assumptions, and the quadratic model must be minimized *exactly.* These restrictions may make the algorithm inconvenient from a practical perspective.

The purpose of this work is to build upon the positive features of some existing coordinate descent methods and create a general flexible framework that can be used to solve any problem of the form (1). We will adopt some of the ideas in [3, 12, 27] (including variable block structure; the incorporation of partial approximate second order information; the practicality of approximate solves and inexact updates; cheap line search) and create a new flexible randomized coordinate descent framework that is not only computationally practical, but is also supported by strong theoretical guarantees.

1.2 Major Contributions

In this section we list several of the major contributions of this paper, which correspond to the central ideas of our FCD algorithm.

1. **Blocks can vary throughout iterations.** Existing randomized coordinate descent methods initially partition the data $x \in \mathbf{R}^N$ into $n(\leq N)$ blocks ($x = (x^1, \ldots, x^n) \in \mathbf{R}^N$) and at each iteration, one/all/several of the blocks are selected according to some probability distribution, and those blocks are then updated. For those methods, the block partition is *fixed throughout the entire algorithm.* So, for example, coordinates in block x^1 will always be updated all together as a group, independently of any other block x^i $i \neq 1$. One of the main contributions of this work is that we allow the blocks to *vary throughout the algorithm.* i.e., the method is not restricted to a fixed block structure. No partition of x need be initialized. Rather, when FCD is initialized, a parameter $1 \leq \tau \leq N$ is chosen and then, at each iteration of FCD, a subset $S \subseteq \{1, \ldots, N\}$ of size $|S| = \tau$ is randomly selected/generated, and the coordinates of x corresponding to the indices in S are updated. To the best of our knowledge, the only other paper that allows for this random mixing of coordinates/varying blocks strategy, is the work by Qu et al. [27].[2] This variable block structure is crucial with regards to our next major contribution.

   Remark 1. *We note that the algorithms of Tseng and Yun [38, 39, 40] also allow a certain amount of coordinate mixing. However, their algorithms enforce deterministic rules for coordinate subset selection (either a 'generalized Gauss-Seidel', or Gauss-Southwell rule), which is different from our randomized selection rule. This difference is important because their deterministic strategy can be expensive/inconvenient to implement in practice, whereas our random scheme is simple and cheap to implement.*

2. **Model: Incorporation of partial second order information.** FCD uses a quadratic model to determine the search direction. The model is defined by any symmetric positive

[2]An earlier version of this work, which, to the best of our knowledge, was the first to propose varying random block selection, is cited by [27].

definite matrix $H^S(x_k)$, which depends on both the subset of coordinates S, and also on the current point x_k. i.e., $H^S(x_k)$ is not fixed; rather, it can change/vary throughout the algorithm. We stress that this is a key advantage over most existing methods, which enforce the symmetric positive definite matrix defining their quadratic model to be fixed with respect to the fixed block structure, and/or iteration counter. To the best of our knowledge, the only works that allow the matrix $H^S(x_k)$ to vary with respect to the subset S is this one (FCD), and the work by Qu et al. [27]. A crucial observation is that, if $H^S(x_k)$ approximates the Hessian, our approach allows *every element of the Hessian to be accessed.* On the other hand, other existing methods can only access blocks along the diagonal of (some perturbation of) the Hessian, (including [7, 11, 12, 30, 29, 36].)

   Furthermore, the only restriction we make on the choice of the matrix $H^S(x_k)$ is that it be symmetric and positive definite. This is a much weaker assumption than made by Qu et al. [27], who insist that the matrix defining their quadratic model be a principle submatrix of some fixed/predefined $N \times N$ matrix M say, and the large matrix M must be symmetric and *sufficiently positive definite.* (See Section 2.1 in [27].)

3. **Inexact search directions.** To ensure that FCD is computationally practical, it is imperative that the iterates are inexpensive, and this is achieved this through the use of *inexact* updates. That is, the model can be *approximately* minimized, giving *inexact search directions.* This strategy makes intuitive sense because, if the current point is far from the true solution, it may be wasteful/unnecessary to exactly minimize the model. Any algorithm can be used to approximately minimize the model, and if $H^S(x_k)$ approximates the Hessian, then the search direction obtained by minimizing the model is an approximate Newton-type direction. Importantly, the stopping conditions for the 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.

   Remark 2. *The precise form of our inexactness termination criterion is important because, not only is it computationally practical (i.e., easy to verify), it also allowed us to derive iteration complexity results for FCD (see point 5). We considered several other termination criterion formulations, but we were unable to obtain corresponding iteration complexity results (although global convergence results were possible). Currently, the majority of randomized CD methods require exact solves for their inner subproblems and we believe that a major reason for this is because iteration complexity results are significantly easier to obtain in the exact case. Coming up with practical inexact termination criteria for randomized CD methods that also allow the derivation of iteration complexity results seems to be a major hurdle in this area, although some progress is being made, e.g., [6, 9, 8, 35].*

4. **Line search.** FCD includes a line search to ensure a monotonic decrease of the objective function as iterates progress. (The line search is needed because we only make weak assumptions on the matrix $H^S(x_k)$.) The line search is inexpensive to perform because, at each iteration, *it depends on a subset of coordinates S only.* One of the major advantages of incorporating second-order information combined with a 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. In fact, for all experiments that we performed, unit step sizes were accepted by the line search for the majority of the iterations.

5. **Convergence theory.** We provide a complete convergence theory for FCD. In particular, we provide high probability iteration complexity guarantees for the algorithm in both

the convex and strongly convex cases, and in the cases of both inexact or exact search directions. Our results show that FCD converges at a sublinear rate for convex functions, and at a linear rate for strongly convex functions.

1.3 Paper Outline

The paper is organised as follows. In Section 2 we introduce the notation and definitions that are used throughout this paper, including the definition of the quadratic model, and stationarity conditions. A thorough description of the FCD algorithm is presented in Section 3, including how the blocks are selected/sampled at each iteration (Section 3.1), a description of the search direction, several suggestions for selecting the matrices $H^S(x_k)$, and analysis of the subproblem/model termination conditions (Section 3.2), a definition of the line search (Section 3.3) and we also present several concrete problem examples (Section 3.4). In Section 4 we show that the line search in FCD will always be satisfied for some positive step α, and that the objective function value is guaranteed to decrease at every iteration. Section 5 introduces several technical results, and in Section 6 we present our main iteration complexity results. In Section 7 we give a simplified analysis of FCD in the smooth case. Finally, several numerical experiments are presented in Section 9, which show that the algorithm performs very well in practice, even on ill-conditioned problems.

2 Preliminaries

Here we introduce the notation and definitions that are used in this paper, and we also present some important technical results. Throughout the paper we denote the standard Euclidean norm by $\|\cdot\| \equiv \sqrt{\langle\cdot,\cdot\rangle}$ and the ellipsoidal norm by $\|\cdot\|_A \equiv \sqrt{\langle\cdot, A\cdot\rangle}$, where A is a symmetric positive definite matrix. Furthermore, $\mathbf{R}^N_{++}$ denotes a vector in $\mathbf{R}^N$ with (strictly) positive components.

Subgradient and subdifferential. For a function $\Phi : \mathbf{R}^N \to \mathbf{R}$ the elements $s \in \mathbf{R}^N$ that satisfy

$$\Phi(y) \geq \Phi(x) + \langle s, y - x\rangle, \quad \forall y \in \mathbf{R}^N,$$

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

Convexity. A function $\Phi : \mathbf{R}^N \to \mathbf{R}$ is strongly convex with convexity parameter $\mu_\Phi > 0$ if

$$\Phi(y) \geq \Phi(x) + \langle s, y - x\rangle + \tfrac{\mu_\Phi}{2}\|y - x\|^2, \quad \forall x, y \in \mathbf{R}^N, \tag{2}$$

where $s \in \partial\Phi(x)$. If $\mu_\Phi = 0$. then function Φ is said to be convex.

Convex conjugate and proximal mapping. The proximal mapping of a convex function Ψ at the point $x \in \mathbf{R}^N$ is defined as follows:

$$\mathrm{prox}_\Psi(x) := \arg\min_{y\in\mathbf{R}^N} \Psi(y) + \tfrac{1}{2}\|y - x\|^2. \tag{3}$$

Furthermore, for a convex function $\Phi : \mathbf{R}^N \to \mathbf{R}$, it's convex conjugate Φ^* is defined as $\Phi^*(y) := \sup_{u\in\mathbb{R}^N} \langle u, y\rangle - \Phi(u)$. From Chapter 1 of [1], we see that $\mathrm{prox}_\Psi(\cdot)$, and it's convex conjugate $\mathrm{prox}_{\Psi^*}(\cdot)$, are nonexpansive:

$$\|\mathrm{prox}_\Psi(y) - \mathrm{prox}_\Psi(x)\| \leq \|y - x\|, \quad \text{and} \quad \|\mathrm{prox}_{\Psi^*}(y) - \mathrm{prox}_{\Psi^*}(x)\| \leq \|y - x\|. \tag{4}$$

Coordinate decomposition of $\mathbf{R}^N$. 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 (column vectors), where $U_i \in \mathbf{R}^{N\times 1}$. 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 $x^i \in \mathbf{R}$ denotes the ith coordinate of x.

Define $[N] := \{1, \ldots, N\}$. Then we let $S \subseteq [N]$ and $U_S \in \mathbb{R}^{N\times|S|}$ be the collection of columns from matrix U that have column indices in the set S. We denote the vector

$$x^S := \sum_{i\in S} U_i^T x = U_S^T x. \tag{5}$$

Coordinate decomposition of Ψ. The function $\Psi : \mathbf{R}^N \to \mathbf{R}$ is assumed to be coordinate separable. That is, we assume that $\Psi(x)$ can be decomposed as:

$$\Psi(x) = \sum_{i=1}^N \Psi_i(x^i), \tag{6}$$

where the functions $\Psi_i : \mathbf{R} \to \mathbf{R}$ are convex. We denote by $\Psi_S : \mathbf{R}^{|S|} \to \mathbf{R}$, where $S \subseteq [N]$, the function

$$\Psi_S(x^S) = \sum_{i\in S} \Psi_i(x^i), \tag{7}$$

where $x^S \in \mathbf{R}^{|S|}$ is the collection of components from x that have indices in set S. The following relationship will be used repeatedly in this work. For $x \in \mathbf{R}^N$, index set $S \subseteq [N]$, and $t, \hat{t} \in \mathbf{R}^{|S|}$, it holds that:

$$\Psi(x + U_S t^S) - \Psi(x + U_S \hat{t}^S) \overset{(6)+(7)}{=} \Psi_S(x^S + t^S) - \Psi_S(x^S + \hat{t}^S). \tag{8}$$

Clearly, when $\hat{t}^S \equiv 0$, we have the special case $\Psi(x + U_S t^S) - \Psi(x) = \Psi_S(x^S + t^S) - \Psi_S(x^S)$.

Subset Lipschitz continuity of f. Throughout the paper we assume that the gradient of f is Lipschitz, uniformly in x, for all subsets $S \subseteq [N]$. This means that, for all $x \in \mathbf{R}^N$, $S \subseteq [N]$ and $t^S \in \mathbf{R}^{|S|}$ we have

$$\|\nabla_S f(x + U_S t^S) - \nabla_S f(x)\| \le L_S \|t^S\|, \tag{9}$$

where $\nabla_S f(x) \overset{(5)}{=} U_S^T \nabla f(x)$. An important consequence of (9) is the following standard inequality [22, p.57]:

$$f(x + U_S t^S) \le f(x) + \langle \nabla_S f(x), t^S \rangle + \tfrac{L_S}{2}\|t^S\|^2. \tag{10}$$

Radius of the Levelset. Let X_* denote the set of optimal solutions of (1), and let x_* be any member of that set. We define

$$\mathcal{R}(x) = \max_{y} \max_{x_* \in X_*} \{\|y - x_*\| \ : \ F(y) \le F(x)\}, \tag{11}$$

which is a measure of the size of the level set of F given by x. In this work we make the standard assumption that $\mathcal{R}(x_0)$ is bounded at the initial iterate x_0. We also define a scaled version of (11)

$$\mathcal{R}_w(x) = \max_{y} \max_{x_* \in X_*} \{\|y - x_*\|_w \ : \ F(y) \le F(x)\}, \tag{12}$$

where $w \in \mathbf{R}^N_{++}$ and $\|u\|_w = \left(\sum_{i=1}^N w_i u_i^2\right)^{1/2}$.

2.1 Piecewise Quadratic Model

For fixed $x \in \mathbf{R}^N$, we define a piecewise quadratic approximation of F around the point $(x+t) \in \mathbf{R}^N$ as follows: $F(x+t) \approx f(x) + Q(x;t)$, where

$$Q(x;t) := \langle \nabla f(x), t\rangle + \tfrac{1}{2}\|t\|^2_{H(x)} + \Psi(x+t) \tag{13}$$

and $H(x)$ is *any* symmetric positive definite matrix. We also define

$$Q_S(x, t^S) := \langle \nabla_S f(x), t^S\rangle + \tfrac{1}{2}\|t^S\|^2_{H^S(x)} + \Psi_S(x^S + t^S), \tag{14}$$

and $H^S(x) \in \mathbf{R}^{|S|\times|S|}$ is a square submatrix (principal minor) of $H(x)$ that corresponds to the selection of columns and rows from $H(x)$ with column and row indices in set S. Notice that $Q_S(x, t^S)$ is the quadratic model for the collection of coordinates in S.

