Math. Program., Ser. B (2015) 151:3–34
DOI 10.1007/s10107-015-0892-3

FULL LENGTH PAPER

Coordinate descent algorithms

Stephen J. Wright[1]

Received: 30 November 2014 / Accepted: 17 February 2015 / Published online: 25 March 2015

Abstract Coordinate descent algorithms solve optimization problems by successively performing approximate minimization along coordinate directions or coordinate hyperplanes. They have been used in applications for many years, and their popularity continues to grow because of their usefulness in data analysis, machine learning, and other areas of current interest. This paper describes the fundamentals of the coordinate descent approach, together with variants and extensions and their convergence properties, mostly with reference to convex objectives. We pay particular attention to a certain problem structure that arises frequently in machine learning applications, showing that efficient implementations of accelerated coordinate descent algorithms are possible for problems of this type. We also present some parallel variants and discuss their convergence properties under several models of parallel execution.

Keywords Coordinate descent · Randomized algorithms · Parallel numerical computing

Mathematics Subject Classification 49M20 · 90C25

1 Introduction

Coordinate descent (CD) algorithms for optimization have a history that dates to the foundation of the discipline. They are iterative methods in which each iterate is

The author was supported by NSF Awards DMS-1216318 and IIS-1447449, ONR Award N00014-13-1-0129, AFOSR Award FA9550-13-1-0138, and Subcontract 3F-30222 from Argonne National Laboratory.

✉ Stephen J. Wright
swright@cs.wisc.edu

[1] Department of Computer Sciences, University of Wisconsin-Madison, 1210 W. Dayton St., Madison, WI 53706-1685, USA

obtained by fixing most components of the variable vector x at their values from the current iteration, and approximately minimizing the objective with respect to the remaining components. Each such subproblem is a lower-dimensional (even scalar) minimization problem, and thus can typically be solved more easily than the full problem.

CD methods are the archetype of an almost universal approach to algorithmic optimization: solving an optimization problem by solving a sequence of simpler optimization problems. The obviousness of the CD approach and its acceptable performance in many situations probably account for its long-standing appeal among practitioners. Paradoxically, the apparent lack of sophistication may also account for its unpopularity as a subject for investigation by optimization researchers, who have usually been quick to suggest alternative approaches in any given situation. There are some very notable exceptions. The 1970 text of Ortega and Rheinboldt [39, Section 14.6] included a comprehensive discussion of "univariate relaxation," and such optimization specialists as Luo and Tseng [30,31], Tseng [54], and Bertsekas and Tsitsiklis [5] made important contributions to understanding the convergence properties of these methods in the 1980s and 1990s.

The situation has changed in recent years. Various applications (including several in computational statistics and machine learning) have yielded problems for which CD approaches are competitive in performance with more reputable alternatives. The properties of these problems (for example, the low cost of calculating one component of the gradient, and the need for solutions of only modest accuracy) lend themselves well to efficient implementations of CD, and CD methods can be adapted well to handle such special features of these applications as nonsmooth regularization terms and a small number of equality constraints. At the same time, there have been improvements in the algorithms themselves and in our understanding of them. Besides their extension to handle the features just mentioned, new variants that make use of randomization and acceleration have been introduced. Parallel implementations that lend themselves well to modern computer architectures have been implemented and analyzed. Perhaps most surprisingly, these developments are relevant even to the most fundamental problem in numerical computation: solving the linear equations $Aw = b$.

In the remainder of this section, we state the problem types for which CD methods have been developed, and sketch the most fundamental versions of CD. Section 2 surveys applications both historical and modern. Section 3 sketches the types of algorithms that have been implemented and analyzed, and presents several representative convergence results. Section 4 focuses on parallel CD methods, describing the behavior of these methods under synchronous and asynchronous models of computation.

Our approach throughout is to describe the CD methods in their simplest forms, to illustrate the fundamentals of the applications, implementations, and analysis. We focus almost exclusively on methods that adjust just one coordinate on each iteration. Most applications use *block* CD methods, which adjust groups of blocks of indices at each iteration, thus searching along a coordinate hyperplane rather than a single coordinate direction. Most derivation and analysis of single-CD methods can be extended without great difficulty to the block-CD setting; the concepts do not change fundamentally.

We mention too that much effort has been devoted to developing more general forms of CD algorithms and analysis, involving weighted norms and other features,

that allow more flexible implementation and allow the proof of stronger and more general (though usually not qualitatively different) convergence results.

1.1 Formulations

The problem considered in most of this paper is the following unconstrained minimization problem:

$$\min_x \ f(x), \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is continuous. Different variants of CD make further assumptions about f. Sometimes it is assumed to be smooth and convex, sometimes smooth and possibly nonconvex, and sometimes smooth but with a restricted domain. (We will make such assumptions clear in each discussion of algorithmic variants and convergence results.)

Motivated by recent popular applications, it is common to consider the following structured formulation:

$$\min_x \ h(x) := f(x) + \lambda \Omega(x), \tag{2}$$

where f is smooth, Ω is a regularization function that may be nonsmooth and extended-valued, and $\lambda > 0$ is a regularization parameter. Ω is often convex and usually assumed to be separable or block-separable. When separable, Ω has the form

$$\Omega(x) = \sum_{i=1}^{n} \Omega_i(x_i). \tag{3}$$

where $\Omega_i : \mathbb{R} \to \mathbb{R}$ for all i. The best known examples of separability are the ℓ_1-norm (in which $\Omega(x) = \|x\|_1$ and hence $\Omega_i(x_i) = |x_i|$) and box constraints (in which $\Omega_i(x_i) = I_{[l_i,u_i]}(x_i)$ is the indicator function for the interval $[l_i, u_i]$). Block separability means that the $n \times n$ identity matrix can be partitioned into column submatrices $U_i, i = 1, 2, \ldots, N$ such that

$$\Omega(x) = \sum_{i=1}^{N} \Omega_i \left(U_i^T x \right). \tag{4}$$

Block-separable examples include group-sparse regularizers in which $\Omega_i(z_i) := \|z_i\|_2$. Formulations of the type (2), with separable or block-separable regularizers, arise in such applications as compressed sensing, statistical variable selection, and model selection.

The class of problems known as empirical risk minimization (ERM) gives rise to a formulation that is particularly amenable to CD; see [51]. These problems have the form

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \phi_i \left(c_i^T w \right) + \lambda g(w), \tag{5}$$

for vectors $c_i \in \mathbb{R}^d$, $i = 1, 2, \ldots, n$ and convex functions ϕ_i, $i = 1, 2, \ldots, n$ and g. We can express linear least-squares, logistic regression, support vector machines, and other problems in this framework. Recalling the following definition of the conjugate t^* of a convex function t:

$$t^*(y) = \sup_z \left(z^T y - t(z) \right), \tag{6}$$

we can write the Fenchel dual [48, Section 31] of (5) as follows:

$$\min_{x \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^n \phi_i^*(-x_i) + \lambda g^* \left(\frac{1}{\lambda n} C x \right), \tag{7}$$

where C is the $d \times n$ matrix whose columns are $c_i, i = 1, 2, \ldots, n$. The dual formulation (7) is has special appeal as a target for CD, because of separability of the summation term.

One interesting case is the system of linear equations

$$Aw = b, \quad \text{where } A \in \mathbb{R}^{m \times n}, \tag{8}$$

which we assume to be a feasible system. The least-norm solution is found by solving

$$\min_{w \in \mathbb{R}^n} \frac{1}{2} \|w\|_2^2 \ \text{ subject to } Aw = b, \tag{9}$$

whose Lagrangian dual is

$$\min_{x \in \mathbb{R}^m} \ f(x) := \frac{1}{2} \|A^T x\|_2^2 - b^T x. \tag{10}$$

(We recover the primal solution from (10) by setting $w = A^T x$.) We can see that (10) is a special case of the Fenchel dual (7) obtained from (5) if we set

$$C \leftarrow A^T, \quad g(w) = \frac{1}{2} \|w\|_2^2, \quad \phi_i(t_i) = I_{\{b_i\}}(t_i), \quad \lambda = 1/n,$$

where $I_{\{b_i\}}$ denotes the indicator function for b_i, which is zero at b_i and infinite elsewhere. (Its conjugate is $I^*_{\{b_i\}}(s_i) = b_i s_i$.) The primal problem (9) can be restated correspondingly as

$$\min_{w \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^m I_{\{b_i\}}(A_i w) + \frac{1}{n} \|w\|_2^2,$$

where A_i denotes the ith row of the matrix A in (8), which has the form (5).

1.2 Outline of coordinate descent algorithms

The basic CD framework for continuously differentiable minimization is shown in Algorithm 1. Each step consists of evaluation of a single component i_k of the gradient ∇f at the current point, followed by adjustment of the i_k component of x, in the opposite direction to this gradient component. (Here and throughout, we use $[\nabla f(x)]_i$ to denote the ith component of the gradient $\nabla f(x)$.) There is much scope for variation within this framework. The components can be selected in a cyclic fashion, in which $i_0 = 1$ and

$$i_{k+1} = [i_k \bmod n] + 1, \quad k = 0, 1, 2, \ldots. \tag{11}$$

They can be required to satisfy an "essentially cyclic" condition, in which for some $T \geq n$, each component is modified at least once in every stretch of T iterations, that is,

$$\cup_{j=0}^{T} \{i_{k-j}\} = \{1, 2, \ldots, n\}, \quad \text{for all } k \geq T. \tag{12}$$

Alternatively, they can be selected randomly at each iteration (though not necessarily with equal probability). Turning to steplength α_k: we may perform exact minimization along the i_k component, or choose a value of α_k that satisfies traditional line-search conditions (such as sufficient decrease), or make a predefined "short-step" choice of α_k based on prior knowledge of the properties of f.

Algorithm 1 Coordinate Descent for (1)

Set $k \leftarrow 0$ and choose $x^0 \in \mathbb{R}^n$;
repeat
  Choose index $i_k \in \{1, 2, \ldots, n\}$;
  $x^{k+1} \leftarrow x^k - \alpha_k [\nabla f(x^k)]_{i_k} e_{i_k}$ for some $\alpha_k > 0$;
  $k \leftarrow k + 1$;
until termination test satisfied;

The CD framework for the separable regularized problem (2), (3) is shown in Algorithm 2. At iteration k, a scalar subproblem is formed by making a linear approximation to f along the i_k coordinate direction at the current iterate x^k, adding a quadratic damping term weighted by $1/\alpha_k$ (where α_k plays the role of a steplength), and treating the relevant regularization term Ω_i explicitly. Note that when the regularizer Ω_i is not present, the step is identical to the one taken in Algorithm 1. For some interesting choices of Ω_i (for example $\Omega_i(\cdot) = |\cdot|$), it is possible to write down a closed-form solution of the subproblem; no explicit search is needed. The operation of solving such subproblems is often referred to as a "shrink operation," which we denote by S_β and define as follows:

$$S_\beta(\tau) := \min_{\chi} \frac{1}{2\beta} \|\chi - \tau\|_2^2 + \Omega_i(\chi). \tag{13}$$

By stating the subproblem in Algorithm 2 equivalently as

$$\min_{\chi} \frac{1}{2\lambda\alpha_k} \left\| \chi - (x_{i_k}^k - \alpha_k [\nabla f(x^k)]_{i_k}) \right\|^2 + \Omega_i(\chi),$$

we can express the CD update as $z_{i_k}^k \leftarrow S_{\lambda\alpha_k}(x_{i_k}^k - \alpha_k [\nabla f(x^k)]_{i_k})$.

