Semi-stochastic coordinate descent

Citation for published version:

Digital Object Identifier (DOI):
10.1080/10556788.2017.1298596

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Publisher's PDF, also known as Version of record

Published In:
Optimization Methods and Software

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Semi-stochastic coordinate descent

Jakub Konečný, Zheng Qu & Peter Richtárik

To cite this article: Jakub Konečný, Zheng Qu & Peter Richtárik (2017): Semi-stochastic coordinate descent, Optimization Methods and Software, DOI: 10.1080/10556788.2017.1298596

To link to this article: http://dx.doi.org/10.1080/10556788.2017.1298596

© 2017 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group

Published online: 13 Mar 2017.

Submit your article to this journal

Article views: 55

View related articles

View Crossmark data
Optimization Methods & Software, 2017
http://dx.doi.org/10.1080/10556788.2017.1298596

Semi-stochastic coordinate descent

Jakub Konečný a*, Zheng Qu b† and Peter Richtárik a

aSchool of Mathematics, The University of Edinburgh, Edinburgh, UK; bDepartment of Mathematics, The University of Hong Kong, Hong Kong

(Received 23 December 2015; accepted 18 February 2017)

We propose a novel stochastic gradient method—semi-stochastic coordinate descent—for the problem of minimizing a strongly convex function represented as the average of a large number of smooth convex functions:

\[ f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x). \]

Our method first performs a deterministic step (computation of the gradient of \( f \) at the starting point), followed by a large number of stochastic steps. The process is repeated a few times, with the last stochastic iterate becoming the new starting point where the deterministic step is taken. The novelty of our method is in how the stochastic steps are performed. In each such step, we pick a random function \( f_i \) and a random coordinate \( j \)—both using non-uniform distributions—and update a single coordinate of the decision vector only, based on the computation of the \( j \)th partial derivative of \( f_i \) at two different points. Each random step of the method constitutes an unbiased estimate of the gradient of \( f \) and moreover, the squared norm of the steps goes to zero in expectation, meaning that the stochastic estimate of the gradient progressively improves. The computational complexity of the method is the sum of two terms: \( O(n \log(1/\epsilon)) \) evaluations of gradients \( \nabla f_i \) and \( O(\hat{\kappa} \log(1/\epsilon)) \) evaluations of partial derivatives \( \nabla_j f_i \), where \( \hat{\kappa} \) is a novel condition number.

Keywords: Stochastic gradient; coordinate descent; empirical risk minimization

AMS Subject Classification: 90C06; 90C15

1. Introduction

In this paper we study the problem of unconstrained minimization of a strongly convex function represented as the average of a large number of smooth convex functions:

\[ \min_{x \in \mathbb{R}^d} f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x). \]  

Many computational problems in various disciplines are of this form. In machine learning, \( f_i(x) \) represents the loss/risk of classifier \( x \in \mathbb{R}^d \) on data sample \( i \), \( f \) represents the empirical risk (= average loss), and the goal is to find a predictor minimizing \( f \). An L2-regularizer of the form \( \mu \|x\|^2 \), for \( \mu > 0 \), could be added to the loss, making it strongly convex and hence easier to minimize.

*Corresponding author. Email: j.konecny@sms.ed.ac.uk
†The paper was written while Zheng Qu held position at University of Edinburgh.

© 2017 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.
Assumptions. We assume that the functions $f_i : \mathbb{R}^d \rightarrow \mathbb{R}$ are differentiable and convex function, with Lipschitz continuous partial derivatives. Formally, we assume that for each $i \in [n] := \{1, 2, \ldots, n\}$ and $j \in [d] := \{1, 2, \ldots, d\}$ there exists $L_{ij} \geq 0$ such that for all $x \in \mathbb{R}^d$ and $h \in \mathbb{R}$,

$$f_i(x + he_j) \leq f_i(x) + \langle \nabla f_i(x), he_j \rangle + \frac{L_{ij}}{2} h^2,$$