Similarly to (8), we have the following important relationship. For $x \in \mathbf{R}^N$, index set $S \subseteq [N]$, and $t, \hat{t} \in \mathbf{R}^{|S|}$, it holds that:

$$\begin{aligned}
Q(x; U_S t^S) - Q(x; U_S \hat{t}^S) &\overset{(13)}{=} \langle \nabla f(x), U_S t^S\rangle + \tfrac{1}{2}\|U_S t^S\|^2_{H(x)} + \Psi(x + U_S t^S)\\
&\quad -\langle \nabla f(x), U_S \hat{t}^S\rangle + \tfrac{1}{2}\|U_S \hat{t}^S\|^2_{H(x)} + \Psi(x + U_S \hat{t}^S)\\
&\overset{(8)}{=} \langle \nabla_S f(x), t^S\rangle - \langle \nabla_S f(x), \hat{t}^S\rangle + \tfrac{1}{2}\|t^S\|^2_{H^S(x)} - \tfrac{1}{2}\|\hat{t}^S\|^2_{H^S(x)}\\
&\quad +\Psi_S(x^S + t^S) - \Psi_S(x^S + \hat{t}^S)\\
&\overset{(14)}{=} Q_S(x, t^S) - Q_S(x, \hat{t}^S)
\end{aligned} \tag{15}$$

2.2 Stationarity conditions.

The following theorem gives the equivalence of some stationarity conditions of problem (1).

Theorem 3. *The following are equivalent first order optimality conditions for problem* (1)*.*

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

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

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

(iv) $x = \operatorname{prox}_{\Psi}(x - \nabla f(x))$*.*

Let us define the continuous function

$$g(x;t) := \nabla f(x) + H(x)t + \operatorname{prox}_{\Psi^*}\big(x + t - \nabla f(x) - H(x)t\big). \tag{16}$$

By Theorem 3, the points that satisfy $g(x;0) = \nabla f(x) + \operatorname{prox}_{\Psi^*}(x - \nabla f(x)) = 0$ are stationary points for problem (1). Hence, $g(x;0)$ is a continuous measure of the distance from the set of stationary points of problem (1). Furthermore, we define

$$g_S(x;t^S) := \nabla_S f(x) + H^S(x)t^S + \operatorname{prox}_{\Psi^*_S}\big(x^S + t^S - \nabla_S f(x) - H^S(x)t^S\big). \tag{17}$$

3 The Algorithm

In this section we present the Flexible Coordinate Descent (FCD) algorithm for solving problems of the form (1). There are three key steps in the algorithm: (Step 4) the coordinates are sampled randomly; (Step 5) the model (14) is solved approximately until rigorously defined stopping conditions are satisfied; (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.

In the algorithm description, and later in the paper, we will use the following definition for the set Ω:

$$\Omega := \partial Q_S(x_k; t_k^S). \tag{18}$$

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

Algorithm 1 Flexible Coordinate Descent (FCD)

```
1: **Input** Choose $x_0 \in \mathbf{R}^N$, $\theta \in (0,1)$ and $1 \le \tau \le N$.
2: **for** $k = 1, 2, \dots$ **do**
3:     Sample a subset of coordinates $S \subseteq [N]$, of size $|S| = \tau$, with probability $\mathbf{P}(S) > 0$
4:     If $g_S(x_k; 0) = 0$ go to Step 3; else select $\eta_k^S \in [0,1)$ and approximately solve
           $$t_k^S \approx \arg\min_{t^S} Q_S(x_k; t^S), \tag{19}$$
       until the following stopping conditions hold:
           $$Q(x_k; U_S t_k^S) < Q(x_k; 0), \tag{20}$$
       and
           $$\inf_{u \in \Omega} \|u - g_S(x_k; t_k^S)\|^2 \le (\eta_k^S \|g_S(x_k; 0)\|)^2 - \|g_S(x_k; t_k^S)\|^2. \tag{21}$$
5:     Perform a backtracking line search along the direction $t_k^S$ starting from $\alpha = 1$. i.e., find
       $\alpha \in (0,1]$ such that
           $$F(x_k) - F(x_k + \alpha U_S t_k^S) \ge \theta \left(\ell(x_k; 0) - \ell(x_k; \alpha U_S t_k^S)\right), \tag{22}$$
       where
           $$\ell(x_k; t) := f(x_k) + \langle \nabla f(x_k), t \rangle + \Psi(x_k + t). \tag{23}$$
6:     Update $x_{k+1} = x_k + \alpha U_S t_k^S$
7: **end for**
```

3.1 Selection of coordinates (Step 3)

One of the crucial ideas of this algorithm is that the selection of coordinates to be updated at each iteration of FCD is chosen *randomly.* This allows the coordinates to be selected very quickly and cheaply. In this section we explain in detail, how the coordinates are selected/sampled at each iteration.

We use $2^{[N]}$ to denote the set of all subsets of $[N]$, which includes the empty set $\emptyset$ and $[N]$ itself. At every iteration of the proposed method we sample a subset from $2^{[N]}$. We denote the probability of a set $S \in 2^{[N]}$ being selected by $\mathbf{P}(S)$. We are interested in probability distributions with $\mathbf{P}(S) > 0 \ \forall S \in 2^{[N]} \backslash \emptyset$ and $\mathbf{P}(\emptyset) = 0$. Therefore, all non-empty sets have positive probability of being selected. This is necessary to ensure that a subset of indices in $[N]$ will be selected at every iteration, otherwise convergence of FCD cannot not be guaranteed.

In what follows we describe a uniform probability distribution for the subsets $\mathbf{P}(S) \ \forall S \in 2^{[N]} \backslash \emptyset$. All the theoretical results in this paper use (a specific instance of) uniform probabilities.

3.1.1 Construction of probability distribution

We construct a probability distribution such that subsets in $2^{[N]}$ with the same cardinality have equal probability of being selected. Below we formalize this construction.

Assumption 4. $\mathbf{P}(S) = \mathbf{P}(S') > 0 \ \forall S, S' \in 2^{[N]}$ with $|S| = |S'|$.

This construction of uniform probabilities has been proposed in [30] for a randomized coordinate descent method. In this subsection we present it again for completeness. Let

$$S_j := \Big\{ \bigcup_{S \in 2^{[N]} : |S| = j} S \Big\}$$

be the set of all subsets $S \in 2^{[N]}$ with cardinality $|S| = j$. The probability of a subset S with cardinality j being selected is denoted with $P(S_j)$. By Assumption 4 we have that $P(S) := q_j > 0 \ \forall S \in 2^{[N]}$ with $|S| = j$. Therefore we have that

$$P(S_j) = \sum_{S \in 2^{[N]} : |S| = j} P(S) = q_j \binom{N}{j}.$$

Using the previous we obtain that the probability distribution $P(S)$ depends uniquely on $P(S_{|S|})$:

$$P(S) = \frac{P(S_{|S|})}{\binom{N}{|S|}} \quad \forall S \in 2^{[N]}. \tag{24}$$

Therefore, by defining the probability distribution $P(S_j) \ \forall j = 1, 2, \dots, N$ we obtain the probability distribution for all $S \in 2^{[N]}$.

3.1.2 Sampling sets with fixed cardinality

In this section we describe a special case of the probability distribution $P(S_j) \ \forall j = 1, 2, \dots, N$ in (24). We have the following definition.

Definition 5. Given an integer $1 \le \tau \le N$, a set S with $|S| = \tau$ is selected with probability one, i.e., $P(S_\tau) = q_\tau = 1$.

The probability distribution for the subsets in $2^{[N]}$, associated with Definition 5, is

$$P(S) = \frac{1}{\binom{N}{\tau}} \ \forall S \in 2^{[N]} \text{ with } |S| = \tau, \quad P(S) = 0 \ \forall S \in 2^{[N]} \text{ with } |S| \neq \tau. \tag{25}$$

In all the theoretical result in this paper we will use the sampling described in Definition 5.[3] We state this formally in the following assumption.

Assumption 6. In Step 3 of FCD (Algorithm 1), we assume that the sampling procedure used to generate the subset of coordinates S in that given in Definition 5.

[3]Richtárik and Takáč [30] refer to this sampling scheme as 'τ-nice sampling.

Below we present a technical result which uses the probability distribution (25) and is used in the worst-case iteration complexity results of this paper. Let θ_i with $i = 1, 2, \dots, N$ be some constants, then

$$\mathbf{E}\Big[\sum_{i\in S}\theta_i\Big] = \sum_{S\in 2^{[N]}:|S|=\tau} P(S)\sum_{i\in S}\theta_i = \sum_{i=1}^{N}\theta_i \sum_{S\in 2^{[N]}:|S|=\tau, i\in S} P(S)$$
$$= \sum_{i=1}^{N}\theta_i \sum_{S\in 2^{[N]}:|S|=\tau, i\in S} \frac{1}{\binom{N}{\tau}} = \frac{1}{\binom{N}{\tau}}\sum_{i=1}^{N}\theta_i\binom{N-1}{\tau-1} = \frac{\tau}{N}\sum_{i=1}^{N}\theta_i. \tag{26}$$

There exists a simple and efficient way to simulate the selection of subsets S from distribution (25) in $\mathcal{O}(N)$ time. Before we describe the process we discuss some preliminaries.

Let v be a vector which has the components of $[N]$ as its components. For example, if $N = 3$, then $v = [1, 2, 3]$. Let $\mathcal{D}$ with $|\mathcal{D}| = N!$ be the set of all permutations of vector v. Let $\mathcal{D}_\tau := \{u \in \mathbf{R}^\tau \mid u_i = \tilde{v}_i \ \forall i = 1, 2, \dots, \tau \ \forall \tilde{v} \in \mathcal{D}\}$, which has cardinality $N!/(N-\tau)!$. The previous is obtained by noticing that each component in $\mathcal{D}_\tau$ corresponds to $(N-\tau)!$ components in $\mathcal{D}$. By ignoring ordering in set $\mathcal{D}_\tau$ we obtain the set $\tilde{\mathcal{D}}_\tau := \{u \in \mathbf{R}^\tau \mid u \in \mathcal{D}_\tau$ where ordering matters$\}$, which has cardinality $|\tilde{\mathcal{D}}_\tau| = \binom{N}{\tau}$, since if we ignore ordering then every set in $\mathcal{D}_\tau$ appears $\tau!$ times. Notice that $\tilde{\mathcal{D}}_\tau$ is the set of all subsets of $2^{[N]}$ with cardinality equal to τ. Moreover, let $h_\tau(\tilde{v}) = [\tilde{v}_1, \tilde{v}_2, \dots, \tilde{v}_\tau]$ be the function that maintains only the first τ components of the input vector.

Let us assume that we use a procedure which generates uniformly at random permutations of v; each random permutation $\tilde{v} \in \mathcal{D}$ has $1/N!$ probability to be selected. Then we use the indices in $h_\tau(\tilde{v})$ as the coordinates to work on for the current iteration in Step 4 of FCD Algorithm. It is easy to show that for all $\tilde{v} \in \mathcal{D}$, $h_\tau(\tilde{v}) \in \tilde{\mathcal{D}}_\tau$ has probability $1/\binom{N}{\tau}$ to be selected. Notice that this procedure generates subsets of indices/coordinates with cardinality τ and probability distribution (25).

Fortunately there are many algorithms that calculate uniformly at random permutations of vector v in $\mathcal{O}(N)$ time, which also have very low memory requirements. For example the Knuth shuffling algorithm in [17, p. 145].

3.2 The search direction and Hessian approximation (Step 4)

The search direction (Step 4 of FCD) is determined as follows. Given a set of coordinates S, FCD forms a model for S (14), and minimizes the model approximately until the stopping conditions (20) and (21) are satisfied, giving an 'inexact' search direction. We also describe the importance of the choice of the matrix H, which defines the model and is an approximate second order information term. Henceforth, we will use the shorthand $H_k^S \equiv H^S(x_k)$.

3.2.1 The search direction

At each iteration of FCD, the search direction (Step 4 of FCD) is found as follows. The subproblem (19), (where $Q_S(x_k; t^S)$ is defined in (14)) is approximately solved, and the search direction t_k^S is accepted when the stopping conditions (20) and (21) are satisfied, for some $\eta_k^S \in [0, 1)$. Notice that

$$Q(x; U_S t^S) - Q(x; 0) \overset{(15)}{=} Q_S(x; t^S) - Q_S(x; 0)$$
$$\overset{(14)}{=} \langle \nabla_S f(x), t^S\rangle + \tfrac{1}{2}\|t^S\|_{H^S}^2 + \Psi_S(x^S + t^S) - \Psi_S(x^S). \tag{27}$$

Hence, from (27), the stopping conditions (20) and (21) depend on subset S only, and are therefore inexpensive to verify, meaning that they are *implementable.*

Remark 7.

(i) At some iteration k, it is possible that $g_S(x_k; 0) = 0$. In this case, it is easy to verify that the optimal solution of subproblem (19) is $t_k^S = 0$. Therefore, before calculating t_k^S we check first if condition $g_S(x_k; 0) = 0$ is satisfied.

(ii) Notice that, unless at optimality (i.e., $g(x_k; 0) = 0$), there will always exist a subset S such that $g_S(x_k; 0) \neq 0$, which implies that $t_k^S \neq 0$. Hence, FCD will not stagnate.

3.2.2 The Hessian approximation

One of the most important features of this method is that the quadratic model (14) incorporates second order information in the form of a symmetric positive definite matrix H_k^S. This is key because, depending upon the choice of H_k^S, it makes the method robust. Moreover, at each iteration, the user has complete freedom over the choice of $H_k^S \succ 0$.

We now provide a few suggestions for the choice of H_k^S. (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. **Identity.** Here, $H_k^S = I$, so that *no second order information is employed by the method.*

2. **Multiple of the identity.** One could set $H_k^S = \nu I$, for some $\nu > 0$. In particular, a popular choice is $\nu = L_S$, where L_S is the Lipschitz constant defined in (9).

3. **Diagonal matrix.** When H_k^S is a diagonal matrix, H_k^S and it's inverse are inexpensive to work with. In particular, $H_k^S = \text{diag}(\nabla_S^2 f(x_k))$ captures partial curvature information, and is an effective choice in practice, particularly when $\text{diag}(\nabla^2 f(x))$ is a good approximation to $\nabla^2 f(x)$. Moreover, if f is quadratic, then $\nabla^2 f(x_k)$ is constant for all k, so $\text{diag}(\nabla^2 f(x))$ can be computed and stored at the start of the algorithm and elements can be accessed throughout the algorithm as necessary.