Algorithm 2 Coordinate Descent for (2),(3)

Set $k \leftarrow 0$ and choose $x^0 \in \mathbb{R}^n$;
repeat
  Choose index $i_k \in \{1, 2, \ldots, n\}$;
  $z_{i_k}^k \leftarrow \arg\min_\chi \; (\chi - x_{i_k}^k)^T [\nabla f(x^k)]_{i_k} + \frac{1}{2\alpha_k} \|\chi - x_{i_k}^k\|_2^2 + \lambda \Omega_i(\chi)$ for some $\alpha_k > 0$;
  $x^{k+1} \leftarrow x^k + (z_{i_k}^k - x_{i_k}^k) e_{i_k}$;
  $k \leftarrow k + 1$;
until termination test satisfied;

Algorithms 1 and 2 can be extended to block-CD algorithms in a straightforward way, by updating a block of coordinates (denoted by the column submatrix U_{i_k} of the identity matrix) rather than a single coordinate. In Algorithm 2, it is assumed that the choice of block is consistent with the block-separable structure of the regularization function Ω, that is, U_{i_k} is a concatenation of several of the submatrices U_i in (4).

1.3 Application to linear equations

For the formulation (10) that arises from the linear system $Aw = b$, let us assume that the rows of A are normalized, that is,

$$\|A_i\|_2 = 1 \quad \text{for } i = 1, 2, \ldots, m. \tag{14}$$

Applying Algorithm 1 to (10) with $\alpha_k \equiv 1$, each step has the form

$$x^{k+1} \leftarrow x^k - \left(A_{i_k} A^T x^k - b_{i_k} \right) e_{i_k}. \tag{15}$$

If we maintain and update the estimate w^k of the solution to the primal problem (9) after each update of x^k, according to $w^k = A^T x^k$, we obtain

$$w^{k+1} \leftarrow w^k - \left(A_{i_k} A^T x^k - b_{i_k} \right) A_{i_k}^T = w^k - \left(A_{i_k} w^k - b_{i_k} \right) A_{i_k}^T, \tag{16}$$

which is the update formula for the Kaczmarz algorithm [22]. Following this update, we have using (14) that

$$A_{i_k} w^{k+1} = A_{i_k} w^k - \left(A_{i_k} w^k - b_{i_k} \right) = b_{i_k},$$

so that the i_k equation in the system $Aw = b$ is now satisfied. This method if sometimes known as the "method of successive projections" because it projects onto the feasible hyperplane for a single constraint at every iteration.

1.4 Relationship to other methods

Stochastic gradient (SG) methods, also undergoing a revival of interest because of their usefulness in data analysis and machine learning applications, minimize a smooth function f by taking a (negative) step along an estimate g^k of the gradient $\nabla f(x^k)$ at iteration k. It is often assumed that g^k is an unbiased estimate of $\nabla f(x^k)$, that is, $\nabla f(x^k) = E(g^k)$, where the expectation is taken over whatever random variables were used in obtaining g^k from the current iterate x^k. Randomized CD algorithms can be viewed as a special case of SG methods, in which $g^k = n[\nabla f(x^k)]_{i_k} e_{i_k}$, where i_k is chosen uniformly at random from $\{1, 2, \ldots, n\}$. Here, i_k is the random variable, and we have

$$E(g^k) = \frac{1}{n} \sum_{i=1}^{n} n \left[\nabla f(x^k)\right]_i e_i = \nabla f(x^k),$$

certifying unbiasedness. However, CD algorithms have the advantage over general SG methods that descent in f can be guaranteed at every iteration. Moreover, the variance of the gradient estimate g^k shrinks to zero as the iterates converge to a solution x^*, since every component of $\nabla f(x^*)$ is zero. By contrast, in general SG methods, the gradient estimates g^k may be nonzero even when x^k is a solution.

The relationship between CD and SG methods can also be discerned from the Fenchel dual pair (5) and (7). SG methods are quite popular for solving formulation (5), where the estimate g^k is obtained by taking a single term i_k from the summation and using $\nabla \phi_{i_k}(c_{i_k}^T w) c_{i_k}$ as the estimate of the gradient of the *full* summation. This approach corresponds to applying CD to the dual (7), where the component i_k of x is selected for updating at iteration k. This relationship is typified by the Kaczmarz algorithm for $Aw = b$, which can be derived either as CD applied to the dual formulation (10) or as SG applied to the sum-of-squares problem

$$\min_w \frac{1}{2} \|Aw - b\|_2^2 = \frac{1}{2} \sum_{i=1}^{m} (A_i w - b_i)^2. \tag{17}$$

CD is related in an obvious way to the Gauss–Seidel method for $n \times n$ systems of linear equations, which adjusts the i_k variable to ensure satisfaction of the i_k equation, at iteration k. (Successive over-relaxation (SOR) modifies this approach by scaling each Gauss–Seidel step by a factor $(1 + \omega)$ for some constan $\omega \in [0, 1)$, chosen so as to improve the convergence rate.) Standard Gauss–Seidel and SOR use the cyclic choice of coordinates (11), whereas a random choice of i_k would correspond to "randomized" versions of these methods. To make the connections more explicit: The Gauss–Seidel method applied to the normal equations for (8)—that is, $A^T A w = A^T b$—is equivalent

to applying Algorithm 1 to the least-squares problem (17), when the steplength α_k is chosen to minimize the objective exactly along the given coordinate direction. SOR also corresponds to Algorithm 1, with α_k chosen to be a factor $(1 + \omega)$ times the exact minimum. These equivalences allow the results of Sect. 3 to be used to derive convergence rates for Gauss–Seidel applied to the normal equations, including linear convergence when $A^T A$ is nonsingular. Note that these results do not require feasibility of the original Eq. (8).

2 Applications

We mention here several applications of CD methods to practical problems, some dating back decades and others relatively new. Our list is necessarily incomplete, but it attests to the popularity of CD in a wide variety of application communities.

Bouman and Sauer [7] discuss an application to positron emission tomography (PET) in which the objective has the form (2) where f is smooth and convex and Ω is a sum of terms of the form $|x_j - x_l|^q$ for some pairs of components (j, l) of x and some $q \in [1, 2]$. Ye et al. [56] apply a similar method to a different objective arising from optical diffusion tomography.

Liu et al. [26] describe a block CD approach for linear least squares plus a regularization function consisting of a sum of ℓ_∞ norms of subvectors of x. The technique is applied to semantic basis discovery, which learns from data how to identify and classify the functional MRI response of a person's brain when they hear certain English words.

Canutescu and Dunbrack [11] describe a cyclic CD method for determining protein structure, adjusting the dihedral angles in a protein chain so that the atom at the end of the chain comes close to a specified position in space.

Florian and Chen [17] recover origin-destination matrices from observed traffic flows by alternately solving a bilevel optimization problem over two blocks of variables: the origin-destination demands and the proportion of each origin-destination flow assigned to each arc in the network.

Breheny and Huang [10] discuss CD for linear and logistic regression with nonconvex separable regularization terms, reporting results for genetic association and gene expression studies. The SparseNet algorithm [33] applied to problems with these same nonconvex separable regularizers uses warm-started cyclic CD as an inner loop to solve a sequence of problems in which the regularization parameter λ in (2) and the parameters defining concavity of the regularization functions are varied.

Friedman et al. [18] propose a block CD algorithm for estimating a sparse inverse covariance matrix, given a sample covariance matrix S and taking the variable in their formulation to be a modification W of S, such that W^{-1} is sparse. The resulting "graphical lasso" algorithm cycles through the rows/columns of W (in the style of block CD), solving a standard lasso problem to calculate each update. The same authors [19] apply CD to generalized linear models such as linear least squares and logistic regression, with convex regularization terms. Their framework include such formulations as lasso, graphical lasso, elastic net, and the Dantzig selector, and is implemented in the package `glmnet`.

Chang et al. [12] use cyclic and stochastic CD to solve a squared-loss formulation of the support vector machine (SVM) problem in machine learning, that is,

$$\min_{w} \sum_{i=1}^{m} \max\left(1 - y_i x_i^T w, 0\right)^2 + \frac{\lambda}{2} w^T w. \tag{18}$$

where $(x_i, y_i) \in \mathbb{R}^N \times \{0, 1\}$ are feature vector/label pairs and λ is a regularization parameter. This problem is an important instance of the ERM form (5). In the best known early application of CD to SVM, Platt [41] deals with a hinge-loss formulation of SVM, which is identical to (18) except that the square on each term of the summation is omitted. The dual of this problem has bounds on its variables along with a single linear constraint. Platt's procedure SMO (for "sequential minimal optimization"), applied to the dual, changes two variables at a time, with the variable pair chosen according to a "greedy" criterion and the search direction chosen to maintain feasibility of the linear constraint.

Sardy et al. [49] consider the basis-pursuit formulation of wavelet denoising:

$$\min_{x} \frac{1}{2}\|\Phi x - y\|_2^2 + \lambda \|x\|_1.$$

This formulation is equivalent to the well known lasso of Tibshirani [53] and has become famous because of its applicability to sparse recovery and compressed sensing. Although this formulation fits the ERM framework (5) and could thus be dualized before applying CD, the approach of [49] applies block CD directly to the primal formulation.

Applications of block CD approaches to transceiver design for cellular networks and to tensor factorization are discussed in Razaviyayn [44, Section 8].

Finally, we mention several popular problem classes and algorithms that can be interpreted as CD algorithms, but for which such an interpretation may not be particularly helpful in understanding the performance of the algorithm. First, we consider low-rank matrix completion problems in which we are presented with limited information about a rectangular matrix $M \in \mathbb{R}^{m \times n}$ and seek matrices $U \in \mathbb{R}^{n \times r}$ and $V \in \mathbb{R}^{m \times r}$ (with r small) such that UV^T is consistent with the observations of M. When the observations satisfy a restricted isometry property (an assumption commonly made in compressed sensing; see [45, Definition 3.1] for a definition that applies to matrix completion), the block CD approach of Jain et al. [21, Algorithm 1] converges to a solution. This approach defines the objective to be the least-squares fit between the observations and their predicted values according to the product UV^T—a function that is nonconvex with respect to (U, V)—and minimizes alternately over U and V, respectively. Standard analysis of CD for nonconvex functions would yield at best stationarity of accumulation points, but much stronger results are attained in [21] because of special assumptions that are made on the problem in this paper.