(2)

where $e_j$ is the $j$th standard basis vector in $\mathbb{R}^d$, $\nabla f_i(x) \in \mathbb{R}^d$ is the gradient of $f_i$ at point $x$ and $\langle \cdot, \cdot \rangle$ is the standard inner product. This assumption was recently used in the analysis of the accelerated coordinate descent method APPROX [2]. We further assume that $f$ is $\mu$-strongly convex. That is, we assume that there exists $\mu > 0$ such that for all $x, y \in \mathbb{R}^d$,

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2} \|y - x\|^2.$$

(3)

Context. Batch methods such as gradient descent (GD) enjoy a fast (linear) convergence rate: to achieve $\epsilon$-accuracy, GD needs $O(\kappa \log(1/\epsilon))$ iterations, where $\kappa$ is a condition number. The drawback of GD is that in each iteration one needs to compute the gradient of $f$, which requires a pass through the entire data set. This is prohibitive to do many times if $n$ is very large.

Stochastic gradient descent (SGD) in each iteration computes the gradient of a single randomly chosen function $f_i$ only—this constitutes an unbiased (but noisy) estimate of the gradient of $f$—and makes a step in that direction [8,16,20,23]. The rate of convergence of SGD is slower, $O(1/\epsilon)$, but the cost of each iteration is independent of $n$. Variants with non-uniform selection probabilities were considered in [24], a mini-batch variant (for Support Vector Machines with hinge loss) was analysed in [21].

Recently, there has been progress in designing algorithms that achieve the fast $O(\log(1/\epsilon))$ rate without the need to scan the entire data set in each iteration. The first class of methods to have achieved this are stochastic/randomized coordinate descent methods.

When applied to (1), coordinate descent methods (CD) [10,15] can, like SGD, be seen as an attempt to keep the benefits of GD (fast linear convergence) while reducing the complexity of each iteration. A CD method only computes a single partial derivative $\nabla f_i(x)$ at each iteration and updates a single coordinate of vector $x$ only. When chosen uniformly at random, partial derivative is also an unbiased estimate of the gradient. However, unlike the SGD estimate, its variance decrease to zero as one approaches the optimum. While CD methods are able to obtain linear convergence, they typically need $O((d/\mu) \log(1/\epsilon))$ iterations when applied to (1) directly. CD method typically significantly outperform GD, especially on sparse problems with a very large number of variables/coordinates [10,15].

An alternative to applying CD to (1) is to apply it to the dual problem. This is possible under certain additional structural assumptions on the functions $f_i$. This is the strategy employed by stochastic dual coordinate ascent (SDCA) [11,19], whose rate is

$$O((n + \kappa) \log(1/\epsilon)).$$

The condition number $\kappa$ here is the same as the condition number appearing in the rate of GD. Despite this, this is a vast improvement on the computational complexity achieved by GD which has an iteration cost $n$ times larger than SDCA. Also, the linear convergence rate is superior to the sublinear rate $O(1/\epsilon)$ achieved by SGD, and the method indeed typically performs much better in practice. Accelerated [18] and mini-batch [21] variants of SDCA have also been proposed. We refer the reader to QUARTZ [11] for a general analysis involving the update of a random subset of dual coordinates, following an arbitrary distribution.

Recently, there has been progress in designing primal methods which match the fast rate of SDCA. Stochastic average gradient (SAG) [17], and more recently SAGA [1], move in a direction composed of old stochastic gradients. The semi-stochastic gradient descent (S2GD) [6,7]
Table 1. Runtime complexity of various algorithms.

<table>
<thead>
<tr>
<th>Method</th>
<th>Runtime</th>
<th>Paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>$O(n \kappa C_{\text{grad}} \log (1/\epsilon))$</td>
<td>e.g. [9]</td>
</tr>
<tr>
<td>SGD</td>
<td>$O(C_{\text{grad}}/\epsilon)$</td>
<td>[20,23]</td>
</tr>
<tr>
<td>CD</td>
<td>$O(n \kappa C_{pd} \log (1/\epsilon))$</td>
<td>[10,15]</td>
</tr>
<tr>
<td>SDCA</td>
<td>$O((n + \kappa) C_{\text{grad}} \log (1/\epsilon))$</td>
<td>[11,19,25]</td>
</tr>
<tr>
<td>SVRG/S2GD</td>
<td>$O((n \kappa C_{\text{grad}} + \kappa C_{\text{grad}}) \log (1/\epsilon))$</td>
<td>[5,6,22]</td>
</tr>
<tr>
<td>S2CD</td>
<td>$O((n \kappa C_{\text{grad}} + \kappa C_{pd}) \log (1/\epsilon))$</td>
<td>This paper</td>
</tr>
</tbody>
</table>