4. **Principal minor of the Hessian.** When $H_k^S = \nabla_S^2 f(x_k)$, second order information is incorporated into FCD, which can be very beneficial for ill-conditioned problems. However, this choice of H_k^S is usually more computationally expensive to work with. In practice, $\nabla_S^2 f(x_k)$ may be used in a matrix-free way and not explicitly stored. For example, there may be an analytic formula for performing matrix-vector products with $\nabla_S^2 f(x_k)$, or techniques from automatic differentiation could be employed, see [25, Section 7].

5. **Quasi-Newton approximation.** One may choose H_k^S to be an approximation to $\nabla_S^2 f(x_k)$ based on a limited-memory BFGS update scheme, see [25, Section 8]. This approach might be suitable in cases where the problem is ill-conditioned, but performing matrix-vector products with $\nabla_S^2 f(x_k)$ is expensive.

Remark 8. *We make the following remarks regarding the choice of H_k^S.*

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

2. *If H_k^S is chosen to be a diagonal matrix, then the quadratic model* (14) *is* separable*, so the subproblems for each coordinate in S are independent. Moreover, in some cases this gives rise to a* closed form solution *for the search direction t_k^S. (For example, if $\Psi(x) \equiv \|x\|_1$, then soft thresholding may be used.)*

An advantage of the FCD algorithm (if Option 4 is used for H_k^S) is that *all elements of the Hessian can be accessed.* This is because the set of coordinates can change at every iteration, and so too can matrix H_k^S. This makes FCD extremely *flexible* and is particularly advantageous when there are large off diagonal elements in the Hessian. The importance of incorporation of information from off-diagonal elements is demonstrated in numerical experiments in Section 9.

3.2.3 Termination criteria for model minimization

In Step 4 of FCD, the termination criteria (20) and (21) are used to determine whether an acceptable search direction has been found. For composite functions of the form (1), *both* conditions are required to ensure that t_S^k is a descent direction. Moreover, (20) is important for intuitively obvious reasons: the model should be decreased at each iteration. We will now attempt to explain/justify the use of the second termination condition (21).

The parameter η_k^S plays a key role; it determines how 'inexactly' the quadratic model (19) may be solved (or equivalently, how 'inexact' the search direction t_k^S is allowed to be). If one sets $\eta_k^S = 0$, then the model (19) is minimized exactly, leading to exact search directions. On the other hand, if $\eta_k^S > 0$ then the model is approximately minimized, leading to inexact search directions.

To obtain iteration complexity guarantees for FCD, the termination conditions must be chosen carefully. In particular, it becomes obvious in Lemma 15, that (20) and (21) are the appropriate conditions. We now proceed to show that the conditions are mathematically sound. By rearranging (21) we obtain

$$\inf_{u\in\Omega} \|u - g_S(x_k; t_k^S)\|^2 + \|g_S(x_k; t_k^S)\|^2 \leq (\eta_k^S \|g_S(x_k; 0)\|)^2. \tag{28}$$

The left hand side in (28) is a continuous function because the 'inf' operator is an orthonormal projection onto an inner product space $\Omega := \nabla_S f(x_k) + \partial\Psi_S(x_k^S + t_k^S)$, see (18), and $\|g_S(x_k; t_k^S)\|^2$ is continuous as a function of t_k^S. Furthermore, there exists a point t_k^S such that (28) is satisfied, and this is the minimizer of (19). In particular, if t_k^S is the minimizer of (19), then from Theorem 3 we have that $g_S(x_k; t_k^S) = 0 \in \Omega$. Hence, the left hand side in (28) is zero and the condition is satisfied for any $\eta^k \in [0, 1)$ and $g_S(x_k; 0)$. Since the right hand side in (28) is a non-negative constant, the left hand side is continuous and the minimizer of (19) satisfies (28), then there exists a closed ball centered at the minimizer such that within this ball (28) is satisfied. Therefore, any convergent method which can be used to minimize (19) will eventually satisfy the termination condition (28) without solving (19) exactly, unless the right hand side in (28) is zero.

We stress that this projection operation is inexpensive for the applications that we consider, e.g., when Ψ is the ℓ_1 norm or the ℓ_2 norm. (See Section 9 for concrete examples.)

3.3 The line search (Step 5)

The stopping conditions (20) and (21) ensure that t_k^S is a descent direction, but if the full step $x_k + U_S t_k^S$ is taken, a reduction in the function value (1) is not guaranteed. To this end, we include a line search step in our algorithm in order to guarantee a monotonic decrease in the function F. Essentially, the line search guarantees the sufficient decrease of F at every iteration, where sufficient decrease is measured by the loss function (23).

In particular, for fixed $\theta \in (0,1)$, we require that for some $\alpha \in (0,1]$, (22) is satisfied. (In Lemma 12 we prove the existence of such and α.) Notice that

$$\begin{aligned} \ell(x; U_S t^S) - \ell(x;0) &\overset{(23)}{=} \langle \nabla_S f(x), t^S \rangle + \Psi(x + U_S t^S) - \Psi(x) \\ &\overset{(8)}{=} \langle \nabla_S f(x), t^S \rangle + \Psi_S(x^S + t^S) - \Psi_S(x^S), \end{aligned} \tag{29}$$

which shows that the calculation of the right hand side of (22) only depends upon block S, so it is inexpensive. Moreover, the line search condition (Step 5) involves the difference between function values $F(x_k) - F(x_k + \alpha U_S t_k^S)$. Fortunately, while function values can be expensive to compute, computing the difference between function values at successive iterates need not be. (See Section 3.4.)

3.4 Concrete examples

Now we give two examples to show that the difference between function values at successive iterates can be computed efficiently, which demonstrates that the line-search is implementable. The first example considers a convex composite function, where the smooth term is a quadratic loss term and the second example considers a logistic regression problem.

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 $x \in \mathbf{R}^N$. Then

$$\begin{aligned} F(x_k) - F(x_k + \alpha U_S t_k^S) &\overset{(8)}{=} f(x_k) + \Psi_S(x^S) - f(x_k + \alpha U_S t_k^S) - \Psi_S(x_k^S + \alpha t_k^S) \\ &= \Psi_S(x^S) - \alpha \langle \nabla_S f(x), t_k^S \rangle - \tfrac{\alpha^2}{2} \|A_i t_k^S\|_2^2 - \Psi_S(x_k^S + \alpha t_k^S). \end{aligned} \tag{30}$$

The calculation $F(x_k) - F(x_k + \alpha U_S t_k^S)$, as a function of α, only depends on subset S. Hence, it is inexpensive. Moreover, often the quantities in (30) are already needed in the computation of the search direction t, so regarding the line search step, they essentially come "for free".

Logistic regression example. Suppose that $f(x) \equiv \sum_{j=1}^m \log(1 + e^{-b_j a_j^T x})$ and $\Psi(x) \neq 0$, 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 (30). Let us split calculation of $F(x_k) - F(x_k + \alpha U_S t_k^S)$ in parts. The first part $\Psi_S(x^S) - \Psi_S(x_k^S + \alpha t_k^S)$ is inexpensive, since it depends only on subset S. The second part $f(x_k) - f(x_k + \alpha U_S t_k^S)$ 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 $f(x_k + \alpha U_S t_k^S)$ $\forall k \geq 1$ less expensively. In particular, let us assume that the following terms:

$$e^{-b_j a_j^T x_0} \quad \forall j \quad \text{and} \quad f(x_0) = \sum_{j=1}^m \log(1 + e^{-b_j a_j^T x_0}), \tag{31}$$

are calculated once and stored in memory. Then, at iteration 1, the calculation of $f(x_0 + \alpha U_S t_0^S) = \sum_{j=1}^m \log(1 + e^{-b_j a_j^T x_0} e^{-\alpha b_j a_j^T (U_S t_0^S)})$ is required for different values of α by the backtracking line search algorithm. The most demanding task in calculating $f(x_0 + \alpha U_S t_0^S)$ is the calculation of the products $b_j a_j^T (U_S t_0^S)$ $\forall j$ *once*, which is inexpensive since $\forall j$ this operation depends only on subset S. Having $b_j a_j^T (U_S t_0^S)$ $\forall j$ and (31) calculation of $f(x_0) - f(x_0 + \alpha U_S t_0^S)$ for different values of α is inexpensive. At the end of the process, $f(x_1)$ and $e^{-b_j a_j^T x_1}$ $\forall j$ are given for free, and the process can be repeated for the calculation of $f(x_1) - f(x_1 + \alpha U_S t_1^S)$ etc.

4 Bounded step-size and monotonic decrease of F

Throughout this section we denote $H_k^S \equiv H^S(x_k)$. The following assumptions are made about H_k^S and f. Assumption 9 explains that the Hessian approximation H_k^S must be positive definite for all subsets of coordinates S at all iterations k. Assumption 10 explains that the gradient of f must be Lipschitz for all subsets S.

Assumption 9. There exist $0 < \lambda_S \le \Lambda_S$, such that the sequence of symmetric $\{H_k^S\}_{k\ge 0}$ satisfies:

$$0 < \lambda_S \le \lambda_{\min}(H_k^S) \quad \text{and} \quad \lambda_{\max}(H_k^S) \le \Lambda_S, \quad \text{for all } S \subseteq [N]. \tag{32}$$

Assumption 10. The function f is smooth and satisfies (9) for all $S \subseteq [N]$.

The next assumption regards the relationship between the parameters introduced in Assumptions 9 and 10.

Assumption 11. The relation $\lambda_S \le L_S$ holds for all $S \subseteq [N]$.

The following lemma shows that if t_k^S is nonzero, then F is decreased. The proof closely follows that of [3, Theorem 3.1].

Lemma 12. *Let Assumptions 6, 9, 10 and 11 hold, and let $\theta \in (0,1)$. Given x_k, let S and $t_k^S \ne 0$ be generated by FCD (Algorithm 1) with $\eta_k^S \in [0,1)$. Then Step 6 of FCD will accept a step-size α that satisfies*

$$\alpha \ge \alpha_S, \quad \text{where} \quad \alpha_S := (1-\theta)\tfrac{\lambda_S}{2L_S} > 0. \tag{33}$$

Furthermore,

$$F(x_k) - F(x_k + \alpha U_S t_k^S) > \theta(1-\theta)\tfrac{\lambda_S^2}{4L_S}\|t_k^S\|^2. \tag{34}$$

Proof. From (20), we have $0 > Q(x_k; U_S t_k^S) - Q(x_k;0) \overset{(13)+(23)}{=} \ell(x_k; U_S t_k^S) - \ell(x_k;0) + \frac{1}{2}\|t_k^S\|^2_{H_k^S}$. Rearranging gives

$$\ell(x_k;0) - \ell(x_k; U_S t_k^S) > \tfrac{1}{2}\|t_k^S\|^2_{H_k^S} \overset{(32)}{\ge} \tfrac{1}{2}\lambda_S\|t_k^S\|^2. \tag{35}$$

Denote $x_{k+1} = x_k + \alpha U_S t_k^S$. By Assumption 10, for all $\alpha \in (0,1]$, we have

$$F(x_{k+1}) \le f(x_k) + \alpha\langle \nabla_S f(x_k), t_k^S\rangle + \tfrac{L_S}{2}\alpha^2\|t_k^S\|^2 + \Psi(x_k + \alpha U_S t_k^S).$$

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

$$\begin{aligned} F(x_k) - F(x_{k+1}) &\ge -\alpha\langle \nabla_S f(x_k), t_k^S\rangle - \tfrac{L_S}{2}\alpha^2\|t_k^S\|^2 - \Psi(x_k + \alpha U_S t_k^S) + \Psi(x_k) \\ &\overset{(29)}{=} \ell(x_k;0) - \ell(x_k; \alpha U_S t_k^S) - \tfrac{L_S}{2}\alpha^2\|t_k^S\|^2. \end{aligned} \tag{36}$$

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

$$\ell(x_k;0) - \ell(x_k;\alpha U_S t_k^S) \ge \alpha(\ell(x_k;0) - \ell(x_k; U_S t_k^S)). \tag{37}$$

Then

$$\begin{aligned} F(x_k) - F(x_{k+1}) &- \theta(\ell(x_k;0) - \ell(x_k;\alpha U_S t_k^S)) \\ &\overset{(36)}{\ge} (1-\theta)\big(\ell(x_k;0) - \ell(x_k;\alpha U_S t^S)\big) - \tfrac{L_S}{2}\alpha^2\|t_k^S\|^2 \\ &\overset{(37)}{\ge} \alpha(1-\theta)\big(\ell(x_k;0) - \ell(x_k;U_S t_k^S)\big) - \tfrac{L_S}{2}\alpha^2\|t_k^S\|^2 \\ &\overset{(35)}{>} \tfrac{1}{2}\big(\alpha(1-\theta)\lambda_S\|t_k^S\|^2 - L_S\alpha^2\|t_k^S\|^2\big) \\ &= \tfrac{\alpha}{2}\big((1-\theta)\lambda_S - L_S\alpha\big)\|t_k^S\|^2 \ge 0, \end{aligned} \tag{38}$$

if $(1-\theta)\lambda_S - L_S\alpha \geq 0$. Therefore, we observe that if α satisfies $0 < \alpha \leq (1-\theta)\frac{\lambda_S}{L_S}$, then α also satisfies the backtracking line search step of FCD (i.e., a function value reduction is achieved). 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 (33).