Second, we consider the "alternating-direction method of multipliers" (ADMM) [8,13], which has gained great currency in the past few years because of its usefulness in solving regularized problems in statistics and machine learning, and in designing parallel algorithms. Each major iteration of ADMM consists of an (approximate)

minimization of the augmented Lagrangian function for a constrained optimization problem over each block of primal variables in turn, followed by an update to the Lagrange multiplier estimates. It might seem appealing to do multiple cycles of updating the primal variable blocks, in the manner of cyclic block CD, thus finding a better approximation to the solution of each subproblem over *all* primal variables and moving the method closer to the standard augmented Lagrangian approach. Eckstein and Yao [14] show, however, that this "approximate augmented Lagrangian" approach has a fundamentally different theoretical interpretation from ADMM, and a computational comparison between the two approaches [14, Section 5] appears to show an advantage for ADMM.

3 Coordinate descent: algorithms, convergence, implementations

We now describe the most important variants of CD and present their convergence properties, including the proofs of some fundamental results. We also discuss the implementation of accelerated CD methods for problems of the form (7) and for the Kaczmarz algorithm for $Aw = b$. As mentioned in the introduction, we deal with the most elementary framework possible, to expose the essential properties of the methods.

3.1 Powell's example

We start with a simple but intriguing example due to Powell [43, formula (2)] of a function in $\mathbb{R}^3$ for which cyclic CD fails to converge to a stationary point. The nonconvex, continuously differentiable function $f : \mathbb{R}^3 \to \mathbb{R}$ is defined as follows:

$$f(x_1, x_2, x_3) = -(x_1x_2 + x_2x_3 + x_1x_3) + \sum_{i=1}^{3} (|x_i| - 1)_+^2. \tag{19}$$

It has minimizers at the corners $(1, 1, 1)^T$ and $(-1, -1, -1)^T$ of the unit cube, but CD with exact minimization, started near (but just outside of) one of the other vertices of the cube cycles around the neighborhoods of six points that are close to the six non-optimal vertices (Fig. 1). Powell shows that the cyclic nonconvergence behavior is rather special and is destroyed by small perturbations on this particular example, and we can note that a randomized CD method applied to this example would be expected to converge to the vicinity of a solution within a few steps. Still, this example and others in [43] make it clear that we cannot expect a general convergence result for nonconvex functions, of the type that are available for full-gradient descent. Results are available for the nonconvex case under certain additional assumptions that still admit interesting applications. Bertsekas [4, Proposition 2.7.1] describes convergence of a cyclic approach applied to nonconvex problems, under the assumption that the minimizer along any coordinate direction from any point x is unique. More recent work [1,6] focuses on CD with two blocks of variables, applied to functions that satisfy the so-called Kurdyka-Łojasiewicz (KL) property, such as semi-algebraic functions.

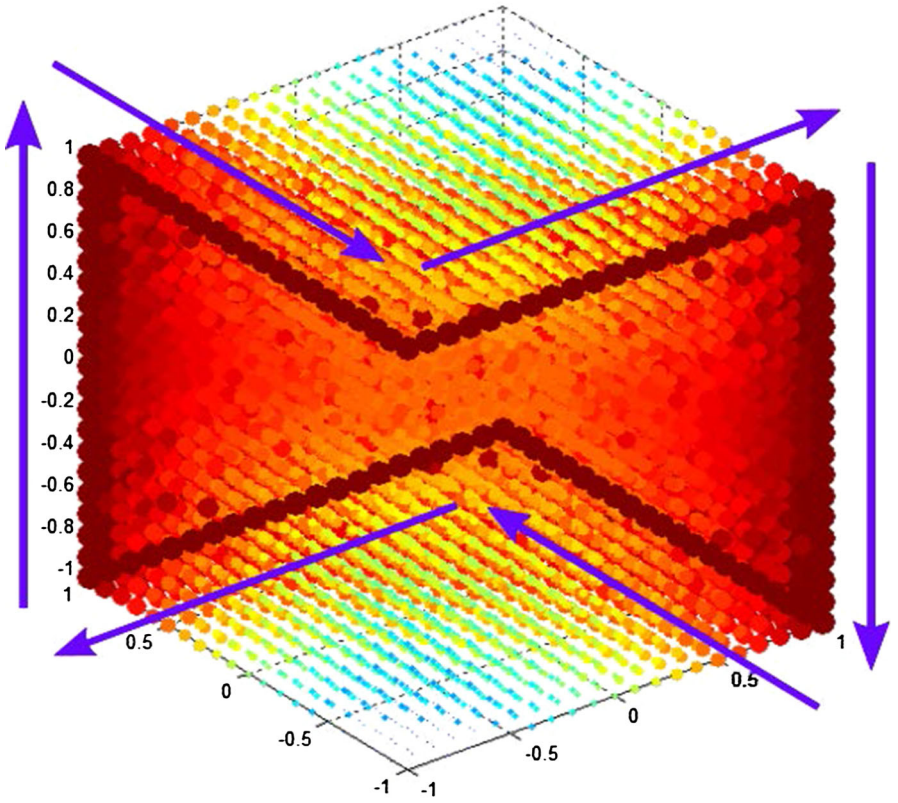

Fig. 1 Example of Powell [43] showing nonconvergence of cyclic CD

Convergence of subsequences or the full sequence $\{x^k\}$ to stationary points can be proved in this setting.

3.2 Assumptions and notation

For most of this section, we focus on the unconstrained problem (1), where the objective f is *convex* and Lipschitz continuously differentiable. In some places, we assume strong convexity with respect to the Euclidean norm, that is, existence of a modulus of convexity $\sigma > 0$ such that

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\sigma}{2}\|y - x\|_2^2, \quad \text{for all } x, y. \tag{20}$$

(Henceforth, we use $\|\cdot\|$ to denote the Euclidean norm $\|\cdot\|_2$, unless otherwise specified.) We define Lipschitz constants that are tied to the component directions, and are key to the algorithms and their analysis. The first set of such constants are the *component Lipschitz constants*, which are positive quantities L_i such that for all $x \in \mathbb{R}^n$ and all

$t \in \mathbb{R}$ we have

$$|[\nabla f(x + te_i)]_i - [\nabla f(x)]_i| \leq L_i |t|, \tag{21}$$

We define the *coordinate Lipschitz constant* $L_{\max}$ to be such that

$$L_{\max} = \max_{i=1,2,\ldots,n} L_i. \tag{22}$$

The standard Lipschitz constant L is such that

$$\|\nabla f(x + d) - \nabla f(x)\| \leq L\|d\|, \tag{23}$$

for all x and d of interest. By referring to relationships between norm and trace of a symmetric matrix, we can assume that $1 \leq L/L_{\max} \leq n$. (The upper bound is achieved when $f(x) = e(e^T x)$, for $e = (1, 1, \ldots, 1)^T$.) We also define the *restricted Lipschitz constant* L_{res} such that the following property is true for all $x \in \mathbb{R}^n$, all $t \in \mathbb{R}$, and all $i = 1, 2, \ldots, n$:

$$\|\nabla f(x + te_i) - \nabla f(x)\| \leq L_{\text{res}} |t|. \tag{24}$$

Clearly, $L_{\text{res}} \leq L$. The ratio

$$\Lambda := L_{\text{res}}/L_{\max} \tag{25}$$

is important in our analysis of asynchronous parallel algorithms in Sect. 4. In the case of f convex and twice continuously differentiable, we have by positive semidefiniteness of the $\nabla^2 f(x)$ at all x that

$$|[\nabla^2 f(x)]_{ij}| \leq \left([\nabla^2 f(x)]_{ii}[\nabla^2 f(x)]_{jj}\right)^{1/2},$$

from which we can deduce that

$$1 \leq \Lambda \leq \sqrt{n}.$$

However, we can derive stronger bounds on Λ for functions f in which the coupling between components of x is weak. In the extreme case in which f is separable, we have $\Lambda = 1$. The coordinate Lipschitz constant corresponds $L_{\max}$ to the maximal absolute value of the diagonal elements of the Hessian $\nabla^2 f(x)$, while the restricted Lipschitz constant L_{res} is related to the maximal column norm of the Hessian. Therefore, if the Hessian is positive semidefinite and diagonally dominant, the ratio Λ is at most 2.

The following assumption is useful in the remainder of the paper.

Assumption 1 The function f in (1) is convex and uniformly Lipschitz continuously differentiable, and attains its minimum value f^* on a set $\mathcal{S}$. There is a finite R_0 such that the level set for f defined by x^0 is bounded, that is,

$$\max_{x^* \in \mathcal{S}} \max_x \{\|x - x^*\| \ : \ f(x) \leq f(x^0)\} \leq R_0. \tag{26}$$

3.3 Randomized algorithms

In randomized CD algorithms, the update component i_k is chosen randomly at each iteration. In Algorithm 3 we consider the simplest variant in which each i_k is selected from $\{1, 2, \ldots, n\}$ with equal probability, independently of the selections made at previous iterations. (We can think of this scheme as "sampling with replacement" from the set $\{1, 2, \ldots, n\}$.)

Algorithm 3 Randomized CD for (1)

Choose $x^0 \in \mathbb{R}^n$;
Set $k \leftarrow 0$;
repeat
  Choose index i_k with uniform probability from $\{1, 2, \ldots, n\}$, independently of choices at prior iterations;
  Set $x^{k+1} \leftarrow x^k - \alpha_k [\nabla f(x^k)]_{i_k} e_{i_k}$ for some $\alpha_k > 0$;
  $k \leftarrow k + 1$;
until termination test satisfied;

We denote expectation with respect to a single random index i_k by $E_{i_k}(\cdot)$, while $E(\cdot)$ denotes expectation with respect to all random variables $i_0, i_1, i_2, \ldots$.

We prove a convergence result for the randomized algorithm, for the simple steplength choice $\alpha_k \equiv 1/L_{\max}$. (The proof is a simplified version of the analysis in Nesterov [37, Section 2]. A result similar to (27) is proved by Shalev-Schwartz and Tewari [50] for certain types of ℓ_1-regularized problems.)