Notes: We use $C_{\text{grad}}$ to denote the evaluation cost of the gradient of one single function $\nabla f_i$, and use $C_{pd}$ to denote the evaluation cost of a partial derivative $\nabla_{j} f_i$.

and stochastic variance reduced gradient (SVRG) [5,22] methods employ a different strategy: one first computes the gradient of $f$, followed by $O(\kappa)$ steps where only stochastic gradients are computed. These are used to estimate the change of the gradient, and it is this direction which combines the old gradient and the new stochastic gradient information which is used in the update.

Main result. In this work we develop a new method—semi-stochastic coordinate descent (S2CD)—for solving (1), enjoying a fast rate similar to methods such as SDCA, SAG, S2GD, SVRG, SAGA, mS2GD and QUARTZ. S2CD can be seen as a hybrid between S2GD and CD. In particular, the complexity of our method is the sum of two terms:

$$O(n \log (1/\epsilon))$$

evaluations $\nabla f_i$ (that is, $\log (1/\epsilon)$ evaluations of the gradient of $f$) and

$$O(\tilde{\kappa} \log (1/\epsilon))$$

evaluations of $\langle e_j, \nabla f_i \rangle$ for randomly chosen functions $f_i$ and randomly chosen coordinates $j$, where $\tilde{\kappa}$ is a new condition number which is defined in (13) and larger than $\kappa$. We summarize in Table 1 the runtime complexity of the various algorithms. Note that $\tilde{\kappa}$ enters the complexity only in the term involving the evaluation cost of a partial derivative $\nabla_{j} f_i$, which can be substantially smaller than the evaluation cost of $\nabla f_i$. Hence, our complexity result can be both better or worse than previous results, depending on whether the increase of the condition number can or cannot be compensated by the lower cost of the stochastic steps based on the evaluation of partial derivatives.

Outline. The paper is organized as follows. In Section 2 we describe the S2CD algorithm and in Section 3 we state a key lemma and our main complexity result. The proof of the lemma is provided in Section 4 and the proof of the main result in Section 5.

2. S2CD algorithm

In this section we describe the semi-stochastic coordinate descent method (Algorithm 1).

The discussion on the choice of $m$ and $h$ in Algorithm 1 is deferred to Section 3. As we will see, the parameters $m$ and $h$ depends on the target accuracy and the number of iterations. We next provide a more detailed description of the algorithm.

The method has an outer loop (an ‘epoch’), indexed by counter $k$, and an inner loop, indexed by $t$. At the beginning of epoch $k$, we compute and store the gradient of $f$ at $x_k$. Subsequently, S2CD enters the inner loop in which a sequence of vectors $y_{k,t}$ for $t = 0, 1, \ldots, t_k$ is computed in
Algorithm 1 Semi-stochastic coordinate descent (S2CD)

**parameters**: $m$ (max # of stochastic steps per epoch); $h > 0$ (stepsize parameter); $x_0 \in \mathbb{R}^d$ (starting point)

**for** $k = 0, 1, 2, \ldots$ **do**

- Compute and store $\nabla f(x_k) = \frac{1}{n} \sum_i \nabla f_i(x_k)$
- Initialize the inner loop: $y_{k,0} \leftarrow x_k$
- Let $t_k = T \in \{1, 2, \ldots, m\}$ with probability $(1 - \mu h)^{m-T}/\beta$
  **for** $t = 0$ to $t_k - 1$ **do**
    - Pick coordinate $j \in \{1, 2, \ldots, d\}$ with probability $p_j$
    - Pick function index $i$ from the set $\{i: L_{ij} > 0\}$ with probability $q_{ij}$
    - Update $j$th coordinate: $y_{k,t+1} \leftarrow y_{k,t} - h p_j^{-1} \left( \nabla f(x_k) + \frac{1}{nq_{ij}} (\nabla f_i(y_{k,t}) - \nabla f_i(x_k)) \right) e_j$
  **end for**
- Reset the starting point: $x_{k+1} \leftarrow y_{k,t_k}$
  **end for**