We now show that (34) holds. By the previous arguments, the line search condition (22) is guaranteed to be satisfied for some step size α. Then,

$$\begin{aligned} F(x_k) - F(x_{k+1}) &\overset{(22)+(37)}{\geq} \theta\alpha(\ell(x_k;0) - \ell(x_k;U_S t_k^S)) \\ &\overset{(35)}{>} \tfrac{\alpha\theta\lambda_S}{2}\|t_k^S\|^2 \geq \tfrac{\alpha_S\theta\lambda_S}{2}\|t_k^S\|^2 \end{aligned} \tag{39}$$

Using the definition of α_S in (33) gives the result. □

The following lemma bounds the norm of the direction t_k^S in terms of $g_S(x_k,0)$. This proof closely follows that of [3, Theorem 3.1].

Lemma 13. *Let Assumptions 6, 9, 10 and 11 hold, and let $\theta \in (0,1)$. Given x_k, let S, $t_k^S \neq 0$ and α be generated by FCD (Algorithm 1) with $\eta_k^S \in [0,1)$. Then*

$$\|t_k^S\| \geq \gamma_S\|g_S(x_k;0)\|, \quad \textit{where} \quad \gamma_s := \tfrac{1-\eta_k^S}{1+2\Lambda_S}. \tag{40}$$

Moreover,

$$F(x_k) - F(x_k + \alpha U_S t_k^S) > \theta(1-\theta)\tfrac{\lambda_S^2}{4L_S}\gamma_S^2\|g_S(x_k;0)\|^2. \tag{41}$$

Proof. From the termination condition (21) we have that $\|g_S(x_k;t_k^S)\| \leq \eta_k^S\|g_S(x_k;0)\|$. Using the reverse triangular inequality and the fact that $\text{prox}_{\Psi_S^*}(\cdot)$ is nonexpansive, we have that

$$\begin{aligned} (1-\eta_k^S)\|g_S(x_k;0)\| &\leq \|g_S(x_k;0)\| - \|g_S(x_k;t_k^S)\| \\ &\leq \|g_S(x_k;t_k^S) - g_S(x_k;0)\| \\ &\overset{(17)}{=} \|H_k^S t_k^S + \text{prox}_{\Psi_S^*}\big(x_k^S + t_k^S - (\nabla_S f(x_k) + H_k^S t_k^S)\big) \\ &\quad -\text{prox}_{\Psi_S^*}\big(x_k^S - \nabla_S f(x_k)\big)\| \\ &\overset{(4)}{\leq} \|H_k^S t_k^S\| + \|(I - H_k^S)t_k^S\| \\ &\leq (1 + 2\|H_k^S\|)\|t_k^S\| \\ &\overset{(32)}{\leq} (1 + 2\Lambda_S)\|t_k^S\|. \end{aligned}$$

Rearranging gives (40), and combining (40) with (34) gives (41). □

5 Technical results

In this section we will present several technical results that will be necessary when establishing our main iteration complexity results for FCD. The first result (Theorem 14) is taken from [29] and will be used to finish all our iteration complexity results for FCD. The second result (Lemma 15) provides an upper bound on the decrease in the model that is obtained when an inexact search direction is used in FCD. The final two results provide upper bounds on the difference of function values $F(x_{k+1}) - F(x_k)$ (Lemma 16) and the expected distance to the solution $\mathbf{E}[F(x_{k+1}) - F^*]$ (Lemma 18).

The following Theorem will be used at the end of all our proofs to obtain iteration complexity results for FCD. (It will be used with $\xi_k = F(x_k) - F^*$.)

Theorem 14 (Theorem 1 in [29]). *Fix $\xi_0 \in \mathbf{R}^N$ and let $\{\xi_k\}_{k\geq 0}$ be a sequence of random vectors in $\mathbf{R}^N$ with ξ_{k+1} depending on ξ_k only. Let $\{\xi_k\}_{k\geq 0}$ be a nonnegative non increasing sequence of random variables and assume that it has one of the following properties:*

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

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

Choose accuracy level $0 < \epsilon < \xi_0$, confidence level $\rho \in (0,1)$. If (i) holds and we choose $\epsilon < c_1$ and $K \geq \frac{c_1}{\epsilon}\left(1 + \ln\frac{1}{\rho}\right) + 2 - \frac{c_1}{\xi_0}$, or if property (ii) holds and we choose $K \geq c_2 \ln\frac{\xi_0}{\epsilon\rho}$, then $\mathbf{P}(\xi_K \leq \epsilon) \geq 1 - \rho$.

The following result gives an upper bound on the decrease in the quadratic model when an inexact update is used in FCD.

Lemma 15. *Let Assumptions 6, 9, and 10 hold, let $\theta \in (0,1)$, and let t_*^S denote the exact minimizer of subproblem* (19)*. Given x_k, let S, $t_k^S \neq 0$ and α be generated by FCD (Algorithm 1) with $\eta_k^S \in [0,1)$, and let $x_{k+1} = x_k + \alpha U_S t_k^S$. Then*

$$Q(x_k; U_S t_k^S) - Q(x_k; 0) \leq Q(x_k; U_S t_*^S) - Q(x_k; 0) + \frac{2(\eta_k^S)^2 L_S}{\theta(1-\theta)\lambda_S^3\gamma_S^2}(F(x_k) - F(x_{k+1})),$$

where η_k^S, θ and γ_S are defined in (21), (22) *and* (40)*, respectively.*

Proof. We have that

$$Q(x_k; U_S t_k^S) - Q(x_k; 0) = \left(Q(x_k; U_S t_k^S) - Q(x_k; U_S t_*^S)\right) + \left(Q(x_k; U_S t_*^S) - Q(x_k; 0)\right). \quad (42)$$

The term $Q(x_k; U_S t_k^S) - Q(x_k; U_S t_*^S)$ can be bounded using strong convexity of Q_S (see [2, p.460]). That is, for every $u \in \partial Q_S(x_k; t_k^S)$:

$$\begin{aligned} Q(x_k; U_S t_k^S) - Q(x_k; U_S t_*^S) &\overset{(15)}{=} Q_S(x_k; t_k^S) - Q_S(x_k; t_*^S) \leq \tfrac{1}{2\lambda_S}\|u\|^2 \\ &\leq \tfrac{1}{2\lambda_S}\|u + g_S(x_k; t_k^S) - g_S(x_k; t_k^S)\|^2 \\ &\leq \tfrac{1}{2\lambda_S}\Big(\|u - g_S(x_k; t_k^S)\|^2 + \|g_S(x_k; t_k^S)\|^2\Big). \end{aligned} \quad (43)$$

Therefore, using termination condition (21), where Ω is defined in (18), we have that

$$\begin{aligned} Q(x_k; U_S t_k^S) - Q(x_k; U_S t_*^S) &\leq \tfrac{1}{2\lambda_S}\Big(\inf_{u\in\Omega}\|u - g_S(x_k; t_k^S)\|^2 + \|g_S(x_k; t_k^S)\|^2\Big) \\ &\leq \tfrac{1}{2\lambda_S}(\eta_k^S\|g_S(x_k; 0)\|)^2. \end{aligned}$$

Using Lemma 13 in the previous inequality, followed by (42), gives the result. □

In the next Lemma we establish an upper bound on the difference between consecutive function values obtained using FCD.

Lemma 16. *Let Assumptions 6, 9, 10 and 11 hold. Let $\theta \in (0,1)$, fix $\tilde{\eta} \in [0,1)$, and let t_*^S denote the exact minimizer of subproblem* (19)*. Given x_k, let S, $t_k^S \neq 0$ and α be generated by FCD (Algorithm 1) with $\eta_k^S \in [0,\tilde{\eta}]$, and let $x_{k+1} = x_k + \alpha U_S t_k^S$. Then*

$$F(x_{k+1}) - F(x_k) \leq \chi(\tilde{\eta})\big(Q(x_k; U_S t_*^S) - Q(x_k; 0)\big), \quad (44)$$

where

$$\chi(\tilde{\eta}) = \min_{S\in 2^{[N]}\,:\,|S|=\tau} \frac{\theta(1-\theta)\lambda_S^3\gamma_S^2}{2L_S\Big(\tilde{\eta}^2 + \lambda_S\gamma_S^2(L_S - (1-\theta)\lambda_S)\Big)}. \quad (45)$$

Proof. Using block Lipschitz continuity of f (10), and the definition of Q (13) we get

$$\begin{aligned} F(x_{k+1}) &\leq F(x_k) + (Q(x_k; \alpha U_S t_k^S) - Q(x_k; 0)) - \tfrac{\alpha^2}{2}\|t_k^S\|_{H_k^S}^2 + \tfrac{\alpha^2 L_S}{2}\|t_k^S\|^2 \\ &\overset{(32)}{\leq} F(x_k) + (Q(x_k; \alpha U_S t_k^S) - Q(x_k; 0)) + \tfrac{\alpha^2}{2}\left(\tfrac{L_S}{\lambda_S} - 1\right)\|t_k^S\|_{H_k^S}^2. \end{aligned}$$

Using convexity of Q with respect to α gives

$$\begin{aligned} F(x_{k+1}) &\leq F(x_k) + \alpha(Q(x_k; U_S t_k^S) - Q(x_k; 0)) + \tfrac{\alpha^2}{2}\left(\tfrac{L_S}{\lambda_S} - 1\right)\|t_k^S\|_{H_k^S}^2 \\ &\leq F(x_k) + \alpha_S(Q(x_k; U_S t_k^S) - Q(x_k; 0)) + \tfrac{\alpha^2}{2}\left(\tfrac{L_S}{\lambda_S} - 1\right)\|t_k^S\|_{H_k^S}^2, \end{aligned} \tag{46}$$

where the second inequality holds because $Q(x_k; U_S t_k^S) - Q(x_k; 0) < 0$ (20), and $\alpha \geq \alpha_S$ (33). By rearranging (39) we get $\frac{\alpha}{2}\|t_k^S\|_{H_k^S}^2 \leq \frac{1}{\theta}(F(x_k) - F(x_{k+1}))$. Using this in (46) gives

$$F(x_{k+1}) \leq F(x_k) + \alpha_S(Q(x_k; U_S t_k^S) - Q(x_k; 0)) + \tfrac{\alpha}{\theta}\left(\tfrac{L_S}{\lambda_S} - 1\right)(F(x_k) - F(x_{k+1}))$$

By Assumption 11, $L_S \geq \lambda_S$, so we can replace $\alpha \in (0, 1]$ with 1 to get

$$F(x_{k+1}) \leq F(x_k) + \alpha_S(Q(x_k; U_S t_k^S) - Q(x_k; 0)) + \tfrac{1}{\theta}\left(\tfrac{L_S}{\lambda_S} - 1\right)(F(x_k) - F(x_{k+1})). \tag{47}$$

Using Lemma 15 and the definition of $\tilde{\eta}$ we get

$$\begin{aligned} F(x_{k+1}) &\leq F(x_k) + \alpha_S(Q(x_k; U_S t_*^S) - Q(x_k; 0)) + \tfrac{1}{\theta}\left(\tfrac{2\alpha_S \tilde{\eta}^2 L_S}{(1-\theta)\lambda_S^3 \gamma_S^2} + \tfrac{L_S}{\lambda_S} - 1\right)(F(x_k) - F(x_{k+1})) \\ &\overset{(33)}{\leq} F(x_k) + \alpha_S(Q(x_k; U_S t_*^S) - Q(x_k; 0)) + \tfrac{1}{\theta}\left(\tfrac{\tilde{\eta}^2}{\lambda_S^2 \gamma_S^2} + \tfrac{L_S}{\lambda_S} - 1\right)(F(x_k) - F(x_{k+1})). \end{aligned}$$

Rearranging the previous, using the definitions of $\chi(\tilde{\eta})$ and α_S, gives the result. □

Remark 17. *Notice that when exact directions are used in FCD (i.e., $\tilde{\eta} = 0$), we have*

$$\chi(0) = \min_{S \in 2^{[N]} \,:\, |S| = \tau} \frac{\theta(1-\theta)\lambda_S^2}{2L_S^2 - 2(1-\theta)\lambda_S L_S} \tag{48}$$

i.e., $\chi(\tilde{\eta})$ depends cubically on λ_S (45), whereas if exact directions are used, $\chi(0)$ only depends upon λ_S^2. However, if $L_S = \lambda_S$, but inexact directions are used ($\tilde{\eta} \neq 0$), then

$$\chi(\tilde{\eta}) = \min_{S \in 2^{[N]} \,:\, |S| = \tau} \frac{1-\theta}{2} \frac{\theta \lambda_S^2 \gamma_S^2}{\tilde{\eta}^2 + \theta \lambda_S^2 \gamma_S^2}, \tag{49}$$

which again shows a dependence upon λ_S^2. Moreover, if $L_S = \lambda_S$, then $\chi(0) = (1-\theta)/2$ (i.e., constant). This is summarized in the following table.

	Inexact Directions ($\tilde{\eta} \neq 0$)	Exact Directions ($\tilde{\eta} = 0$)
$\lambda_S \neq L_S$	cubic	squared
$\lambda_S = L_S$	squared	constant

Table 1: Dependence of $\chi(\tilde{\eta})$ (45) upon λ_S (Assumption 9).

The final result in this section gives an upper bound on the expected distance between the current function value and the optimal value.