Theorem 1 *Suppose that* Assumption 1 *holds. Suppose that* $\alpha_k \equiv 1/L_{\max}$ *in Algorithm 3. Then for all* $k > 0$ *we have*

$$E(f(x^k)) - f^* \leq \frac{2nL_{\max}R_0^2}{k}. \tag{27}$$

When $\sigma > 0$ *in* (20)*, we have in addition that*

$$E\left(f(x^k)\right) - f^* \leq \left(1 - \frac{\sigma}{nL_{\max}}\right)^k (f(x^0) - f^*). \tag{28}$$

Proof By application of Taylor's theorem, and using (21) and (22), we have

$$\begin{aligned} f(x^{k+1}) &= f\left(x^k - \alpha_k \left[\nabla f(x^k)\right]_{i_k} e_{i_k}\right) \\ &\leq f(x^k) - \alpha_k \left[\nabla f(x^k)\right]_{i_k}^2 + \frac{1}{2}\alpha_k^2 L_{i_k} \left[\nabla f(x^k)\right]_{i_k}^2 \\ &\leq f(x^k) - \alpha_k \left(1 - \frac{L_{\max}}{2}\alpha_k\right) \left[\nabla f(x^k)\right]_{i_k}^2 \\ &= f(x^k) - \frac{1}{2L_{\max}} \left[\nabla f(x^k)\right]_{i_k}^2, \end{aligned} \tag{29}$$

where we substituted the choice $\alpha_k = 1/L_{\max}$ in the last equality. Taking the expectation of both sides of this expression over the random index i_k, we have

$$\begin{aligned} E_{i_k} f(x^{k+1}) &\leq f(x^k) - \frac{1}{2L_{\max}} \frac{1}{n} \sum_{i=1}^{m} \left[\nabla f(x^k)\right]_i^2 \\ &= f(x^k) - \frac{1}{2nL_{\max}} \|\nabla f(x^k)\|^2. \end{aligned} \tag{30}$$

(We used here the facts that x^k does not depend on i_k, and that i_k was chosen from among $\{1, 2, \ldots, n\}$ with equal probability.) We now subtract $f(x^*)$ from both sides this expression, take expectation of both sides with respect to *all* random variables $i_0, i_1, \ldots$, and use the notation

$$\phi_k := E(f(x^k)) - f^*. \tag{31}$$

to obtain

$$\phi_{k+1} \leq \phi_k - \frac{1}{2nL_{\max}} E\left(\|\nabla f(x^k)\|^2\right) \leq \phi_k - \frac{1}{2nL_{\max}} \left[E(\|\nabla f(x^k)\|)\right]^2. \tag{32}$$

(We used Jensen's Inequality in the second inequality.) By convexity of f we have for any $x^* \in \mathcal{S}$ that

$$f(x^k) - f^* \leq \nabla f(x^k)^T (x^k - x^*) \leq \|\nabla f(x^k)\| \|x^k - x^*\| \leq R_0 \|\nabla f(x^k)\|,$$

where the final inequality is because $f(x^k) \leq f(x^0)$, so that x^k is in the level set in (26). By taking expectations of both sides, we obtain

$$E(\|\nabla f(x^k)\|) \geq \frac{1}{R_0} \phi_k.$$

When we substitute this bound into (32), and rearrange, we obtain

$$\phi_k - \phi_{k+1} \geq \frac{1}{2nL_{\max}} \frac{1}{R_0^2} \phi_k^2.$$

We thus have

$$\frac{1}{\phi_{k+1}} - \frac{1}{\phi_k} = \frac{\phi_k - \phi_{k+1}}{\phi_k \phi_{k+1}} \geq \frac{\phi_k - \phi_{k+1}}{\phi_k^2} \geq \frac{1}{2nL_{\max} R_0^2}.$$

By applying this formula recursively, we obtain

$$\frac{1}{\phi_k} \geq \frac{1}{\phi_0} + \frac{k}{2nL_{\max} R_0^2} \geq \frac{k}{2nL_{\max} R_0^2},$$

so that (27) holds, as claimed.

In the case of f strongly convex with modulus $\sigma > 0$, we have by taking the minimum of both sides with respect to y in (20), and setting $x = x^k$, that

$$f^* \geq f(x^k) - \frac{1}{2\sigma}\|\nabla f(x^k)\|^2.$$

By using this expression to bound $\|\nabla f(x^k)\|^2$ in (32), we obtain

$$\phi_{k+1} \leq \phi_k - \frac{\sigma}{nL_{\max}}\phi_k = \left(1 - \frac{\sigma}{nL_{\max}}\right)\phi_k.$$

Recursive application of this formula leads to (28).

Note that the same convergence expressions can be obtained for more refined choices of steplength α_k, by making minor adjustments to the logic in (29). For example, the choice $\alpha_k = 1/L_{i_k}$ leads to the same bounds (27) and (28). The same bounds hold too when α_k is the exact minimizer of f along the coordinate search direction; we modify the logic in (29) for this case by taking the minimum of all expressions with respect to α_k, and use the fact that $\alpha_k = 1/L_{\max}$ is in general a suboptimal choice.

We can compare (27) with the corresponding result for full-gradient descent with constant steplength $\alpha_k = 1/L$ [where L is from (23)]. The iteration

$$x^{k+1} = x^k - \frac{1}{L}\nabla f(x^k)$$

leads to a convergence expression

$$f(x^k) - f^* \leq \frac{2LR_0^2}{k} \tag{33}$$

(see, for example, [36]). Since, as we have noted, L can be as large as $nL_{\max}$, the bound in this expression may be equivalent to (27) in extreme cases. More typically, these two Lipschitz constants are comparable in size, and the appearance of the additional factor n in (27) indicates that we pay a price in terms of slower convergence for using only one component of $\nabla f(x^k)$, rather than the full vector.

Expected linear convergence rates have been proved under assumptions weaker than strong convexity; see for example the "essential strong convexity" property of [28], the "optimal strong convexity" property of [27], the "generalized error bound" property of [34], and [55, Assumption 2], which concerns linear growth in a measure of the gradient with distance from the solution set.

A variant on Algorithm 3 uses "sampling without replacement." Here the computation proceeds in "epochs" of n consecutive iterations. At the start of each epoch, the set $\{1, 2, \ldots, n\}$ is shuffled. The iterations then proceed by setting i_k to each entry in turn from the ordered set. This kind of randomization has been shown in several contexts to be superior to the sampling-with-replacement scheme analyzed above, but a theoretical understanding of this phenomenon remains elusive.

3.3.1 Randomized Kaczmarz Algorithm

It is worth proving an expected linear convergence result for the Kaczmarz iteration (16) for linear equations $Aw = b$ as a separate, more elementary analysis. In one sense, the result is a special case of Theorem 1 since, as we showed above, the iteration (16) is obtained by applying Algorithm 3 to the dual formulation (10). In another sense, the result is stronger, since we obtain a linear rate of convergence without requiring strong convexity of the objective (10), that is, the system $Aw = b$ is allowed to have multiple solutions.

We denote by $\lambda_{\min,\mathrm{nz}}$ the minimum nonzero eigenvalue of AA^T and let $P(\cdot)$ denote projection onto the solution set of $Aw = b$. We have

$$\begin{aligned}\|w^{k+1} - P(w^{k+1})\|^2 &\le \|w^k - A_{i_k}^T(A_{i_k}w^k - b_{i_k}) - P(w^k)\|^2 \\ &= \frac{1}{2}\|w^k - P(w^k)\|^2 - (A_{i_k}w^k - b_{i_k})^2,\end{aligned}$$

where we have used normalization of the rows (14) and the fact that $A_{i_k}P(x^k) = b_{i_k}$. By taking expectations of both sides with respect to i_k, we have

$$\begin{aligned}E_{i_k}\|w^{k+1} - P(w^{k+1})\|^2 &\le \|w^k - P(w^k)\|^2 - E_{i_k}(A_{i_k}w^k - b_{i_k})^2 \\ &= \frac{1}{2}\|w^k - P(w^k)\|^2 - \frac{1}{m}\|Aw^k - b\|^2 \\ &\le \left(1 - \frac{\lambda_{\min,\mathrm{nz}}}{m}\right)\|w^k - P(w^k)\|^2.\end{aligned}$$

By taking expectations of both sides with respect to all random variables $i_0, i_1, \ldots$, and proceeding recursively, we obtain

$$E\|w^k - P(w^k)\|^2 \le \left(1 - \frac{\lambda_{\min,\mathrm{nz}}}{m}\right)^k \|w^0 - P(w^0)\|^2.$$

(This analysis is slightly generalized from Strohmer and Vershynin [52] to allow for nonunique solutions of $Aw = b$; see also [24].)

3.4 Accelerated randomized algorithms

The accelerated randomized algorithm, specified here as Algorithm 4, was proposed by Nesterov [37]. It assumes that an estimate is available of modulus of strong convexity $\sigma \ge 0$ from (20), as well as estimates of the component-wise Lipschitz constants L_i from (21). (The algorithm remains valid if we simply use $L_{\max}$ in place of L_{i_k} for all k.)

The approach is a close relative of the accelerated (full-)gradient methods that have become extremely popular in recent years. These methods have their origin in a 1983 paper of Nesterov [35] and owe much of their recent popularity to a recent incarnation known as FISTA [2] and an exposition in Nesterov's 2004 monograph [36], as well

Algorithm 4 Accelerated Randomized CD for (1)

Choose $x^0 \in \mathbb{R}^n$;
Set $k \leftarrow 0$, $v^0 \leftarrow x^0$, $\gamma_{-1} \leftarrow 0$;
repeat
Choose γ_k to be the larger root of

$$\gamma_k^2 - \frac{\gamma_k}{n} = \left(1 - \frac{\gamma_k \sigma}{n}\right)\gamma_{k-1}^2.$$

Set

$$\alpha_k \leftarrow \frac{n - \gamma_k \sigma}{\gamma_k (n^2 - \sigma)}, \quad \beta_k \leftarrow 1 - \frac{\gamma_k \sigma}{n}; \tag{34}$$

Set $y^k \leftarrow \alpha_k v^k + (1 - \alpha_k) x^k$;
Choose index $i_k \in \{1, 2, \ldots, n\}$ with uniform probability and set $d^k = [\nabla f(y^k)]_{i_k} e_{i_k}$;
Set $x^{k+1} \leftarrow y^k - (1/L_{i_k}) d^k$;
Set $v^{k+1} \leftarrow \beta_k v^k + (1 - \beta^k) y^k - (\gamma_k / L_{i_k}) d^k$;
$k \leftarrow k + 1$;
until termination test satisfied;

as ease of implementation and good practical experience. In their use of momentum in the choice of step—the search direction combines new gradient information with the previous search direction—these methods are also related to such other classical techniques as the heavy-ball method (see [42]) and conjugate gradient methods.

Nesterov [37, Theorem 6] proves the following convergence result for Algorithm 4.

Theorem 2 *Suppose that* Assumption 1 *holds, and define*

$$S_0 := \sup_{x^* \in \mathcal{S}} L_{\max} \|x^0 - x^*\|^2 + \left(f(x^0) - f^*\right)/n^2.$$

Then for all $k \geq 0$ *we have*

$$\begin{aligned} & E(f(x^k)) - f^* \\ & \leq S_0 \frac{\sigma}{L_{\max}} \left[\left(1 + \frac{\sqrt{\sigma / L_{\max}}}{2n}\right)^{k+1} - \left(1 - \frac{\sqrt{\sigma / L_{\max}}}{2n}\right)^{k+1} \right]^{-2} \qquad (35) \\ & \leq S_0 \left(\frac{n}{k+1}\right)^2. \qquad (36) \end{aligned}$$

In the strongly convex case $\sigma > 0$, the term $(1 + \sqrt{\sigma / L_{\max}}/(2n))^{k+1}$ eventually dominates the second term in brackets in (35), so that the linear convergence rate suggested by this expression is significantly faster than the corresponding rate (28) for Algorithm 3. Essentially, the measure $\sigma / L_{\max}$ of conditioning in (28) is replaced by its square root in (35), suggesting a decrease by a factor of $\sqrt{L_{\max}/\sigma}$ in the number of iterations required to meet a specified error tolerance. In the sublinear rate bound (36), which holds even for weakly convex f, the $1/k$ bound of (27) is replaced by a $1/k^2$

factor, implying a reduction from $O(1/\epsilon)$ to $O(1/\sqrt{\epsilon})$ in the number of iterations required to meet a specified error tolerance.