A stochastic way, starting from $y_{k,0} = x_k$. The number $t_k$ of stochastic steps in the inner loop is random, following a geometric law:

$$P(t_k = T) = \frac{(1 - \mu h)^{m-T}}{\beta}, \quad T \in \{1, \ldots, m\},$$

where

$$\beta := \sum_{i=1}^m (1 - \mu h)^{m-i}. \quad (4)$$

In each step of the inner loop, we seek to compute $y_{k,t+1}$, given $y_{k,t}$. In order to do so, we sample coordinate $j$ with probability $p_j$ and subsequently sample $i$ with probability $q_{ij}$, where the probabilities are given by

$$\omega_i := |\{j: L_{ij} \neq 0\}|, \quad v_j := \sum_{i=1}^n \omega_i L_{ij}, \quad p_j := v_j / \sum_{j=1}^d v_j, \quad q_{ij} := \omega_i L_{ij}/v_j, \quad p_{ij} := p_j q_{ij}. \quad (5)$$

Note that $L_{ij} = 0$ means that function $f_i$ does not depend on the $j$th coordinate of $x$. Hence, $\omega_i$ the number of coordinates function $f_i$ depends on—a measure of sparsity of the data.\(^3\) It can be shown that $f$ has a 1-Lipschitz gradient with respect to the weighted Euclidean norm with weights $\{v_j\}$ [2, Theorem 1]. Hence, we sample coordinate $j$ proportionally to this weight $v_j$. Note that $p_{ij}$ is the joint probability of choosing the pair $(i, j)$.

Having sampled coordinate $j$ and function index $i$, we compute two partial derivatives: $\nabla f_i(x_k)$ and $\nabla f_i(y_{k,t})$ (we compressed the notation here by writing $\nabla f_i(x)$ instead of $\langle \nabla f_i(x), e_j \rangle$), and combine these with the pre-computed value $\nabla f(x_k)$ to form an update of the form

$$y_{k,t+1} \leftarrow y_{k,t} - h p_j^{-1} G_{ki}^{ij} e_j = y_{k,t} - h G_{ki}^{ij}, \quad (6)$$

where

$$G_{ki}^{ij} := \nabla f_i(x_k) + \frac{1}{nq_{ij}} (\nabla f_i(y_{k,t}) - \nabla f_i(x_k)). \quad (8)$$

Note that only a single coordinate of $y_{k,t}$ is updated at each iteration.
In the entire text (with the exception of the statement of Theorem 3.2 and a part of Section 5.3, where $E$ denotes the total expectation) we will assume that all expectations are conditional on the entire history of the random variables generated up to the point when $y_{k,i}$ was computed. With this convention, it is possible to think that there are only two random variables: $j$ and $i$. By $E$ we then mean the expectation with respect to both of these random variables, and by $E_i$ we mean expectation with respect to $i$ (that is, conditional on $j$). With this convention, we can write

$$E_i[G_{kl}^{ij}] = \sum_{i=1}^{n} q_i G_{kl}^{ij} \tag{8}$$

$$= \nabla_j f(x_k) + \frac{1}{n} \sum_{i=1}^{n} (\nabla_j f_i(y_{k,i}) - \nabla_j f(x_k)) \tag{1}$$

which means that conditioned on $j$, $G_{kl}^{ij}$ is an unbiased estimate of the $j$th partial derivative of $f$ at $y_{k,i}$. An equally easy calculation reveals that the random vector $g_{kl}^{ij}$ is an unbiased estimate of the gradient of $f$ at $y_{k,i}$:

$$E[g_{kl}^{ij}] \tag{7}$$

$$= E[p_j^{-1} G_{kl}^{ij} e_j] = E[E_i[p_j^{-1} G_{kl}^{ij} e_j]]$$

$$= E[p_j^{-1} e_j E_i[G_{kl}^{ij}]] \tag{9}$$

$$= E[p_j^{-1} e_j \nabla_j f(y_{k,i})] = \nabla f(y_{k,i}).$$

Hence, the update step performed by S2CD is a stochastic gradient step of fixed stepsize $h$.