Lemma 18. *Let Assumptions 6, 9, 10 and 11 hold. Let $\theta \in (0,1)$, fix $\tilde{\eta} \in [0,1)$, and let $\mu_f \geq 0$ be the strong convexity constant of f. Given x_k, let S, $t_k^S \neq 0$ and α be generated by FCD (Algorithm 1) with $\eta_k^S \in [0, \tilde{\eta}]$, and let $x_{k+1} = x_k + \alpha U_S t_k^S$. Then*

$$\mathbf{E}[F(x_{k+1}) - F^*|x_k] \leq \left(1 - \tfrac{\tau\chi(\tilde{\eta})\zeta}{N}\right)(F(x_k) - F^*) - (\mu_f - \Lambda_{\max}\zeta)\tfrac{\tau\chi(\tilde{\eta})\zeta}{2N}\|x_k - x_*\|^2, \tag{50}$$

where $\zeta \in [0,1]$, $\chi(\tilde{\eta})$ is defined in (45) *and*

$$\Lambda_{\max} := \max_{S \in 2^{[N]} \,:\, |S| = \tau} \Lambda_S. \tag{51}$$

Proof. The assumptions of Lemma 16 are satisfied, so (44) holds. Notice that t_*^S is the vector that makes the difference $Q(x_k; U_S t_*^S) - Q(x_k; 0)$, as small/negative as possible over the subspace defined by the columns of U_S. Therefore, for any vector $y_k^S \neq t_*^S$, $Q(x_k; U_S t_*^S) - Q(x_k; 0) \leq Q(x_k; U_S y_k^S) - Q(x_k; 0)$. Choosing $y_k^S = \zeta(x_*^S - x_k^S)$ for any $\zeta \in [0,1]$ gives

$$\begin{aligned}
F(x_{k+1}) - F(x_k) &\leq \chi(\tilde{\eta})(Q(x_k; \zeta U_S(x_*^S - x_k^S)) - Q(x_k; 0)) \\
&\overset{(13)}{\leq} \chi(\tilde{\eta})\Big(\zeta\langle \nabla f(x_k), U_S(x_*^S - x_k^S)\rangle + \frac{\zeta^2}{2}\|x_*^S - x_k^S\|_{H_k^S}^2 \\
&+ \Psi_S(x_k^S + \zeta(x_*^S - x_k^S)) - \Psi_S(x_k^S)\Big).
\end{aligned}$$

Using convexity of Ψ, (32) and the definition $\Lambda_{\max}$ (51) we get

$$\begin{aligned}
F(x_{k+1}) - F(x_k) \leq \chi(\tilde{\eta})\zeta\Big(\langle \nabla f(x_k), U_S(x_*^S - x_k^S)\rangle + \Psi_S(x_*^S) - \Psi_S(x_k^S)\Big) \\
+ \tfrac{\chi(\tilde{\eta})\Lambda_{\max}\zeta^2}{2}\|x_*^S - x_k^S\|^2.
\end{aligned} \tag{52}$$

Taking expectation of (52) and using convexity of f gives

$$\begin{aligned}
\mathbf{E}[F(x_{k+1}) - F(x_k)|x_k] &\overset{(26)}{\leq} \tfrac{\tau\chi(\tilde{\eta})\zeta}{N}\Big(\langle \nabla f(x_k), x_* - x_k\rangle + \Psi(x_*) - \Psi(x_k)\Big) \\
&+ \tfrac{\tau\chi\Lambda_{\max}}{N}\tfrac{\zeta^2}{2}\|x_k - x_*\|^2 \\
&\leq -\tfrac{\tau\chi(\tilde{\eta})\zeta}{N}(F(x_k) - F^*) - \tfrac{\tau\chi(\tilde{\eta})\zeta}{N}\tfrac{\mu_f}{2}\|x_k - x_*\|_2^2 \\
&+ \tfrac{\tau\chi(\tilde{\eta})\Lambda_{\max}}{N}\tfrac{\zeta^2}{2}\|x_k - x_*\|^2.
\end{aligned}$$

Rearranging and subtracting F^* from both sides gives the result. □

6 Iteration complexity results for inexact FCD

In this section we present iteration complexity results for FCD applied to problem (1) when an inexact update is used.

6.1 Iteration complexity in the convex case

Here we present iteration complexity results for FCD in the convex case.

Theorem 19. *Let F be the convex function defined in* (1) *and let Assumptions 6, 9, 10 and 11 hold. Choose an initial point $x_0 \in \mathbf{R}^N$, a target confidence $\rho \in (0,1)$, fix $\tilde{\eta} \in [0,1)$, and recall that $\chi(\tilde{\eta})$ is defined in* (45)*. Let the target accuracy ϵ and iteration counter K be chosen in either of the following two ways:*

(i) Let $m_1 := \max\{\mathcal{R}^2(x_0), F(x_0) - F^*\}$, $0 < \epsilon < F(x_0) - F^*$ *and*

$$K \geq \frac{2N}{\tau\chi(\tilde{\eta})} \frac{m_1}{\epsilon}\left(1 + \ln\frac{1}{\rho}\right) + 2 - \frac{2N}{\tau\chi(\tilde{\eta})}\frac{m_1}{(F(x_0) - F^*)}, \tag{53}$$

(ii) Let $0 < \epsilon < \min\{\mathcal{R}^2(x_0), F(x_0) - F^*\}$ *and*

$$K \geq \frac{2N}{\tau\chi(\tilde{\eta})}\frac{\mathcal{R}^2(x_0)}{\epsilon}\ln\frac{F(x_0) - F^*}{\epsilon\rho}. \tag{54}$$

If $\{x_k\}_{k\geq 0}$ *are the random points generated by FCD (Algorithm 1), with* $\eta_k^S \in [0, \tilde{\eta}]$ *for all* k *and all* $S \in 2^{[N]}$*, then* $\mathbf{P}(F(x_K) - F^* \leq \epsilon) \geq 1 - \rho$.

Proof. By Lemma 12 we have that $F(x_k) \leq F(x_0)$ for all k, so $\|x_k - x_*\| \leq \mathcal{R}(x_0)$ for all $x_* \in X^*$, where $\mathcal{R}(x_0)$ is defined in (11). Then, using Lemma 18 with $\mu_f = 0$ gives

$$\mathbf{E}[F(x_{k+1}) - F^*|x_k] \leq \left(1 - \tfrac{\tau\chi(\tilde{\eta})\zeta}{N}\right)(F(x_k) - F^*) + \tfrac{\tau\chi(\tilde{\eta})\Lambda_{\max}\zeta^2}{2N}\mathcal{R}^2(x_0). \tag{55}$$

Minimizing the right hand side of the previous with respect to ζ gives: $\zeta = \min\left\{\frac{F(x_k) - F^*}{\Lambda_{\max}\mathcal{R}^2(x_0)}, 1\right\}$. Then, (55) becomes

$$\mathbf{E}[F(x_{k+1}) - F^*|x_k] \leq \begin{cases} \left(1 - \frac{\tau\chi(\tilde{\eta})}{2N}\frac{F(x_k) - F^*}{\Lambda_{\max}\mathcal{R}^2(x_0)}\right)(F(x_k) - F^*) & \text{if } \frac{F(x_k) - F^*}{\Lambda_{\max}\mathcal{R}^2(x_0)} \leq 1 \\ \left(1 - \frac{\tau\chi(\tilde{\eta})}{2N}\right)(F(x_k) - F^*) & \text{otherwise,} \end{cases}$$

or alternatively,

$$\mathbf{E}[F(x_{k+1}) - F^*|x_k] \leq \left(1 - \tfrac{\tau\chi(\tilde{\eta})}{2N}\min\left\{\tfrac{F(x_k) - F^*}{\Lambda_{\max}\mathcal{R}^2(x_0)}, 1\right\}\right)(F(x_k) - F^*). \tag{56}$$

It is enough to study (56) under the condition $F(x_k) - F^* \leq \Lambda_{\max}\mathcal{R}^2(x_0)$.[4] Hence, we have the following two cases.

(i) Letting $c_1 = \frac{2N}{\tau\chi(\tilde{\eta})}\max\{\Lambda_{\max}\mathcal{R}^2(x_0), F(x_0) - F^*\}$, (56) becomes

$$\begin{aligned} \mathbf{E}[F(x_{k+1}) - F^*|x_k] &\leq \left(1 - \tfrac{\tau\chi(\tilde{\eta})}{2N}\tfrac{F(x_k) - F^*}{\Lambda_{\max}\mathcal{R}^2(x_0)}\right)(F(x_k) - F^*) \\ &\leq \left(1 - \tfrac{F(x_k) - F^*}{c_1}\right)(F(x_k) - F^*). \end{aligned}$$

Notice that for this choice of c_1, $\epsilon < F(x_0) - F^* < c_1$, so it suffices to apply Theorem 14(i) and the result follows.

(ii) Now we choose $c_2 = \frac{2N}{\tau\chi(\tilde{\eta})}\frac{\mathcal{R}^2(x_0)}{\epsilon} > 1$. Observe that, whenever $\epsilon \leq F(x_k) - F^*$, by (56) we have $\mathbf{E}[F(x_{k+1}) - F^*|x_k] \leq (1 - \frac{1}{c_2})(F(x_k) - F^*)$. It remains to apply Theorem 14(ii), and the result follows.

$\square$

[4]Notice that, at any particular iteration of FCD, either of the two options inside the 'min' in (56) could be the smallest. However, $\Lambda_{\max}\mathcal{R}^2(x_0)$ is fixed throughout the iterations, and $F(x_k) - F^*$ is becoming smaller as the iterations progress by Lemma 12. Therefore, eventually it will be the case that the first term inside the 'min' in (56) will be the smallest, i.e., $F(x_k) - F^* \leq \Lambda_{\max}\mathcal{R}^2(x_0)$.

6.2 Iteration complexity in the strongly convex case

In this section we establish iteration complexity results for FCD when the objective function F, and the smooth function f, are strongly convex, with strong convexity parameters $\mu_f > 0$ and $\mu_F > 0$, respectively. Moreover, $\mu_f \le \mu_F$.

Before presenting our iteration complexity results, we make the following assumption. It explains that at least one of the chosen matrices $\{H_k^S\}_{k\ge 0}$ has an eigenvalue greater than μ_f.

Assumption 20. We assume that $\Lambda_{\max} \ge \mu_f > 0$, where $\Lambda_{\max}$ is defined in (51).

We also define the following variable, which appears in the complexity results:

$$\delta := \begin{cases} \frac{\mu_f+\mu_F}{4\Lambda_{\max}}\left(1+\frac{\mu_f}{\mu_F}\right) & \text{if } \mu_F+\mu_f < 2\Lambda_{\max} \\ 1-\frac{\Lambda_{\max}-\mu_f}{\mu_F} & \text{otherwise.} \end{cases} \tag{57}$$

It is straightforward to show that $\delta \in (0,1]$. In particular, if $\mu_F + \mu_f < 2\Lambda_{\max}$, the conclusion follows because $\mu_F \ge \mu_f \Rightarrow 1 + \frac{\mu_f}{\mu_F} \le 2$. On the other hand, if $2\Lambda_{\max} \le \mu_F + \mu_f$, then by inspection, we have $(\mu_F + \mu_f - \Lambda_{\max})/\mu_F \in (0,1]$.

Theorem 21. *Let f and F be the strongly convex functions defined in* (1) *with $\mu_f > 0$ and $\mu_F > 0$ respectively, and let Assumptions 6, 9, 10, 11 and 20 hold. Choose an initial point $x_0 \in \mathbf{R}^N$, a target confidence $\rho \in (0,1)$, a target accuracy $\epsilon > 0$, and fix $\tilde\eta \in [0,1)$. Let $\chi(\tilde\eta)$ and δ be defined in* (45) *and* (57) *respectively, and let*

$$K \ge \frac{N}{\tau\chi(\tilde\eta)\delta}\ln\left(\frac{F(x_0)-F^*}{\epsilon\rho}\right). \tag{58}$$

If $\{x_k\}_{k\ge0}$ are the random points generated by FCD (Algorithm 1), with $\eta_k^S \in [0,\tilde\eta]$ for all k and all $S \in 2^{[N]}$, then $\mathbf{P}(F(x_K) - F^* \le \epsilon) \ge 1-\rho$.

Proof. Define $\xi_k := F(x_k) - F^*$. From Lemma 18 we have that

$$\begin{aligned} \mathbf{E}[\xi_{k+1}|x_k] &\le \left(1-\tfrac{\tau\chi(\tilde\eta)\zeta}{N}\right)\xi_k + (\Lambda_{\max}\zeta-\mu_f)\tfrac{\tau\chi(\tilde\eta)\zeta}{2N}\|x_k-x_*\|^2 \\ &\overset{(2)}{\le} \left(1-\tfrac{\tau\chi(\tilde\eta)\zeta}{N}\right)\xi_k + (\Lambda_{\max}\zeta-\mu_f)\tfrac{\tau\chi(\tilde\eta)\zeta}{N\mu_F}\xi_k. \end{aligned} \tag{59}$$

Notice that (2) can only be applied in (59) when $\mu_f \le \Lambda_{\max}\zeta$. But, differentiating the right hand side of (59) with respect to ζ gives $\zeta = \min\{(\mu_F+\mu_f)/2\Lambda_{\max}, 1\}$.[5] Then

$$\mathbf{E}[\xi_{k+1}|x_k] \le \begin{cases} \left(1-\frac{\tau\chi(\tilde\eta)}{N}\frac{(\mu_F+\mu_f)}{4\Lambda_{\max}}\left(1+\frac{\mu_f}{\mu_F}\right)\right)\xi_k & \text{if } \frac{\mu_F+\mu_f}{2\Lambda_{\max}} < 1 \\ \left(1-\frac{\tau\chi(\tilde\eta)}{N}\left(1-\frac{\Lambda_{\max}-\mu_f}{\mu_F}\right)\right)\xi_k & \text{otherwise.} \end{cases}$$

Now, using (57), we can write $\mathbf{E}[\xi_{k+1}|x_k] \le \left(1-\frac{\tau\chi(\tilde\eta)\delta}{N}\right)\xi_k$. The result follows by applying Theorem 14(ii) with $c_2 = \frac{N}{\tau\chi(\tilde\eta)\delta} > 1$. ☐

[5]We could simply have set $\zeta = 1$ initially, so that $\mu_f \le \Lambda_{\max}\zeta$ is satisfied by Assumption 20. However, we obtain a better complexity result by taking a smaller ζ.