3.5 Efficient implementation of the accelerated algorithm

One fact detracts from the appeal of accelerated CD methods over standard methods: the higher cost of each iteration of Algorithm 5. Both standard and accelerated variants require calculation of one element of the gradient, but Algorithm 3 requires an update of just a single component of x, whereas Algorithm 4 also requires manipulation of the generally dense vectors y and v. Moreover, the gradient is evaluated at x^k in Algorithm 3, where the argument changes by only one component from the prior iteration, a fact that can be exploited in several contexts. In Algorithm 4, the argument y^k for the gradient changes more extensively from one iteration to the next, making it less obvious whether such economies are available. However, by using a change of variables due to Lee and Sidford [23], it is possible to implement the accelerated randomized CD approach efficiently for problems with certain structure, including the linear system $Aw = b$ and certain problems of the form (5).

Algorithm 5 Accelerated Randomized Kaczmarz for (8), (14)

Choose $w^0 \in \mathbb{R}^n$;
Set $k \leftarrow 0$, $\tilde{v}^0 \leftarrow w^0$, $\gamma_{-1} \leftarrow 0$;
repeat
Choose γ_k to be the larger root of

$$\gamma_k^2 - \frac{\gamma_k}{n} = \left(1 - \frac{\gamma_k \sigma}{n}\right)\gamma_{k-1}^2.$$

Set

$$\alpha_k \leftarrow \frac{n - \gamma_k \sigma}{\gamma_k (n^2 - \sigma)}, \quad \beta_k \leftarrow 1 - \frac{\gamma_k \sigma}{n}; \tag{37}$$

Set $\tilde{y}^k \leftarrow \alpha_k \tilde{v}^k + (1 - \alpha_k) w^k$;
Choose index $i_k \in \{1, 2, \ldots, m\}$ with uniform probability and set $\tilde{d}^k = (A_{i_k} \tilde{y}^k - b_{i_k}) A_{i_k}^T$;
Set $w^{k+1} \leftarrow \tilde{y}^k - \tilde{d}^k$;
Set $\tilde{v}^{k+1} \leftarrow \beta_k \tilde{v}^k + (1 - \beta^k)\tilde{y}^k - \gamma_k \tilde{d}^k$;
$k \leftarrow k + 1$;
until termination test satisfied;

We explain the Lee-Sidford technique in the context of the Kaczmarz algorithm for (8), assuming normalization of the rows of A (14). As we explained in (16), the Kaczmarz algorithm is obtained by applying CD to the dual formulation (10) with variables x, but operating in the space of "primal" variables w using the transformation $w = A^T x$. If we apply the transformations $\tilde{v}^k = A^T v^k$ and $\tilde{y}^k = A^T y^k$ to the other vectors in Algorithm 4, and use the fact of normalization (14) (and hence $(AA^T)_{ii} = 1$ for all $i = 1, 2, \ldots, m$) to note that $L_i \equiv 1$ in (21), we obtain Algorithm 5.

When the matrix A is dense, there is only a small factor of difference between the per-iteration workload of the standard Kaczmarz algorithm and its accelerated variant,

Algorithm 5. Both require $O(m+n)$ operations per iteration. However, when A is sparse, the computational difference between the two algorithms becomes substantial. At iteration k, the standard Kaczmarz algorithm requires computation proportion to a small multiple of the number of nonzeros in row A_{i_k} (which we denote by $|A_{i_k}|$). Meanwhile, iteration k of Algorithm 5 requires manipulation of the dense vectors $\tilde{v}^k$ and $\tilde{y}^k$—both $O(n)$ processes—and the benefits of sparsity are lost. This apparent defect was partly remedied in [29] by "caching" the updates to these vectors, resulting in a number of cycles within which updates gradually "fill in." The more effective approach of [23] performs a change of variables from $\tilde{v}^k$ and $\tilde{y}^k$ to two other vectors $\hat{v}^k$ and $\hat{y}^k$ that *can* be updated in $O(|A_{i_k}|)$ operations. To describe this representation, we start by noting that if we substitute for w^k and w^{k+1} in the formulas of Algorithm 5, we obtain the updates to $\tilde{v}^k$ and $\tilde{y}^k$ in the following form:

$$\left[\tilde{v}^{k+1}\ \tilde{y}^{k+1}\right] = \left[\tilde{v}^k\ \tilde{y}^k\right] R_k - S_k, \tag{38}$$

where

$$R_k := \begin{bmatrix} \beta_k & \alpha_{k+1}\beta_k \\ (1-\beta_k) & (1-\alpha_{k+1}\beta_k) \end{bmatrix},$$
$$S_k := \left(A_{i_k}\tilde{y}^k - b_{i_k}\right) A_{i_k}^T \left[\gamma_k\ (1-\alpha_{k+1}+\alpha_{k+1}\gamma_k)\right].$$

Note that R_k is a 2×2 matrix while S_k is an $n\times 2$ matrix with nonzeros only in those rows for which $A_{i_k}^T$ has a nonzero entry. We define a change of variables based on another 2×2 matrix B_k, as follows:

$$\left[\tilde{v}^k\ \tilde{y}^k\right] = \left[\hat{v}^k\ \hat{y}^k\right] B_k, \tag{39}$$

where we initialize with $B_0 = I$. By substituting this representation into (38), we obtain

$$\left[\hat{v}^{k+1}\ \hat{y}^{k+1}\right] B_{k+1} = \left[\hat{v}^k\ \hat{y}^k\right] B_k R_k - S_k,$$

so we can maintain validity of the representation (39) at iteration $k+1$ by setting

$$B_{k+1} := B_k R_k, \quad \left[\hat{v}^{k+1}\ \hat{y}^{k+1}\right] := \left[\hat{v}^k\ \hat{y}^k\right] - S_k B_{k+1}^{-1}. \tag{40}$$

The computations in (40) can be performed in $O(|A_{i_k}|)$ operations, and can replace the relatively expensive computations of $\tilde{y}^k$ and $\tilde{v}^{k+1}$ in Algorithm 5. The only other operation of note in this algorithm—computation of $A_{i_k}\tilde{y}^k - b_{i_k}$—can also be performed in $O(|A_{i_k}|)$ operations using the $(\hat{v}^k, \hat{y}^k)$ representation, by noting from (39) that

$$A_{i_k}\tilde{y}^k = (A_{i_k}\hat{v}^k)(B_k)_{12} + (A_{i_k}\hat{y}^k)(B_k)_{22}.$$

This efficient implementation can be extended to the dual empirical risk minimization problem (7) for certain choices of regularization function $g(\cdot)$, for example,

$g(z) = \|z\|^2/2$; see [25]. As pointed out in [23], the key requirement for the efficient scheme is that the gradient term $[\nabla f(y^k)]_{i_k}$ can be evaluated efficiently after an update to the two vectors in the alternative representation of y^k, and to the two coefficients in this representation. Another variant of this implementation technique appears in [16, Section 5].

3.6 Cyclic variants

We have the following result from [3] for the cyclic variant of Algorithm 1.

Theorem 3 *Suppose that* Assumption 1 *holds. Suppose that* $\alpha_k \equiv 1/L_{\max}$ *in* Algorithm 1*, with the index* i_k *at iteration* k *chosen according to the cyclic ordering* (11) *(with* $i_0 = 1$*). Then for* $k = n, 2n, 3n, \ldots$*, we have*

$$f(x^k) - f^* \leq \frac{4nL_{\max}(1 + nL^2/L_{\max}^2)R_0^2}{k+8}. \tag{41}$$

When $\sigma > 0$ *in the strong convexity condition* (20)*, we have in addition for* $k = n, 2n, 3n, \ldots$ *that*

$$f(x^k) - f^* \leq \left(1 - \frac{\sigma}{2L_{\max}(1 + nL^2/L_{\max}^2)}\right)^{k/n} \left(f(x^0) - f^*\right). \tag{42}$$

Proof The result (41) follows from Theorems 3.6 and 3.9 in [3] when we note that (i) each iteration of Algorithm BCGD in [3] corresponds to a "cycle" of n iterations in Algorithm 1; (ii) we update coordinates rather than blocks, so that the parameter p in [3] is equal to n; (iii) we set $\bar{L}_{\max}$ and $\bar{L}_{\min}$ in [3] both to $L_{\max}$.

Comparing the complexity bounds for the cyclic variant with the corresponding bounds proved in Theorem 1 for the randomized variant, we see that since $L \geq L_{\max}$ in general, the numerator in (41) is $O(n^2)$, in contrast to $O(n)$ term in (27). A similar factor of n in seen in comparing (28) to (42), when we note that $(1-\epsilon)^{1/n} \approx 1 - \epsilon/n$ for small values of ϵ. The bounds in Theorem 3 are deterministic, however, rather than being bounds on expected nonoptimality, as in Theorem 1.

We noted in Sect. 3.2 that the ratio $L/L_{\max}$ lies in the interval $[1, n]$ when f is a convex quadratic function and both parameters are set to their best values. Lower values of this ratio are attained on functions that are "more decoupled" and larger values attained when there is a greater dependence between the coordinates. Larger values lead to weaker bounds in Theorem 3, which accords with our intuition; we expect CD methods to require more iterations to resolve the coupling of the coordinates.

We are free to make other, larger choices of $L_{\max}$; they need only satisfy the conditions (21) and (22). Larger values of $L_{\max}$ lead to shorter steps $\alpha_k = 1/L_{\max}$ and different complexity expressions. For $L_{\max} = L$, for example, the bound in (41) becomes

$$\frac{4n(n+1)LR_0^2}{k+8},$$

which is worse by a factor of approximately $2n^2$ than the bound (33) for the full-step gradient descent approach. For $L_{\max} = \sqrt{n}L$, we obtain

$$\frac{8n^{3/2}LR_0^2}{k+8},$$

which still trails (33) by a factor of $4n^{3/2}$.

3.7 Extension to separable regularized case

In this section we consider the separable regularized formulation (2), (3) where f is smooth and strongly convex, and each Ω_i, $i = 1, 2, \ldots, n$ is convex. We prove a result similar to the second part of Theorem 1 for a randomized version of Algorithm 2. The proof is a simplified version of the analysis from [47]. It makes use of the following assumption.

Assumption 2 The function f in (2) is uniformly Lipschitz continuously differentiable and strongly convex with modulus $\sigma > 0$ [see (20)]. The functions Ω_i, $i = 1, 2, \ldots, n$ are convex. The function h in (2) attains its minimum value h^* at a unique point x^*.