Before we describe our main complexity result in the next section, let us briefly comment on a few special cases of S2CD:

- If $n = 1$ (this can be always achieved simply by grouping all functions in the average into a single function), S2CD reduces to a stochastic CD algorithm with importance sampling, as studied in [10,11,15], but written with many redundant computations. Indeed, the method in this case does not require the $x_k$ iterates, nor does it need to compute the gradient of $f$, and instead takes on the form:

$$y_{0,t+1} \leftarrow y_{0,t} - h p_j^{-1} \nabla_j f(y_{0,t}) e_j,$$

where $p_j = L_{1j}/\sum_j L_{1j}$.

- It is possible to extend the S2CD algorithm and results to the case when coordinates are replaced by (non-overlapping) blocks of coordinates, as in [15]—we did not do it here for the sake of keeping the notation simple. In such a setting, we would obtain semi-stochastic block coordinate descent. In the special case with all variables forming a single block, the algorithm reduces to the S2GD method described in [6], but with non-uniform probabilities for the choice of $i$—proportional to the Lipschitz constants of the gradient of the functions $f_i$ (this is also studied in [22]). As in [22], the complexity result then depends on the average of the Lipschitz constants.

Note that the algorithm, as presented, assumes knowledge of the strong convexity parameter $\mu$. We have done this for simplicity of exposition: the method works also if $\mu$ is not explicitly known—in that case, we can simply replace $\mu$ by 0 and the method will still depend on the true strong convexity parameter. The change to the complexity results will be only minor in constants and all our conclusions hold. Likewise, it is possible to give an $O(1/\epsilon)$ complexity result in the non-strongly convex case $f$ using standard regularization arguments (e.g. see [6]).
3. Complexity result

In this section, we state and describe our complexity result; the proof is provided in Section 5.

An important step in our analysis is proving a good upper bound on the variance of the (unbi-
ased) estimator $g_{ij}^{kt} = p_j^{-1} G_{ij}^{kt} e_j$ of $\nabla f(y_{k,t})$, one that we can ‘believe’ would diminish to zero as
the algorithm progresses. This is important for several reasons. First, as the method approaches
the optimum, we wish $g_{ij}^{kt}$ to be progressively closer to the true gradient, which in turn will be
close to zero. Indeed, if this was the case, then S2CD behaves like GD with fixed stepsize $h$
close to optimum. In particular, this would indicate that using fixed stepsizes makes sense.

In light of the above discussion, the following lemma plays a key role in our analysis:

**Lemma 3.1** The iterates of the S2CD algorithm satisfy

$$\mathbb{E}[\|g_{ij}^{kt}\|^2] \leq 4\hat{L}(f(y_{k,t}) - f(x_0)) + 4\hat{L}(f(x_k) - f(x_0)), \tag{10}$$

where

$$\hat{L} := \frac{1}{n} \sum_{j=1}^{d} v_j \left( \frac{5}{n} \sum_{j=1}^{d} \sum_{i=1}^{n} \omega_i L_{ij} \right). \tag{11}$$

The proof of this lemma can be found in Section 4.

Note that as $y_{k,t} \to x_*$ and $x_k \to x_*$, the bound (10) decreases to zero. This is the main feature
of modern fast stochastic gradient methods: the squared norm of the stochastic gradient estimate
progressively diminishes to zero, as the method progresses, in expectation. Therefore it is possi-
ble to use constant step-size in this type of algorithms. Note that the standard SGD method does
not have this property: indeed, there is no reason for $\mathbb{E}[\|\nabla f(x)\|^2]$ to be small even if $x = x_*$.