7 Iteration complexity of FCD applied to smooth functions

The iteration complexity results presented in Section 6 simplify in the smooth case. Therefore, in this section we assume that $\Psi = 0$ so that $F \equiv f$. In the smooth case, the quadratic model becomes

$$Q(x_k; U_S t_k^S) \equiv Q_S(x_k; t_k^S) = \langle \nabla_S f(x_k), t_k^S \rangle + \tfrac{1}{2}\langle H_k^S t_k^S, t_k^S \rangle. \tag{60}$$

Notice that the stationarity conditions also simplify in the following way

$$g_S(x_k; t_k^S) = \nabla_S f(x_k) + H_k^S t_k^S, \tag{61}$$

so that minimizing the (smooth) quadratic model (60), is equivalent to solving the linear system

$$H_k^S t_k^S = -\nabla_S f(x_k). \tag{62}$$

The matrix H_k^S is always symmetric and positive definite (by definition), so an obvious approach is to solve (62) using the Conjugate Gradient method (CG). It is possible to use other iterative methods to approximately minimize (60)/solve (62). However, we will see that using CG gives better iteration complexity guarantees, so, for the results presented in this section, we assume that CG is always used to solve (62).

We have the following theorem, which allows us to determine the value of the function $Q_S(x_k; U_S t_k^S)$ when an inexact direction t_k^S is used.

Lemma 22 (Lemma 7 in [15]). *Let $Az = b$, where A is a symmetric and positive definite matrix. Furthermore, let us assume that this system is solved using CG approximately; CG is terminated prematurely at the jth iteration. Then if CG is initialized with the zero solution, the approximate solution z_j satisfies $z_j^T A z_j = z_j^T b$.*

Further, note that for smooth functions, the FCD stopping condition (21) becomes

$$\frac{\|g_S(x; t_k^S)\|_2}{\|g_S(x; 0)\|_2} \equiv \frac{\|\nabla_S f(x_k) + H_k^S t_k^S\|_2}{\|\nabla_S f(x_k)\|_2} \le \eta_k^S. \tag{63}$$

Therefore, in this section we will make the following assumption.

Assumption 23. We assume that if FCD (Algorithm 1) is applied to problem (1) with $\Psi = 0$, then, for all k and all $S \in 2^{[N]}$, the inexact search direction t_k^S is generated by applying j iterations of CG to (62), initialized with the zero vector, until the stopping condition (63) is satisfied, where $\eta_k^S \in [0, 1)$.

Let t_k^S be the inexact solution obtained by applying CG to (62) starting from the zero vector, and terminating according to (63), (i.e., let Assumption 23 hold). Then, by Lemma 22

$$\langle H_k^S t_k^S, t_k^S \rangle = -\langle \nabla_S f(x_k), t_k^S \rangle. \tag{64}$$

so that

$$Q_S(x_k; t_k^S) = \tfrac{1}{2}\langle \nabla_S f(x_k), t_k^S \rangle \equiv -\tfrac{1}{2}\langle H_k^S t_k^S, t_k^S \rangle < 0. \tag{65}$$

Therefore, combining (65) with the fact that $Q(x_k; 0) = 0$ in the smooth case, the stopping condition $Q(x_k; U_S t_k^S)$ $(= Q_S(x_k; t_k^S)) < Q(x_k; 0)$ (i.e., (20)) is always satisfied.

7.1 Technical Results when $\Psi = 0$

We begin by presenting several technical results that will be needed when establishing iteration complexity results for FCD in the smooth ($\Psi = 0$) case. The first result is analogous to Lemma 13 for smooth functions, while the second is similar to Lemma 12.

Lemma 24. *Let Assumptions 6, 9, 10, 11 and 23 hold. Given x_k, let S, and $t_k^S \neq 0$ be generated by FCD (Algorithm 1), applied to problem* (1) *with $\Psi = 0$, and with $\eta_k^S \in [0,1)$. Then*

$$\tfrac{1-\eta_k^S}{\Lambda_S}\|\nabla_S f(x_k)\|_2 \leq \|t_k^S\|_2. \tag{66}$$

Proof. This proof closely follows that of [3, Theorem 3.1], and of Lemma 13. The termination condition (63) is $\|g_S(x_k;t_k^S)\|_2 \leq \eta_k^S\|g_S(x_k;0)\|_2$. Then

$$\begin{aligned}
(1-\eta_k^S)\|g_S(x_k;0)\|_2 &\leq \|g_S(x_k;0)\|_2 - \|g_S(x_k;t_k^S)\|_2 \\
&\leq \|g_S(x_k;t_k^S) - g_S(x_k;0)\|_2 \\
&\overset{(61)}{=} \|H_k^S t_k^S\|_2 \leq \Lambda_S\|t_k^S\|_2.
\end{aligned}$$

It remains to recall that $\nabla_S f(x_k) \equiv g_S(x_k;0)$. □

Lemma 25. *Let Assumptions 6, 9, 10, 11 and 23 hold, let $\theta \in (0,1/2)$, and fix $\tilde\eta \in [0,1)$.Given x_k, let S, $t_k^S \neq 0$ be generated by FCD (Algorithm 1) applied to problem* (1) *with $\Psi = 0$, and with $\eta_k^S \in [0,\tilde\eta]$. Then Step 6 of FCD will accept a step-size α that satisfies*

$$\alpha \geq \tfrac{\theta}{2}\tfrac{\lambda_S}{L_S} > 0. \tag{67}$$

Moreover,

$$f(x_k) - f(x_k + \alpha U_S t_k^S) \geq \tfrac{\vartheta(\tilde\eta)}{2}\|\nabla_S f(x_k)\|_2^2, \tag{68}$$

where

$$\vartheta(\tilde\eta) := \min_{S:|S|=\tau} \frac{\theta\lambda_S^2}{L_S}\frac{(1-\tilde\eta)^2}{\Lambda_S^2}. \tag{69}$$

Proof. The proof follows that of Lemma 9 in [15]. By Lipschitz continuity of the gradient of f:

$$\begin{aligned}
f(x_k + \alpha U_S t_k^S) &\leq f(x_k) + \alpha\langle\nabla_S f(x_k), t_k^S\rangle + \tfrac{L_S\alpha^2}{2}\|t_k^S\|_2^2 \\
&\overset{(64)}{\leq} f(x_k) - \alpha\|t_k^S\|_{H_k^S}^2 + \tfrac{L_S\alpha^2}{2\lambda_S}\|t_k^S\|_{H_k^S}^2.
\end{aligned}$$

The right hand side of the above inequality is minimized for $\alpha^* = \lambda_S/L_S$, which gives $f(x_k + \alpha^* U_S t_k^S) \leq f(x_k) - \frac{\lambda_S}{2L_S}\|t_k^S\|_{H_k^S}^2$. Observe that this step-size satisfies the backtracking line-search condition (22) because $f(x_k + \alpha^* U_S t_k^S) \leq f(x_k) - \frac{1}{2}\frac{\lambda_S}{L_S}\|t_k^S\|_{H_k^S}^2 < f(x_k) - \theta\frac{\lambda_S}{L_S}\|t_k^S\|_{H_k^S}^2$. 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 in Step 6 of FCD, satisfies (67). This, combined with Lemma 24, results in the following decrease in the objective function:

$$f(x_k) - f(x_k + \alpha U_S t_k^S) \geq \tfrac{\theta\lambda_S}{2L_S}\|t_k^S\|_{H_k^S}^2 \geq \tfrac{\theta\lambda_S^2}{2L_S}\|t_k^S\|_2^2 \geq \tfrac{\theta\lambda_S^2}{2L_S}\tfrac{(1-\eta_k^S)^2}{\Lambda_S^2}\|\nabla_S f(x_k)\|_2^2. \tag{70}$$

Using the definition (69) in (70) gives (68). □

Remark 26. *The quantity $\vartheta(\tilde{\eta})$ in* (69) *will appear in the complexity result for FCD in the smooth case. Notice that, while $\chi(\tilde{\eta})$ depends upon λ_S^3, $\vartheta(\tilde{\eta})$ only depends upon λ_S^2. Interestingly, the dependence of $\vartheta(\tilde{\eta})$ on λ_S is unaffected by an inexact search direction.*

	$\lambda_S = L_S$	$\lambda_S = \Lambda_S$
$\vartheta(\tilde{\eta})$	$\min_{S:\lvert S\rvert=\tau} \frac{\theta(1-\tilde{\eta})^2 L_S}{\Lambda_S^2}$	$\min_{S:\lvert S\rvert=\tau} \frac{\theta(1-\tilde{\eta})^2}{L_S}$

7.2 Iteration Complexity in the Convex Smooth Case

We are now ready to present iteration complexity results for FCD applied to smooth convex functions of the form (1) when $\Psi = 0$.

Theorem 27. *Let Assumptions 6, 9, 10, 11 and 23 hold. Choose an initial point $x_0 \in \mathbf{R}^N$, a target confidence $\rho \in (0,1)$, accuracy $0 < \epsilon < \max\{\mathcal{R}^2(x_0), f(x_0) - f^*\}$, fix $\tilde{\eta} \in [0,1)$. Let $\vartheta(\tilde{\eta})$ be as defined in* (69)*, with $\theta \in (0, 1/2)$ and let*

$$K \geq \frac{2N\mathcal{R}^2(x_0)}{\tau\vartheta(\eta)\epsilon}\left(1 + \ln\frac{1}{\rho}\right) + 2 - \frac{2N\mathcal{R}^2(x_0)}{\tau\vartheta(\eta)(f(x_0) - f^*)}. \tag{71}$$

If $\{x_k\}_{k\geq 0}$ are the random points generated by FCD (Algorithm 1) applied to problem (1) *with $\Psi = 0$, where $\eta_k^S \in [0,1)$ satisfies $\eta_k^S/(1-\eta_k^S)^2 \leq \eta < 1$ for all k and all $S \in 2^{[N]}$, then $\mathbf{P}(F(x_K) - F^* \leq \epsilon) \geq 1 - \rho$.*

Proof. By Lemma 25 $f(x_k) \leq f(x_0)$ for all k, and by convexity of f, we have $f(x_k) - f^* \leq \max_{x^* \in X^*}\langle \nabla f(x_k), x_k - x^*\rangle \leq \|\nabla f(x_k)\|_2 \mathcal{R}(x_0)$. Taking expectation of (70), and applying the previous, gives:

$$f(x_k) - \mathbf{E}[f(x_k + \alpha U_S t_k^S)|x_k] \geq \tfrac{\vartheta(\tilde{\eta})\tau}{2N}\|f(x_k)\|_2^2 \geq \tfrac{\vartheta(\tilde{\eta})\tau}{2N}\left(\tfrac{f(x_k) - f^*}{\mathcal{R}(x_0)}\right)^2.$$

Rearranging the above, and applying Theorem 14(i) with $c_1 = 2N\mathcal{R}^2(x_0)/\tau\vartheta(\tilde{\eta})$, gives the result. □

7.3 Iteration Complexity in the Strongly Convex Smooth Case

We now present iteration complexity results for FCD applied to smooth, strongly convex functions of the form (1), when $\Psi = 0$.

Theorem 28. *Let $\Psi = 0$, let $f = F$ be the strongly convex function defined in* (1) *with $\mu_f > 0$ and let Assumptions 6, 9, 10, 11 and 23 hold. Choose an initial point $x_0 \in \mathbf{R}^N$, a target confidence $\rho \in (0,1)$, a target accuracy $\epsilon > 0$, and fix $\tilde{\eta} \in [0,1)$. Let $\vartheta(\tilde{\eta})$ be as defined in* (69)*, with $\theta \in (0,1/2)$ and let*

$$K \geq \frac{N}{\tau\vartheta(\tilde{\eta})\mu_f}\ln\left(\frac{F(x_0) - F^*}{\epsilon\rho}\right). \tag{72}$$

If $\{x_k\}_{k\geq 0}$ are the random points generated by FCD (Algorithm 1) applied to problem (1) *with $\Psi = 0$, where $\eta_k^S \in [0, \tilde{\eta}]$, then $\mathbf{P}(F(x_K) - F^* \leq \epsilon) \geq 1 - \rho$.*

Proof. Notice that, by strong convexity, $f(x_k) - f^* \leq \frac{1}{2\mu_f}\|\nabla f(x_k)\|_2^2$, (see, e.g., [2, p.460]). Taking the expectation of (68), and using the previous, gives:

$$\mathbf{E}[f(x_k) - f(x_k + \alpha_S U_S t_k^S)|x_k] \geq \tfrac{\tau\vartheta(\tilde{\eta})}{2N}\|\nabla f(x_k)\|_2^2 \geq \tfrac{\tau\vartheta(\tilde{\eta})\mu_f}{N}(f(x_k) - f^*).$$

Rearranging, and applying Theorem 14(ii) with $c_2 = N/(\mu_f\tau\vartheta(\tilde{\eta})) > 1$ gives the result. □

8 Discussion/Comparison

In this section we discuss the complexity results derived in this paper, and compare them with the current state of the art results.

8.1 General Framework

The FCD framework has been designed to be extremely *flexible.* Moreover, this framework generalizes several existing algorithms.

To see this, suppose that FCD is set-up in the following way. Choose $|S| = \tau = 1$, $H_k^i = L_i$ (where L_i is the Lipschitz constant for the ith coordinate), suppose coordinate i is selected with uniform probability and suppose that (19) is solved exactly at every iteration. Then the update generated by FCD is given by

$$t_k^i = \arg\min_{t \in \mathbf{R}} \{\langle \nabla_i f(x), t\rangle + \tfrac{L_i}{2} t^2 + \Psi_i(x^i + t)\}.$$

However, this is equivalent to the update generated by UCDC (see Algorithm 1 and equation (17)) in [29]. So FCD generalizes UCDC, (and accepts a step-size $\alpha = 1$ for all iterations in this case, i.e., no line-search is needed by FCD.)