Our result uses the coordinate Lipschitz constant $L_{\max}$ for f, as defined in (22). Note that the modulus of convexity σ for f is also the modulus of convexity for h. By elementary results for convex functions, we have

$$h(\alpha x + (1-\alpha)y) \leq \alpha h(x) + (1-\alpha)h(y) - \frac{1}{2}\sigma\alpha(1-\alpha)\|x-y\|^2. \tag{43}$$

Theorem 4 *Suppose that* Assumption 2 *holds. Suppose that the indices i_k in* Algorithm 2 *are chosen independently for each k with uniform probability from $\{1, 2, \ldots, n\}$, and that $\alpha_k \equiv 1/L_{\max}$. Then for all $k \geq 0$, we have*

$$E\left(h(x^k)\right) - h^* \leq \left(1 - \frac{\sigma}{nL_{\max}}\right)^k \left(h(x^0) - h^*\right). \tag{44}$$

Proof Define the function

$$H\left(x^k, z\right) := f\left(x^k\right) + \nabla f\left(x^k\right)^T \left(z - x^k\right) + \frac{1}{2}L_{\max}\|z - x^k\|^2 + \lambda\Omega(z),$$

and note that this function is separable in the components of z, and attains its minimum over z at the vector z^k whose i_k component is defined in Algorithm 2. Note by strong convexity (20) that

$$\begin{aligned} H\left(x^k, z\right) &\leq f(z) - \frac{1}{2}\sigma\|z - x^k\|^2 + \frac{1}{2}L_{\max}\|z - x^k\|^2 + \lambda\Omega(z) \\ &= h(z) + \frac{1}{2}(L_{\max} - \sigma)\|z - x^k\|^2. \end{aligned} \tag{45}$$

We have by minimizing both sides over z in this expression that

$$\begin{aligned} H\left(x^k, z^k\right) &= \min_z \; H\left(x^k, z\right) \\ &\le \min_z \; h(z) + \frac{1}{2}(L_{\max} - \sigma)\|z - x^k\|^2 \\ &\le \min_{\alpha \in [0,1]} \; h\left(\alpha x^* + (1-\alpha)x^k\right) + \frac{1}{2}(L_{\max} - \sigma)\alpha^2 \|x^k - x^*\|^2 \\ &\le \min_{\alpha \in [0,1]} \; \alpha h^* \\ &\quad + (1-\alpha)h(x^k) + \frac{1}{2}\left[(L_{\max} - \sigma)\alpha^2 - \sigma\alpha(1-\alpha)\right]\|x^k - x^*\|^2 \\ &\le \frac{\sigma}{L_{\max}} h^* + \left(1 - \frac{\sigma}{L_{\max}}\right) h(x^k), \end{aligned} \tag{46}$$

where we used (45) for the first inequality, (43) for the third inequality, and the particular value $\alpha = \sigma / L_{\max}$ for the fourth inequality (for which value the coefficient of $\|x^k - x^*\|^2$ vanishes). Taking the expected value of $h(x^{k+1})$ over the index i_k, we have

$$\begin{aligned} E_{i_k} h\left(x^{k+1}\right) &= \frac{1}{n}\sum_{i=1}^n \left[f(x^k + (z_i^k - x_i^k)e_i) + \lambda \Omega_i(z_i^k) + \lambda \sum_{j \ne i} \Omega_j(x_j^k) \right] \\ &\le \frac{1}{n}\sum_{i=1}^n \left\{ f(x^k) + [\nabla f(x^k)]_i (z_i^k - x_i^k) + \frac{1}{2} L_{\max}(z_i^k - x_i^k)^2 \right. \\ &\quad \left. + \lambda \Omega_i(z_i^k) + \lambda \sum_{j \ne i} \Omega_j(x_j^k) \right\} \\ &= \frac{n-1}{n} h(x^k) + \frac{1}{n}\left[f(x^k) + \nabla f(x^k)^T (z^k - x^k) \right. \\ &\quad \left. + \frac{1}{2} L_{\max} \|z^k - x^k\|^2 + \lambda \Omega(z^k) \right] \\ &= \frac{n-1}{n} h(x^k) + \frac{1}{n} H\left(x^k, z^k\right). \end{aligned}$$

By subtracting h^* from both sides of this expression, and using (46) to substitute for $H(x^k, z^k)$, we obtain

$$E_{i_k} h\left(x^{k+1}\right) - h^* \le \left(1 - \frac{\sigma}{n L_{\max}}\right)\left(h(x^k) - h^*\right).$$

By taking expectations of both sides of this expression with respect to the random indices $i_0, i_1, i_2, \ldots, i_{k-1}$, we obtain

$$E\left(h(x^{k+1})\right) - h^* \leq \left(1 - \frac{\sigma}{nL_{\max}}\right)\left(E(h(x^k)) - h^*\right).$$

The result follows from a recursive application of this formula.

A result similar to (27) can be proved for the case in which f is convex but not strongly convex, but there are a few technical complications, and we refer the reader to [47] for details.

An extension of the fixed-step approach to separable composite objectives (2), (3) with *nonconvex* smooth part f is discussed in [40], where it is shown that accumulation points of the sequence of iterates are stationary and that a measure of optimality decreases to zero at a sublinear $(1/k)$ rate.

3.8 Computational notes

A full computational comparison between variants of CD (and between CD and other methods) is beyond the scope of this paper. Nevertheless it is worth asking whether various aspects of the convergence analysis presented above—in particular, the distinction between CD variants—can be observed in practice. To this end, we used these methods to minimize a convex quadratic $f(x) = (1/2)x^T Qx$ (with Q symmetric and positive semidefinite) for which $x^* = 0$ and $f^* = 0$. We constructed Q by choosing an integer r from $1, 2, \ldots, n$ and parameters $\eta \in [0, 1]$ and $\zeta > 0$, and defining

$$Q := V_{r,\eta} \Sigma V_{r,\eta}^T + \zeta \mathbf{1}\mathbf{1}^T, \tag{47a}$$

$$V_{r,\eta} := \eta V + (1 - \eta) E_r, \tag{47b}$$

$$E_r := \left[I_{r\times r} \,|\, 0_{r\times(n-r)}\right]^T. \tag{47c}$$

where $V \in \mathbb{R}^{n\times r}$ is a random matrix with $r \leq n$ orthogonal columns, Σ is an $r \times r$ positive diagonal matrix whose diagonal elements were chosen from a log-uniform distribution to have a specified condition number (with maximum diagonal of 1), and $\mathbf{1}$ is the vector $(1, 1, \ldots, 1)^T$. For convenience, we normalized Q so that its maximum diagonal—and thus $L_{\max}$ (22)—is 1.

By choosing η and ζ appropriately, we can obtain a range of values for the quantities described in Sect. 3.2, which enter along with the smallest singular value into the convergence expression. For example, by setting $\zeta = 0$ and $\eta = 0$ we obtain a randomly oriented matrix, possibly singular, with a specified range of nonzero eigenvalues. Nonzero values of η and ζ induce different types of orientation bias. In particular, we see that Λ (25) increases toward its upper bound of $\sqrt{n}$ as ζ increases away from zero.

We tested three CD variants.

- CYCLIC: Cyclic CD, described in Sect. 3.6.
- IID: Randomized CD using sampling with replacement: Algorithm 3.

– EPOCHS: The "sampling without replacement" variant of Algorithm 3, described following the proof of Theorem 1.

For each variant, we tried both a fixed steplength $\alpha_k \equiv 1/L_{\max}$ and the optimal steplength $\alpha_k = 1/Q_{i_k,i_k}$. Thus, there were a total of six algorithmic variants tested.

The starting point x^0 was chosen randomly, with all components from the unit normal distribution $N(0, 1)$. The algorithms were terminated when the objective was reduced by a factor of 10^{-6} over its initial value $f(x^0)$.

The speed of convergence varied widely according to the problem construction parameters η, λ, and $\text{cond}(\Sigma)$, but we can make some general observations. First, on problems that are not well conditioned, the function values $f(x^k)$ decreased rapidly at first, then settled into a *linear* rate of decrease. This linear rate held even for problems in which Q was singular—a significant improvement over the sublinear rates predicted by the theory. Second, the EPOCHS variant of randomized CD tended to converge faster than the IID version, though rarely more than twice as fast. Third, the use of the optimal step was usually better than the fixed step (with sometimes up to six times fewer iterations), but this was by no means always the case. Fourth, while there were extensive regimes of parameter values in which all six variants performed similarly, there were numerous "stressed" settings in which the CYCLIC variants are much slower than the randomized variants, by factors of 10 or more.

4 Parallel CD algorithms

CD methods lend themselves to different kinds of parallel implementation. Even basic algorithm frameworks such as Algorithm 1 may be amenable to application-specific parallelism, when the computations involved in evaluating a single element of the gradient vector are substantial enough to be spread out across cores of a multicore computer. We concern ourselves here with more generic forms of parallelism, which involve multiple instances of the basic CD algorithm, running in parallel on multiple processors.

We can distinguish different types of parallel CD algorithms. *Synchronous* algorithms are those that partition the computation into pieces that can be executed in parallel on multiple processors (or cores of a multicore machine), but that synchronize frequently across all processors, to ensure consistency of the information available to all processors at certain points in time. For example, each processor could update a subset of components of x in parallel (with the subsets being disjoint), and the synchronization step could ensure that the results of all updates are shared across all processors before further computation occurs. The synchronization step often detracts from the performance of algorithms, not only because some processors may be forced to idle while others complete their work, but also because the overheads associated with (hardware and software) locking of memory accesses can be high. Thus, *asynchronous* methods, which weaken or eliminate the requirement of consistent information across processors, are preferred in practice. Analysis of such methods is more difficult, but results have been obtained that accord with practical experience of such methods. Indeed, it can be verified that in certain regimes, linear speedup can be expected across a modest number of processors.

4.1 Synchronous parallelism

We mention several synchronous parallel variants of CD that appear in the recent literature. We note that in the some of these papers, the computational results were obtained by implementing the methods in an asynchronous fashion, disregarding the synchronization step required by the analysis.

Bradley at al. [9] consider a bound-constrained problem that is a reformulation of the problem (2) with specific choices of f and with $\Omega(x) = \|x\|_1$. Their algorithm performs short-step updates of individual components of x in parallel on P processors, with synchronization after each round of parallel updating. This scheme essentially updates a randomly-chosen block of P variables at each cycle. By modifying the analysis of [50], they show that the $1/k$ sublinear convergence rate bound is not affected provided that P is no larger than n/L, where L is the Lipschitz constant from (23).

Jaggi et al. [20] perform a synchronized CD method on the dual ERM model (7) for the case of $g(w) = g^*(w) = (1/2)\|w\|^2$, partitioning components of the dual variable x between cores and sharing a copy of the vector Ax across cores, updating this vector at each synchronization point. The approach can be thought of as a nonlinear block Gauss-Jacobi method (by contrast with the coordinate Gauss–Seidel approaches discussed in Sect. 3).