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

**Theorem 3.2** (Complexity of S2CD) If $0 < h < 1/(2\hat{L})$, then for all $k \geq 0$ we have:

$$\mathbb{E}[f(x_{k+1}) - f(x_0)] \leq \left( \frac{(1 - \mu h)^m}{(1 - (1 - \mu h)^m) (1 - 2\hat{L} h)} + \frac{2\hat{L} h}{1 - 2\hat{L} h} \right) \mathbb{E}[f(x_k) - f(x_0)]. \tag{12}$$

By analysing the above result (one can follow the steps in [6, Theorem 6]), we get the following useful corollary:

**Corollary 3.3** Fix the number of epochs $k \geq 1$, error tolerance $\epsilon \in (0, 1)$ and let $\Delta := \epsilon^{1/k}$
and

$$\hat{k} := \frac{\hat{L}}{\mu} \left( \frac{11}{\mu n} \sum_{j=1}^{d} \sum_{i=1}^{n} \omega_i L_{ij} \right). \tag{13}$$

If we run Algorithm 1 with stepsize $h$ and $m$ set as

$$h = \frac{\Delta}{(4 + 2\Delta)\hat{L}}, \quad m \geq \left( \frac{4}{\Delta} + 2 \right) \log \left( \frac{2}{\Delta} + 2 \right) \hat{k}, \tag{14}$$

then $\mathbb{E}[f(x_k) - f(x_0)] \leq \epsilon(f(x_0) - f(x_*))$. In particular, for $k = \lceil \log(1/\epsilon) \rceil$ we have $(1/\Delta) \leq \exp(1)$, and we can pick

$$k = \lceil \log(1/\epsilon) \rceil, \quad h = \frac{\Delta}{(4 + 2\Delta)\hat{L}} \approx \frac{1}{(4 \exp(1) + 2)\hat{L}} \approx \frac{1}{12.87\hat{L}}, \quad m \geq 26\hat{k}. \tag{15}$$
Remark 3.1 Note that in order to define \( h \) and \( m \) as in (14), we need to fix the target accuracy \( \epsilon \) and the number of iterations \( k \) beforehand.

If we run S2CD with the parameters set as in (15), then in each epoch the gradient of \( f \) is evaluated once (this is equivalent to \( n \) evaluations of \( \nabla f_i \)), and the partial derivative of some function \( f_i \) is evaluated \( 2m \approx 52\hat{\kappa} = O(\hat{\kappa}) \) times. If we let \( C_{\text{grad}} \) be the average cost of evaluating the gradient \( \nabla f_i \) and \( C_{\text{pd}} \) be the average cost of evaluating the partial derivative \( \nabla_j f_i \), then the total work of S2CD can be written as

\[
(nC_{\text{grad}} + mC_{\text{pd}})k \overset{(15)}{=} O \left( (nC_{\text{grad}} + \hat{\kappa}C_{\text{pd}}) \log \left( \frac{1}{\epsilon} \right) \right),
\]

The complexity results of methods such as S2GD/SVRG [5,6,22] and SAG/SAGA [1,17]—in a similar but not identical setup to ours (these papers assume \( f_i \) to be \( L_i \)-smooth)—can be written in a similar form:

\[
O \left( (nC_{\text{grad}} + \kappa C_{\text{grad}}) \log \left( \frac{1}{\epsilon} \right) \right),
\]

where \( \kappa = L_{\text{avg}}/\mu \) with \( L_{\text{avg}} := (1/n) \sum_i L_i \) [22] (or slightly weaker where \( \kappa = L_{\text{max}}/\mu \) with \( L_{\text{max}} := \max_i L_i \) [1,5,6,17]). The difference between our result (16) and existing results (17) is in the term \( \hat{\kappa}C_{\text{pd}} \)—previous results have \( \kappa C_{\text{grad}} \) in that place. This difference constitutes a trade-off: while \( \hat{\kappa} \geq \kappa \) (we comment on this below), we clearly have \( C_{\text{pd}} \leq C_{\text{grad}} \). The comparison of the quantities \( \kappa C_{\text{grad}} \) and \( \hat{\kappa}C_{\text{pd}} \) is in general not straightforward and problem dependent.