Now suppose that we assume that $\Psi = 0$, and that f is strongly convex. For simplicity, let us assume that f is quadratic. Moreover, suppose that FCD is set-up in the following way. Choose $|S| = \tau = 1$, $H_k^i = \nabla_{ii}^2 f$ (for a quadratic function the Hessian $\nabla^2 f(\cdot)$ is independent of x_k), suppose coordinate i is selected with uniform probability and suppose that (19) is solved exactly at every iteration. Then the update generated by FCD is $t_k^i = -(\nabla_i f(x_k))/(\nabla_{ii}^2 f)$. However, this is equivalent to the update generated by SDNA in [27] (Methods 1–3 in [27] are equivalent when $\tau = 1$). So FCD generalizes SDNA, (and also accepts a step-size $\alpha = 1$ for all iterations in this case, i.e., no line-search is needed by FCD.)

Clearly, a central advantage of FCD is it's flexibility. It recovers several existing state-of-the-art algorithms, depending on the particular choice of algorithm parameters. We stress that, for UCDC, one must always compute and use the Lipschitz constants, while for FCD, H_k^S can be any positive semi-definite matrix. Further, SDNA only applies to problems that are smooth and strongly convex, whereas FCD can be applied to general convex problems of the form (1).

8.2 Practicality vs Iteration Complexity

We remark that it is expected that the complexity results for FCD may be slightly worse than for other existing methods. There are several reasons for this:

1. FCD is a general framework that enables substantial user flexibility, as well as computational practicality. The user may choose the block size (i.e., the number of coordinates to be updated at each iteration) the matrix H_k^S. Therefore, the user has full flexibility regarding how *expensive* each iteration of FCD is. For example, choosing τ small, and H_k^S to be diagonal, will lead to cheap iterations. Choosing a larger block size and/or a full matrix H_k^S, will lead to iterations that are more expensive. However, it is anticipated that fewer iterations may be needed when H_k^S is a good approximation to a principal minor of the Hessian. Because of the user friendly, general and flexible framework, the complexity rates for FCD may be slightly worse than for methods that apply to specific problem instances, with specially tuned parameters.

2. Another difference between FCD, and other current methods, is that many methods do not incorporate any second order information, or they enforce that $H_k^S = L_S I$.The restrictions

made in other methods are usually imposed to ensure that their surrogate function/model, overapproximates $F(x_k + U_S t_k^S)$, so that minimizing their model will result in a reduction in the original function value. The assumption that FCD makes regarding H_k^S is much weaker (*any* symmetric positive definite H_k^S is allowed), which means that FCD requires a line search, to guarantee a reduction in the objective function value at every iteration, and ultimately, convergence. We prefer an algorithm with fewer assumptions, so that it is *widely applicable* and *practical*, but we pay the price of a (potentially) slightly weaker complexity result.

3. FCD is one of the (very) few coordinate descent methods that allow inexact updates. It is expected that inexact updates will lead to cheaper iterations (it is intuitive that solving a subproblem inexactly, will usually require less computational effort than solving it exactly), although this may result in slightly more iterations compared with an 'exact' method. We stress that, despite the potential increase in the number of iterations, it is still expected that inexact updates will result in a lower computational run time overall.

We stress that, despite the slightly weaker iteration complexity results for FCD, the algorithm works extremely well in practice, and often outperforms algorithms with 'better' complexity rates, (see the numerical experiments in Section 9). Moreover, the user has some control over the complexity results for FCD through their choice of parameters. For example, λ_S and Λ_S (Assumption 9) are user defined parameters, which appear in the complexity rates for FCD (see (45) and (69)), so the complexity rate for FCD can be larger or smaller, depending on how the user chooses these parameters.

8.3 Comparison of complexity results

We will now compare the complexity results obtained in this work for FCD, with those of UCDC in [29] and PCDM [30].

8.3.1 Comparison of complexity results with UCDC [29]

UCDC is a *serial* method that allows *exact updates only*, so in this section we assume that $|S| = \tau = 1$, and that FCD uses only exact updates ($\tilde{\eta} \equiv 0$). This means that we have L_i for $i = 1, \ldots, N$, and H_k^i is a scalar, for all k. The complexity results in [29] use 'scaled' quantities that depend on the vector of Lipschitz constants ($L = (L_1, \ldots, L_N)$) while the quantities in this work depend on the 'unscaled' 2-norm (i.e., $e = (1, \ldots, 1)$).[6] Moreover, in [29], the authors suppose that strong convexity can come from f or Ψ, or both. Here, to allow easy comparison, we suppose that $\mu_\Psi = 0$, which means that, in this section, if strong convexity is assumed, then $\mu_f = \mu_F$. To this end, define the following constants, which depend a 'scaling' vector $w \in \mathbf{R}^N_{++}$:

$$c_1(w) = \tfrac{2N}{\epsilon} \max\{\mathcal{R}_w^2(x_0), F(x_0) - F^*\}, \text{ and } c_2(w) = \tfrac{2N\mathcal{R}_w^2(x_0)}{\epsilon}. \tag{73}$$

Table 2 presents the complexity results obtained for RCDC [29], and the complexity results obtained in this work for FCD. (We omit the 'log' terms, which are the same for both methods.) The following notation is used in the table. Constants $\xi_0 = F(x_0) - F^*$, $\mu_\phi(w)$ denotes the strong convexity parameter of function ϕ with respect to a 'w-weighted' norm, $\mu = (\mu_f + \mu_\Psi)/(1 + \mu_\Psi)$ and $\mathcal{R}_w(x_0)$ is defined in (12).

[6]Using the notation of [29], we have $n = N$. Further, although UCDC allows arbitrary probabilities in the smooth case, we restrict our attention to uniform probabilities only, so N appears in the C-S and SC-S results in Table 2.

F	RCDC [29]	FCD [this paper]	Theorem
C-N	$c_1(L)$	$c_1(e)/\chi(0)$	19(i)
C-N	$c_2(L)$	$c_2(e)/\chi(0)$	19(ii)
SC-N	$N/\mu_f(L)$	$N/\chi(0)\delta$	21
C-S	$c_2(L)$	$c_2(e)/\vartheta(0)$	27
SC-S	$N/\mu_f(L)$	$N/\vartheta(0)\mu_f$	28

Table 2: Comparison of the iteration complexity results for coordinate descent methods using an inexact update and using an exact update (C=Convex, SC=Strongly Convex, N=Nonsmooth, S = Smooth).

Case C-N(i). In this case, the difference between the complexity rates for UCDC and FCD appears in the constants $c_1(L)$ and $c_1(e)/\chi(0)$. Clearly, if $\mathcal{R}_L^2(x_0) \leq \xi_0^F$ then $c_1(L) = c_1(e)$, so that the the complexity rate of FCD is simply $1/\chi(0)$ times worse than UCDC. (Recall Table 1 for a comparison of $\chi(\cdot)$ values.) On the other hand, if $\mathcal{R}_L^2(x_0) > \xi_0^F$, then it is difficult to compare the complexity rates directly. Notice that the dependence of FCD on the Lipschitz constants is explicit (the constants L_i appear in $\chi(0)$), whereas the dependence of UCDC on the Lipschitz constants is somehow 'hidden' in the weighted term $\mathcal{R}_L^2(x_0)$. However, consider the case when $L_i = L_j$ for all i, j. Then $\|y-x\|_L^2 = \sum_{i=1}^N L_i\|y_i - x_i\|_2^2 = L_i\|y-x\|_2^2$, so that $\mathcal{R}_L^2(x_0) = L_i\mathcal{R}^2(x_0)$ in this case. Then we simply compare L_i vs $1/\chi(0)$. Now suppose we choose $H_k^i = L_i$ for all k, so that $\chi(0) = (1-\theta)/2$. If $\theta \approx 0$, then FCD has a better complexity rate than UCDC whenever $L_i > 2$. Of course, in other circumstances, UCDC may have a better complexity rate than FCD — this is simply one particular example.

Case C-N(ii). Here we compare $c_2(L)$ and $c_2(\mathbf{1})$. Both rates contain the term $2N/\epsilon$, so it remains to compare $\mathcal{R}_L^2(x_0)$ and $\mathcal{R}^2(x_0)/\chi(0)$. Thus, see the discussion for C-N(i).

Case SC-N. Substituting $\mu_f = \mu_F$ into (57), shows that $\delta = \mu_f/\Lambda_{\max}$, so in this case we compare $1/\mu_f(L)$ with $\Lambda_{\max}/\chi(0)\mu_f$. Moreover, the Lipschitz constants appear explicitly in $\chi(0)$ in the analysis in this work, while they are again 'hidden' in the term $\mu_f(L)$ in [29]. Again, this makes it difficult to compare the rates directly. However, consider the following example, where $\mu_f = \Lambda_{\max}$. Then $\lambda_S = L_S$ (because $\lambda_S \leq \Lambda_{\max} = \mu_f \leq L_S$), so $\Lambda_{\max}/\chi(0)\mu_f = 2/(1-\theta)$. Then, if $\theta \approx 0$, FCD has a better complexity rate than UCDC, whenever $\mu_f(L) \leq 1/2$. (Note that it always holds that $\mu_f(L) \leq 1$ [29, (12)], and this does not necessarily imply that $(\mu_f(e) \equiv)\mu_f \leq 1/2$).

Case C-S. Similar to Case C-N(ii), we compare $\mathcal{R}_L^2(x_0)$ and $\mathcal{R}^2(x_0)/\vartheta(0)$. Suppose we have the same set up as Case C-N(i), so that we compare L_i with $1/\vartheta(0)$. Choosing $\lambda_S = \Lambda_S$ and $\theta \approx 1/2$ shows that $1/\vartheta(0) \approx 2L_i$, so that the complexity rates for FCD is approximately twice that for UCDC.

Case SC-S. Here we compare the quantities $1/\mu_f(L)$ and $1/\mu_f\vartheta(0)$. All that can be said for UCDC is that $1/\mu_f(L) \geq 1$ because $\mu_f(L) \leq 1$. Sometimes it may be the case that $1/\mu_f(L) \approx 1$, whereas in other cases, we may have $1/\mu_f(L) \gg 1$. Note that, for FCD, if we choose $\lambda_S = \Lambda_S$, then $1/\mu_f\vartheta(0) = L_S/\theta\mu_f$. For a well conditioned problem, (i.e., $\mu_f \approx L_S$), choosing $\theta \approx 1$ gives $1/\mu_f\vartheta(0) \approx 1$. On the other hand, if $L_S \gg \theta\mu_f$, then $1/\mu_f\vartheta(0) \gg 1$. So, while the rates for FDC and UCDC cannot be compared directly, the complexity rates are similar, when taking into account the problem conditioning.

8.3.2 Comparison of complexity results with PCDM [30]

Now we compare FCD with PCDM [30]. PCDM is a parallel coordinate descent method, that uses *exact updates*, so in this case we suppose that $|S| = \tau \geq 1$, and that FCD also uses exact updates. Furthermore, we will suppose that the coordinates are sampled according to Definition 5 (which is called a τ-nice sampling in [30]). Define $\beta = 1 + \frac{(\omega-1)(\tau-1)}{\max\{1,N-1\}}$, where ω is a measure of the 'separability' of the objective function. Then, the iteration complexity for PCDM is presented in Table 3.

	C-N	SC-N
PCDM [30]	$\frac{2N}{\tau\epsilon}\max\{\beta\mathcal{R}_L^2(x_0), F(x_0)-F^*\}$	$\frac{\beta N}{\tau\mu_f(L)}$
FCD	$\frac{2N}{\tau\epsilon\chi(0)}\max\{\mathcal{R}^2(x_0), F(x_0)-F^*\}$	$\frac{N}{\tau\delta\chi(0)}$

Table 3: Complexity results for PCDM [30].

C-N. Note that $\beta \geq 1$, so if $\beta\mathcal{R}_L^2(x_0) \leq F(x_0) - F^*$, then the complexity rate for FCD is $1/\chi(0)$ times worse than for PCDM. On the other hand, suppose that $\mathcal{R}_L^2(x_0) > F(x_0) - F^*$. Then we must compare the quantities $\beta\mathcal{R}_L^2(x_0)$ and $\mathcal{R}^2(x_0)/\chi(0)$. Similar arguments to the Case C-N(i) in Section 8.3.1 can be made here. However, we note that β depends upon τ, N and ω. While τ and N can be controlled (they are parameters of the algorithm), ω depends upon the problem data, and could be very large, implying large β.

SC-N. In this case we compare the quantities $\beta/\mu_f(L)$ and $1/\delta\chi(0)$. The case for FCD and $1/\delta\chi(0)$ is made in Section 8.3.1, Case SC-N. Now consider PCDM. For a well conditioned problem, and separable problem, we may have $\mu_f(L) \approx 1 \approx \omega$, so that $\beta/\mu_f(L) \approx 1$. However, for a poorly conditioned problem, ω may be very large, and $\mu_f(L)$ may be very small, so that $\beta/\mu_f(L) \gg 1$. Again, while the rates for FDC and PCDM cannot be compared directly, the complexity rates are similar, when taking into account the problem conditioning.

9 Numerical Experiments

In this section we examine the performance of two versions of FCD and two versions of a Uniform Coordinate Descent method (UCDC) [29] on two common optimization problems. The first problem is an ℓ_1-regularized least squares problem of the form (1) with

$$f(x) = \tfrac{1}{2}\|Ax - b\|^2 \quad \text{and} \quad \Psi(x) = c\|x\|_1, \tag{74}$$

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) with

$$f(x) = \sum_{j=1}^{m} \log(1 + e^{-b_j x^T a_j}) \quad \text{and} \quad \Psi(x) = c\|x\|_1, \tag{75}$$

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

For (74) a synthetic sparse large scale experiment is performed and for (75) we compare the methods on two real world large scale problems from machine learning. Notice that for both (74) and (75), $\Psi(x) = c\|x\|_1$, which is fully separable into coordinates. This means that, for

FCD, 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).

9.1 Implementations of FCD and UCDC

In this section we discuss some details of the implementations of methods FCD and UCDC.