Richtarik and Takac [46] describe a method for the separably regularized formulation (2), (3) in which a subset of indices $S_k \subset \{1, 2, \ldots, n\}$ is updated according to the formula in Algorithm 2. The work of updating the components in S_k is divided between processors; essentially, a synchronization step takes place at each iteration. This scheme is enhanced with an acceleration step in [15]; the extra computations associated with the acceleration step too are parallelized, using ideas from [23]. In the scheme of Marecek, Richtarik, and Takac [32], the variable vector x is partitioned into subvectors, and each processor is assigned the responsibility for updating one of these subvectors. On each processor, the updating scheme described in [46] is applied, providing a second level of parallelism. Synchronization takes place at each outer iteration. Details of the information-sharing between processors required for accurate computation of gradients in different applications are described in [32, Section 6].

4.2 Asynchronous parallelism

In asynchronous variants of CD, the variable vector x is assumed to be accessible to each processor, available for reading and updating. (For example, x could be stored in the shared-memory space of a multicore computer, where each core is viewed as a processor.) Each processor runs its own CD process, shown here as Algorithm 6, without any attempt to coordinate or synchronize with other processors. Each iteration on each processor chooses an index i, loads the components of x that are needed to compute the gradient component $[\nabla f(x)]_i$, then updates the ith component x_i. Note that this evaluation may need only a small subset of the components of x; this is the case when the Hessian $\nabla^2 f$ is structurally sparse, for example. On some multicore architectures (for example, the Intel Xeon), the update of x_i can be performed as a

unitary operation; no software or hardware locking is required to block access of other cores to the location x_i.

Algorithm 6 Coordinate Descent for (1) (running on each Processor)

repeat
  Choose index $i \in \{1, 2, \ldots, n\}$;
  Evaluate $[\nabla f(x)]_i$, reading components of x from shared memory as necessary;
  Update $x_i \leftarrow x_i - \alpha[\nabla f(x)]_i$ for some $\alpha > 0$;
until termination;

We can take a global view of the entire parallel process, consisting of multiple processors each executing Algorithm 6, by defining a global counter k that is incremented whenever *any* processor updates an element of x: see Algorithm 7. Note that the *only* difference with the basic framework of Algorithm 1 is in the argument of the gradient component: In Algorithm 1 this is the latest iterate x^k whereas in Algorithm 7 it is a vector $\hat{x}^k$ that is generally made up of components of vectors from previous iterations x^j, $j = 0, 1, \ldots, k$. The reason for this discrepancy is that between the time at which a processor *reads* the vector x from shared storage in order to calculate $[\nabla f(x)]_i$, and the time at which it *updates* component i, *other processors* have generally made changes to x. In consequence, each update step is using slightly stale information about x. To prove convergence results, we need to make assumptions on how much "staleness" can be tolerated, and to modify the convergence analysis quite substantially. Indeed, proofs of convergence even for the most basic asynchronous algorithms are quite technical.

Algorithm 7 Asynchronous CD for (1)

Set $k \leftarrow 0$ and choose $x^0 \in \mathbb{R}^n$;
repeat
  Choose index $i_k \in \{1, 2, \ldots, n\}$;
  $x^{k+1} \leftarrow x^k - \alpha_k[\nabla f(\hat{x}^k)]_{i_k} e_{i_k}$ for some $\alpha_k > 0$;
  $k \leftarrow k + 1$;
until termination test satisfied;

Asynchronous CD algorithms are distinguished from each other mostly by the assumptions they make on the the choice of update components i_k and on the "ages" of the components of $\hat{x}^k$, that is, the iterations at which each component of this vector was last updated. In the terminology of Bertsekas and Tsitsiklis [5], the algorithm is *totally asynchronous* if

(a) each index $i \in \{1, 2, \ldots, n\}$ of x is updated at infinitely many iterations; and
(b) if ν_j^k denotes the iteration at which component j of the vector $\hat{x}^k$ was last updated, then $\nu_j^k \to \infty$ as $k \to \infty$ for all $j = 1, 2, \ldots, n$.

In other words, each component of x is updated infinitely often, and all components used in successive evaluation vectors $\hat{x}^k$ are also updated infinitely often.

The following convergence result for totally asynchronous variants of Algorithm 7 is due to Bertsekas and Tsitsiklis; see in particular [5, Sections 6.1, 6.2, and 6.3.3].

Theorem 5 *Suppose that the problem* (1) *has a unique solution* x^* *and that* f *is convex and continuously differentiable. Suppose that Algorithm 7 is implemented in a totally asynchronous fashion. Suppose that the mapping* T *defined by* $T(x) := x - \alpha \nabla f(x)$ *for some* $\alpha > 0$ *(for which* x^* *is the unique fixed point) is strictly contractive in the* ℓ_∞ *norm, that is,*

$$\|T(x) - x^*\|_\infty \leq \eta \|x - x^*\|_\infty, \quad \text{for some } \eta \in (0, 1). \tag{48}$$

Then if we set $\alpha_k \equiv \alpha$ *in Algorithm 7, the sequence* $\{x^k\}$ *converges to* x^*.

We cannot expect to obtain a convergence rate in this setting (such as sublinear with rate $1/k$), given that the assumptions on the ages of the components in $\hat{x}^k$ are so weak. Although this result can be generalized impressively and its proof is not too complex, we should note that the ℓ_∞ contraction assumption (48) is quite strong. It is violated even by some strictly convex objectives f. For example, when $f(x) = (1/2)x^T Qx$ with

$$Q = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix},$$

we have f strictly convex with minimizer $x^* = 0$. However the mapping $T(x) = (I - \alpha Q)x$ is not contractive for any $\alpha > 0$; we have for example that $\|T(x)\|_\infty \geq \|x\|_\infty$ when $x = (1, -1)^T$.

We turn now to *partly asynchronous* variants of Algorithm 7, in which we make stronger assumptions on the ages of the components of $\hat{x}^k$. Liu and Wright [27] consider a version of Algorithm 7 that is the parallel analog of Algorithm 3, in that each update component i_k is chosen independently and randomly with equal probability from $\{1, 2, \ldots, n\}$. They assume that no component of $\hat{x}^k$ is older than a nonnegative integer τ—the "maximum delay"—for any k. Specifically, they express the difference between x^k and $\hat{x}^k$ in terms of "missed updates" to x, as follows:

$$x^k = \hat{x}^k + \sum_{l \in K(j)} \left(x^{l+1} - x^l \right), \tag{49}$$

where $K(j)$ is a set of iteration numbers drawn from the set $\{j - q : q = 1, 2, \ldots, \tau\}$. The value of τ is related to the number of processors P involved in the computation. If all processors are performing their updates at approximately the same rates, we could expect τ to be a modest multiple of P—perhaps $\tau = 2P$ or $\tau = 3P$, to allow a safety margin for occasional delays. Hence the value of τ is an indicator of potential parallelism in the algorithm.

In [27], the steplengths in Algorithm 7 are fixed as follows:

$$\alpha_k \equiv \frac{\gamma}{L_{\max}}, \tag{50}$$

where γ is chosen to ensure that Algorithm 7 progresses steadily toward a solution, but not too rapidly. Too-rapid convergence would cause the information in $\hat{x}^k$ to become too stale too quickly, so the gradient component $[\nabla f(\hat{x}^k)]_{i_k}$ would lose its relevance as a suitable update for the variable component x_{i_k} at iteration k. Steady convergence is enforced by choosing some $\rho > 1$ and requiring that

$$E\|x^{k-1} - \bar{x}^k\|^2 \leq \rho E\|x^k - \bar{x}^{k+1}\|^2, \tag{51}$$

where $\bar{x}^k$ is the vector that would hypothetically be obtained if we were to apply the the update to *all* components, that is,

$$\bar{x}^{k+1} := x^k - \frac{\gamma}{L_{\max}} \nabla f(\hat{x}^k),$$

and the expectations $E(\cdot)$ are taken over all random variables $i_0, i_2, \ldots$. Condition (51) ensures that the "expected squared update norms" decrease by at most a factor of $1/\rho$ at each iteration.

The main results in [27] apply to composite functions (2), (3), but for simplicity here we state the result in terms of the problem (1), where f is convex and continuously differentiable, with nonempty solution set $\mathcal{S}$ and optimal objective value f^*. We use $P_{\mathcal{S}}$ to denote projection onto $\mathcal{S}$, and recall the definition (25) of the ratio Λ between different varieties of Lipschitz constants. The results also make use of an *optimal strong convexity* condition, which is that the following inequality holds for some $\sigma > 0$:

$$f(x) - f^* \geq \frac{\sigma}{2}\|x - P_{\mathcal{S}}(x)\|^2, \quad \text{for all } x. \tag{52}$$

The following result is a modification of [27, Corollary 2].

Theorem 6 *Suppose that* Assumption 1 *holds, and that*

$$4e\Lambda(\tau + 1)^2 \leq \sqrt{n}. \tag{53}$$

Then by setting $\gamma = 1/2$ *in* (50) *(that is, choosing steplengths* $\alpha_k \equiv 1/(2L_{\max})$*), we have that*

$$E\left(f(x^k)\right) - f^* \leq \frac{n(L_{\max}\|x^0 - P_{\mathcal{S}}(x^0)\|^2 + f(x^0) - f^*)}{n + k}. \tag{54}$$

Assuming in addition that (52) *is satisfied for some* $\sigma > 0$*, we obtain the following linear rate:*

$$\begin{aligned} &E\left(f(x^k)\right) - f^* \\ &\leq \left(1 - \frac{\sigma}{n(\sigma + 2L_{\max})}\right)^k \left(L_{\max}\|x^0 - P_{\mathcal{S}}(x^0)\|^2 + f(x^0) - f^*\right). \end{aligned} \tag{55}$$

A comparison with Theorem 1, which shows convergence rates for serial randomized CD (Algorithm 3) shows a striking similarity in convergence bounds. The factor-of-2 difference in steplength between the serial and parallel variants accounts for most of the difference between the linear rates (28) and (55), while there is an extra term n in the denominator of the sublinear rate (54). We conclude that we do not pay q high overhead (in terms of total workload) for parallel implementation, and hence that near-linear speedup can be expected. (Indeed, computational results in [27] and [28] observe near-linear speedup for multicore asynchronous implementations.)

These encouraging conclusions depend critically on the condition (53), which is an upper bound on the allowable delay τ in terms of n and the ratio Λ from (25). For functions f with weak coupling between the components of x (for example, when off-diagonals in the Hessian $\nabla^2 f(x)$ are small relative to the diagonals), we have Λ not much greater than 1, so the maximum delay can be of the order of $n^{1/4}$ before there is any attenuation of linear speedup. When stronger coupling exists, the restriction on τ may be quite tight, possibly not much greater than 1. A more general convergence result [27, Theorem 1] shows that in this case, we can choose smaller values of γ in (50), allowing graceful degradation of the convergence bounds while still obtaining fairly efficient parallel implementations.

We note that an earlier analysis in [28] made a stronger assumption on $\hat{x}^k$—that it is equal to some earlier iterate x^j of Algorithm 7, where $k \geq j \geq k - \tau$, that is, the earlier iterate is no more than τ cycles old. (A similar assumption was used to analyze convergence of as asynchronous SG algorithm in [38].) This stronger assumption yields stronger convergence results, in that the bound on τ in (53) can be loosened. However, the assumption may not always hold, since some parts of x in memory may be altered by some cores as they are being read by another core, a phenomenon referred to in [27] as "inconsistent reading."

5 Conclusion

We have surveyed the state of the art in convergence of CD methods, with a focus on the most elementary settings and the most fundamental algorithms. The recent literature contains many extensions, enhancements, and elaborations; we refer interested readers to the bibliography of this paper, and note that new works are appearing at a rapid pace.

Coordinate descent method have become an important tool in the optimization toolbox that is used to solve problems that arise in machine learning and data analysis, particularly in "big data" settings. We expect to see further developments and extensions, further customization of the approach to specific problem structures, further adaptation to various computer platforms, and novel combinations with other optimization tools to produce effective "solutions" for key application areas.

Acknowledgments I thank Ji Liu for the pleasure of collaborating with him on this topic over the past two years. I am grateful to the editors and referees of the paper, whose expert and constructive comments led to numerous improvements.

References

1. Attouch, H., Bolte, J., Redont, P., Soubeyran, A.: Proximal alternating minimization and projection methods for nonconvex problems: an approach based on the Kurdyka-Lojasiewicz inequality. Math. Oper. Res. **35**(2), 438–457 (2010)
2. Beck, A., Teboulle, M.: A fast iterative shrinkage-threshold algorithm for linear inverse problems. SIAM J. Imaging Sci. **2**(1), 183–202 (2009)
3. Beck, A., Tetruashvili, L.: On the convergence of block coordinate descent methods. SIAM J. Optim. **23**(4), 2037–2060 (2013)
4. Bertsekas, D.P.: Nonlinear Programming, 2nd edn. Athena Scientific, Belmont (1999)
5. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and Distributed Computation: Numerical Methods. Prentice-Hall Inc, Englewood Cliffs (1989)
6. Bolte, J., Sabach, S., Teboulle, M.: Proximal alternating linearized minimization for nonconvex and nonsmooth problems. Math. Program. Ser. A **146**, 1–36 (2014)
7. Bouman, C.A., Sauer, K.: A unified approach to statistical tomography using coordinate descent optimization. IEEE Trans. Image Process. **5**(3), 480–492 (1996)
8. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction methods of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2011)
9. Bradley, J.K., Kyrola, A., Bickson, D., Guestrin, C.: Parallel coordunate descent for ℓ_1-regularized loss minimization. In: Proceedings of the 28 International Conference on Machine Learning (ICML 2011) (2011)
10. Breheny, P., Huang, J.: Coordinate descent algroithms for nonconvex penalized regression, with applications to biological feature selection. Ann. Appl. Stat. **5**(1), 232–252 (2011)
11. Canutescu, A.A., Dunbrack, R.L.: Cyclic coordinate descent: a robotics algorithm for protein loop closure. Protein Sci. **12**(5), 963–972 (2003)
12. Chang, K., Hsieh, C., Lin, C.: Coordinate descent method for large-scale l2-loss linear support vector machines. J. Mach. Learn. Res. **9**, 1369–1398 (2008)
13. Eckstein, J., Bertsekas, D.P.: On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. Math. Program. **55**, 293–318 (1992)
14. Eckstein, J., Yao, W.: Understanding the convergence of the alternating direction method of multipliers: theoretical and computational perspectives. Technical Report, RUTCOR, Rutgers University (2014)
15. Fercoq, O., Qu, Z., Richtarik, P., Takac, M.: Fast distributed coordinate descent for non-strongly convex losses (2014). arxiv:1405.5300
16. Fercoq, O., Richtarik, P.: Accelerated, parallel, and proximal coordinate descent. Technical Report, School of Mathematics, University of Edinburgh (2013). arXiv:1312.5799
17. Florian, M., Chen, Y.: A coordinate descent method for the bilevel O-D matrix adjustment problem. Int. Trans. Oper. Res. **2**(2), 165–179 (1995)
18. Friedman, J., Hastie, T., Tibshirani, R.: Sparse inverse covariance estimation with the graphical lasso. Biostatistics **9**(3), 432–441 (2008)
19. Friedman, J.H., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. J. Stat. Softw. **33**(1), 1–22 (2010)
20. Jaggi, M., Smith, V., Takác, M., Terhorst, J., Krishnan, S., Hoffman, T., Jordan, M.I.: Communication-efficient distributed dual coordinate ascent. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 27, pp. 3068–3076. Curran Associates (2014)
21. Jain, P., Netrapalli, P., Sanghavi, S.: Low-rank matrix completion using alternating minimization. Technical Report (2012). arXiv:1212.0467
22. Kaczmarz, S.: Angenäherte auflösung von systemen linearer gleichungen. Bulletin International de l'Academie Polonaise des Sciences et des Lettres **35**, 355–357 (1937)
23. Lee, Y.T., Sidford, A.: Efficient accelerated coordinate descent methods and faster algorihtms for solving linear systems. In: 54th Annual Symposium on Foundations of Computer Science, pp. 147–156 (2013)
24. Leventhal, D., Lewis, A.S.: Randomized methods for linear constraints: convergence rates and conditioning. Math. Oper. Res. **35**(3), 641–654 (2010)
25. Lin, Q., Lu, Z., Xiao, L.: An accelerated proximal coordinate gradient method and its application to empirical risk minimization. Technical Report, Microsoft Research (2014). arXiv:1407.1296

26. Liu, H., Palatucci, M., Zhang, J.: Lockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In: Proceedings of the 26th Annual International Conference on Machine Learning. ICML '09, pp. 649–656. ACM, New York, NY, USA (2009)
27. Liu, J., Wright, S.J.: Asynchronous stochastic coordinate descent: parallelism and convergence properties. Technical Report, University of Wisconsin, Madison. (2014). (To appear in SIAM Journal on Optimization). arXiv:1403.3862
28. Liu, J., Wright, S.J., Ré, C., Bittorf, V., Sridhar, S.: An asynchronous parallel stochastic coordinate descent algorithm. Technical Report, Computer Sciences Department, University of Wisconsin-Madison (2013). (To appear in Journal of Machine Learning Research). arXiv:1311.1873
29. Liu, J., Wright, S.J., Sridhar, S.: An accelerated randomized Kaczmarz algorithm. Technical Report, Computer Sciences Department, University of Wisconsin-Madison (2013). (To appear in Mathematics of Computation). arXiv 1310.2887
30. Luo, Z.Q., Tseng, P.: On the convergence of the coordinate descent method for convex differentiable minimization. J. Optim. Theory Appl. **72**(1), 7–35 (1992)
31. Luo, Z.Q., Tseng, P.: Error bounds and convergence analysis of feasible descent methods: a general approach. Ann. Oper. Res. **46**, 157–178 (1993)
32. Marecek, J., Richtarik, P., Takac, M.: Distributed block coordinate descent for minimizing partially separable functions. Technical Report arXiv:1406.0238 (2014)
33. Mazumder, R., Friedman, J.H., Hastie, T.: SparseNet: coordinate descent with nonconvex penalties. J. Am. Stat. Assoc. **106**, 1125–1138 (2011)
34. Necoara, I., Clipici, D.: Distributed random coordinate descent method for composite minimization. Technical Report 1–41, University Politehnica Bucharest (2013)
35. Nesterov, Y.: A method for unconstrained convex problem with the rate of convergence $O(1/k^2)$. Doklady AN SSSR **269**, 543–547 (1983)
36. Nesterov, Y.: Introductory Lectures on Convex Optimization: A Basic Course. Springer, New York (2004)
37. Nesterov, Y.: Efficiency of coordinate descent methods on huge-scale optimization problems. SIAM J. Optim. **22**, 341–362 (2012)
38. Niu, F., Recht, B., Ré, C., Wright, S.J.: Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In: Shawe-Taylor, J., Zemel, R.S., Bartlett, P.L., Pereira, F., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 24, pp. 693–701. Curran Associates (2011)
39. Ortega, J.M., Rheinboldt, W.C.: Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, New York (1970)
40. Patrascu, A., Necoara, I.: Efficient random coordinate descent algorithms for large-scale structured nonconvex optimization. J. Glob. Optim. (2013). doi:10.1007/s10898-014-0151-9
41. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In: Schölkopf, B., Burges, C.J.C., Smola, A.J. (eds.) Advances in Kernel Methods—Support Vector Learning, pp. 185–208. MIT Press, Cambridge (1999)
42. Polyak, B.T.: Introduction to Optimization. Optimization Software, New York (1987)
43. Powell, M.J.D.: On search directions for minimization algorithms. Math. Program. **4**, 193–201 (1973)
44. Razaviyayn, M., Hong, M., Luo, Z.Q.: A unified convergence analysis of block successive minimization methods for nonsmooth optimization. SIAM J. Optim. **23**(2), 1126–1153 (2013)
45. Recht, B., Fazel, M., Parrilo, P.: Guaranteed minimum-rank solutions to linear matrix equations via nuclear norm minimization. SIAM Rev. **52**(3), 471–501 (2010)
46. Richtarik, P., Takac, M.: Parallel coordinate descent methods for big data optimization. Technical Report, School of Mathematics, University of Edinburgh (2013). arXiv:1212.0873
47. Richtarik, P., Takac, M.: Iteration complexity of a randomized block-coordinate descent methods for minimizing a composite function. Math. Program. Ser. A **144**(1), 1–38 (2014)
48. Rockafellar, R.T.: Convex Analysis. Princeton University Press, Princeton (1970)
49. Sardy, S., Bruce, A., Tseng, P.: Block coordinate relaxation methods for nonparametric wavelet denoising. J. Comput. Graph. Stat. **9**, 361–379 (2000)
50. Shalev-Shwartz, S., Tewari, A.: Stochastic methods for ℓ_1-regularized loss minimization. J. Mach. Learn. Res. **12**, 1865–1892 (2011)
51. Shalev-Shwartz, S., Zhang, T.: Stochastic dual coordinate ascent mehods for regularized loss minimization. J. Mach. Learn. Res. **14**, 437–469 (2013)
52. Strohmer, T., Vershynin, R.: A randomized Kaczmarz algorithm with exponential convergence. J. Fourier Anal. Appl. **15**, 262–278 (2009)

53. Tibshirani, R.: Regression shrinkage and selection via the LASSO. J. R. Stat. Soc. B **58**, 267–288 (1996)
54. Tseng, P.: Convergence of a block coordinate descent method for nondifferentiable minimization. J. Optim. Theory Appl. **109**(3), 475–494 (2001)
55. Tseng, P., Yun, S.: A coordinate gradient descent method for nonsmooth separable minimization. Math. Program. Ser. B **117**, 387–423 (2009)
56. Ye, J.C., Webb, K.J., Bouman, C.A., Millane, R.P.: Optical diffusion tomography by iterative-coordinate-descent optimization in a bayesian framework. J. Opt. Soc. Am. A **16**(10), 2400–2412 (1999)