9.1.1 FCD

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

We implement two versions of FCD, which we denote by FCD v.1 and FCD v.2. The two versions only differ in how matrix H_k^S is chosen. In particular, for FCD v.1 we set $H_k^S := \mathrm{diag}(\nabla_S^2 f(x_k))$ for all i and k. In this case subproblem (19) is separable and it has a closed form solution

$$t_k^S = \mathcal{S}(x_k^S - (H_k^S)^{-1}\nabla_S f(x_k^S), c\,\mathrm{diag}((H_k^S)^{-1})),$$

where

$$\mathcal{S}(u,v) = \mathrm{sign}(u)\max(|u| - v, 0) \tag{76}$$

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 (21).

For FCD v.2, we set

$$H_k^S := \nabla_S^2 f(x_k) + \rho I_{N_i}, \text{ for all } i \text{ and } k, \tag{77}$$

where $\rho > 0$ guarantees that H_k^S is positive definite for all i, k. Hence, the subproblem (19) is well defined. The larger ρ is the smaller the condition number of matrix H_k becomes, hence, the faster subproblem (19) 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^S we solve subproblems (19) 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 [3] to result in a robust and efficient deterministic version of FCD, i.e. $\tau = N$ (one block of size N). Note that we never explicitly form matrix H_k^S, we only perform matrix-vector products with it in a matrix-free manner.

9.1.2 UCDC

We also implement two versions of a uniform coordinate descent method as it is described in Algorithm 2 in [29]. 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 (74), the block Lipschitz constants correspond to the maximum eigenvalue of $A_i^T A_i$, where $A_i := AU_S$.) For this reason, we do not compute the actual block Lipschitz constants, rather, we use an overapproximation.

To this end, let $L_j > 0 \, \forall j = 1, 2, \dots, N$ denote the coordinate Lipschitz constants of function f. Then the direction t^S at every iteration is obtained by solving exactly subproblem (19) with

$$H_k^S := \Big(\sum_{j \in i} L_j\Big) I_\tau, \tag{78}$$

using operator (76). Notice that for problem (74), (78) is equivalent to $H_k^S = \text{trace}(A_i^T A_i) I_\tau$, 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 [29] is a special case of FCD where the subproblem (19) is solved exactly and line search is unnecessary. This is because, by setting H_k^S as in (78), then subproblem (19) is an over estimator of function F along block coordinate direction t^S (for details we refer the reader to [29]).

9.2 Termination Criteria and Parameter Tuning

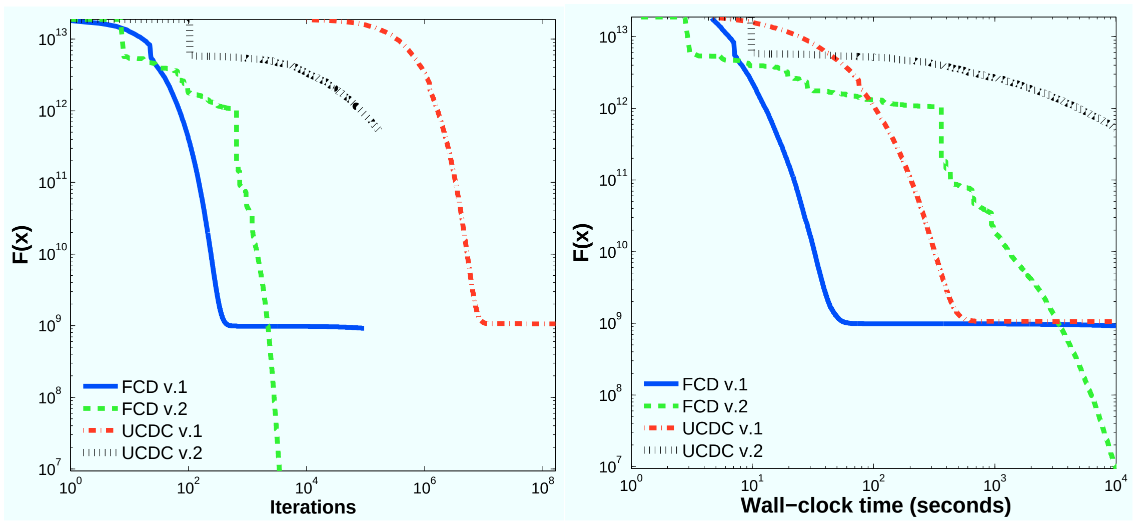
The only termination criteria that FCD and UCDC should have are maximum number of iterations or maximum running time. This is because using subgradients as a measure of distance from optimality or any other operation of similar cost are considered as expensive tasks for large scale problems. In our experiments FCD and UCDC are terminated when their running time exceeds the maximum allowed running time. Furthermore, for FCD we set parameter η_k^S in (21) equal to 0.9 $\forall i, k$ and $\rho = 10^{-6}$ in (77). 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.

9.3 ℓ_1-Regularized Least Squares

In this subsection we present the performance of FCD and UCDC on the ℓ_1-regularized least squares problem (74). For this problem the data A and b were synthetically constructed using a generator proposed in [24, 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 $L_j := \|A_j\|_2^2 \quad j = 1, 2, \dots, N$, and for FCD v.1, FCD v.2 and UCDC v.2, we set $\tau = \lceil 0.01N \rceil$.

The result of this experiment is shown in Figure 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 1a that block methods FCD v.1, FCD 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 computational complexity of the former methods compared to the latter. FCD 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 1b that for purely practical purposes it might



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

Figure 1: Performance of all four methods FCD 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.

be better to have a combination of methods FCD 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 FCD 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.*

9.4 ℓ_1-Regularized Logistic Regression

In this section we present the performance of FCD and UCDC on the ℓ_1-regularized logistic regression problems (75).

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 [42] for further details.

We present the performance of the methods on two sparse large scale data sets. Problem details are given in Table 4, where $A \in \mathbb{R}^{m\times N}$ is a matrix whose rows are training samples.

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

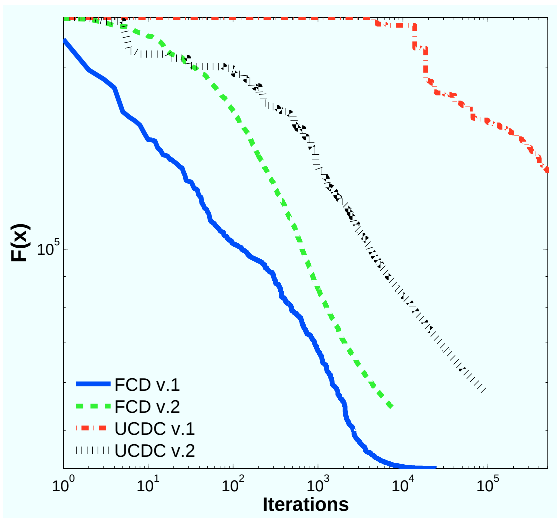
The data sets can be downloaded from the collection of LSVM problems in `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`. For both experiments we set $c = 10$, which resulted in more than 99% classification accuracy of the used data sets.

By [29, Table 10], the coordinate Lipschitz constants for UCDC are

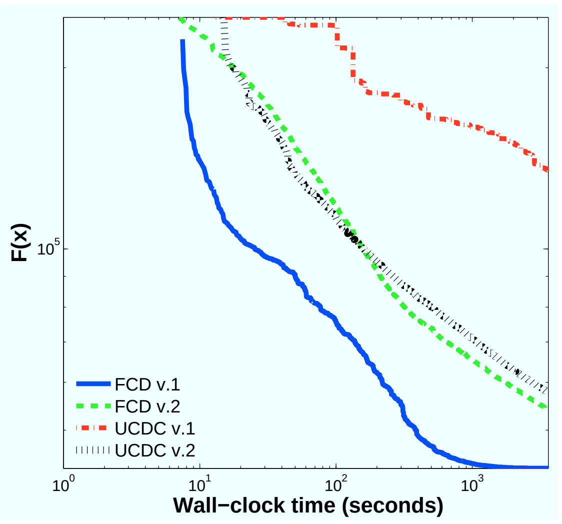
$$L_j := \frac{1}{4}\sum_{q=1}^{m}(A_{qj}y_q)^2 \quad \forall j = 1, 2, \dots, N,$$

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

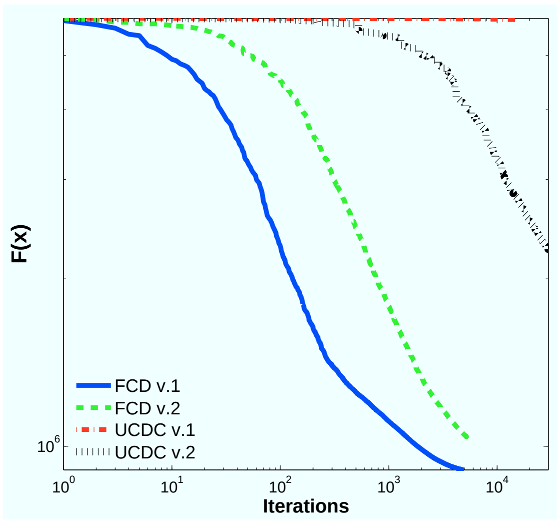
The result of this experiment is shown in Figure 2. In this experiment all methods were



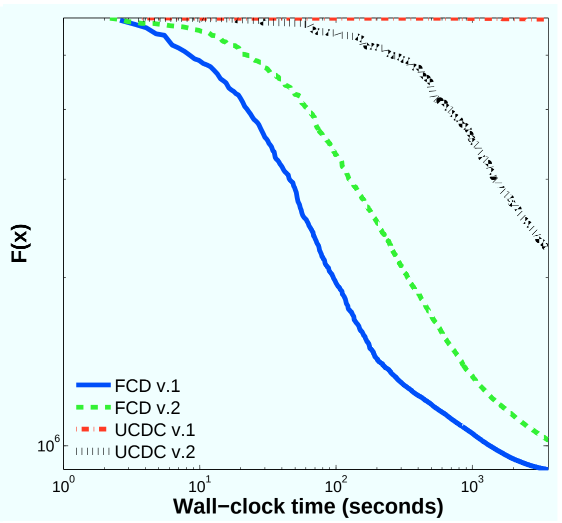
(a) webspam, $F(x)$ against iterations



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



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



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

Figure 2: Performance of all four methods FCD 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.*

terminated after one hour of running time. Notice that FCD versions were more efficient than both UCDC versions, with FCD v.1 being the fastest among all. An interesting observation in Figures 2a and 2c is that FCD versions had similar per iteration computational complexity since they performed similar number of iterations within the maximum allowed running time. However, for FCD 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 FCD v.2. Finally, in this experiment we observed that both FCD versions accepted unit step sizes for a major part of the process.

10 Conclusion

In this work we have presented a flexible randomized block coordinate descent method (FCD) that can be applied to convex composite functions of the form (1).

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 FCD, we present synthetic and real world large scale examples where the number of iterations substantially decreases, as well as the overall time.

We have also presented high probability iteration complexity guarantees to show that FCD converges in expectation.

11 Future directions

- Optimal probabilities: much work showing you can improve the complexity results using optimized (optimal) probabilities. Improve complexity from squared dependence on L_S to linear dependence.
- Parallelization: take advantage of modern computer architecture: alter/eliminate the line search

References

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

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press New York, NY, USA, 2004.

[3] 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].

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

[5] 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.

[6] A. Cassioli, D. Di Lorenzo, and M. Sciandrone. On the convergence of inexact block coordinate descent methods for constrained optimization. *European Journal of Operational Research*, 231(2):274 – 281, 2013.

[7] A. Daneshmand, F. Facchinei, V. Kungurtsev, and G. Scutari. Hybrid random/deterministic parallel algorithms for nonconvex big data optimization. Technical report, State University of New York at Buffalo; University of Rome La Sapienza; Czech Technical University, September 2014. arXiv:1407.4504v2 [cs.DC].

[8] O. Devolder, F. Glineur, and Yu. Nesterov. Intermediate gradient methods for smooth convex problems with inexact oracle. Technical Report 2013017, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2013.

[9] O. Devolder, F. Glineur, and Yu. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1-2):37–75, 2014.

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

[11] F. Facchinei, S. Sagratella, and G. Scutari. Flexible parallel algorithms for big data optimization. *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 7208–7212, May 2014.

[12] F. Facchinei, G. Scutari, and S. Sagratella. Parallel selective algorithms for nonconvex big data optimization. *IEEE Transactions on Signal Processing*, 63(7):1874–1889, April 2015.

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

[14] K. Fountoulakis and J. Gondzio. Performance of first- and second-order methods for big data optimization. Technical Report ERGO-15-005, University of Edinburgh, March 2015. arXiv:1503.03520v2 [math.OC].

[15] K. Fountoulakis and J. Gondzio. A second-order method for strongly convex ℓ_1-regularization problems. *Mathematical Programming*, March 2015. Accepted.

[16] 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].

[17] D. E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms.* Addison-Wesley, 1997. 3rd Edition.

[18] 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.

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

[20] Z. Lu and L. Xiao. On the complexity analysis of randomized block-coordinate descent methods. *Mathematical Programming*, August 2014. Accepted.

[21] 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.

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

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

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

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

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

[27] Z. Qu, P. Richtárik, M. Takáč, and O. Fercoq. SDNA: stochastic dual newton ascent for empirical risk minimization. Technical report, University of Edinburgh, February 2015.

[28] 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].

[29] P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1–2):1–38, 2014.

[30] P. Richtárik and M. Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, pages 1–52, April 2015.

[31] 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].

[32] 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].

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

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

[35] 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].

[36] R. Tappenden, M. Takáč, and P. Richtárik. On the complexity of parallel coordinate descent. Technical report, University of Edinburgh, March 2015. arXiv:1503.03033v1 [math.OC].

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

[38] P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Math. Program., Ser. B*, 117:387–423, 2009.

[39] Paul Tseng and Sangwoon Yun. Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. *Journal of Optimization Theory and Applications*, 140(3):513–535, 2009.

[40] Paul Tseng and Sangwoon Yun. A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training. *Computational Optimization and Applications*, 47(2):179–206, 2010.

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

[42] 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.