Let us now compare the condition numbers \( \hat{\kappa} \) and \( \kappa = L_{\text{avg}}/\mu \). It can be shown that (see [15])

\[
L_i \leq \sum_{j=1}^{d} L_{ij}
\]

and, moreover, this inequality can be tight. Since \( \omega_i \geq 1 \) for all \( i \), we have

\[
\hat{\kappa} = \frac{\hat{L}}{\mu} \overset{(11)}{=} \frac{1}{\mu n} \sum_{j=1}^{d} \sum_{i=1}^{n} \omega_i L_{ij} \geq \frac{1}{\mu n} \sum_{i=1}^{n} \sum_{j=1}^{d} L_{ij} \geq \frac{1}{\mu n} \sum_{i=1}^{n} L_i = \frac{L_{\text{avg}}}{\mu} = \kappa.
\]

Let us denote

\[
\bar{\omega} = \max_i \omega_i, \quad \bar{\omega} = \max_i \omega_i.
\]

That is, \( \omega \) and \( \bar{\omega} \) are, respectively, the smallest and largest number of coordinates that a subfunction depends on. In the case when

\[
L_i = \sum_{j=1}^{d} L_{ij},
\]

it is easy to see that

\[
\omega \kappa \leq \hat{\kappa} \leq \bar{\omega} \kappa.
\]

In addition, when (18) holds, \( \hat{\kappa} \) is smaller than \( \kappa_{\text{max}} := L_{\text{max}}/\mu \) if

\[
\bar{\omega} \sum_{i=1}^{n} L_i \leq n \max_i L_i.
\]
4. Proof of Lemma 3.1

We will prove the following stronger inequality:

$$ \mathbb{E}[\|g_{ij}^{\mu}\|^2] \leq 4\hat{L}(f(y_k) - f(x_*)) + 4 \left( \hat{L} - \frac{\mu}{\max_s p_s} \right)(f(x_k) - f(x_*)). $$

(19)

Lemma 3.1 follows by dropping the negative term.

**STEP 1.** We first break down the left-hand side of (19) into \(d\) terms each of which we will bound separately. By first taking expectation conditioned on \(j\) and then taking the full expectation, we can write:

$$ \mathbb{E}[\|g_{ij}^{\mu}\|^2] = \mathbb{E}[\mathbb{E}_i[\|p_j^{-1}G_{ij}^{\mu}e_j\|^2]] $$

$$ = \mathbb{E}[p_j^{-2}\mathbb{E}_i[(G_{ij}^{\mu})^2]] = \sum_{s=1}^{d} p_s^{-1}\mathbb{E}_i[(G_{ij}^{\mu})^2]. $$

(20)

**STEP 2.** We now further break each of these \(d\) terms into three pieces. That is, for each \(j = 1, \ldots, d\) we have:

$$ \mathbb{E}_i[(G_{ij}^{\mu})^2] = \mathbb{E}_i \left[ \left( \nabla_{ji}(x_k) + \frac{\nabla f_i(y_{k,j}) - \nabla f_i(x_k)}{nq_{ij}} + \frac{\nabla f_i(x_k) - \nabla f_i(x_\ast)}{nq_{ij}} \right)^2 \right] $$

$$ = \mathbb{E}_i \left[ \left( \frac{\nabla f_i(y_{k,j}) - \nabla f_i(x_\ast)}{nq_{ij}} + \nabla f(x_k) - \frac{\nabla f_i(x_k) - \nabla f_i(x_\ast)}{nq_{ij}} \right)^2 \right] $$

$$ \leq 2\mathbb{E}_i \left[ \left( \frac{\nabla f_i(y_{k,j}) - \nabla f_i(x_\ast)}{nq_{ij}} \right)^2 \right] + 2\mathbb{E}_i \left[ \left( \nabla f(x_k) - \frac{\nabla f_i(x_k) - \nabla f_i(x_\ast)}{nq_{ij}} \right)^2 \right] $$

$$ = 2\mathbb{E}_i \left[ \left( \frac{\nabla f_i(y_{k,j}) - \nabla f_i(x_\ast)}{nq_{ij}} \right)^2 \right] $$

$$ + 2\mathbb{E}_i \left[ \left( \frac{\nabla f_i(x_k) - \nabla f_i(x_\ast)}{nq_{ij}} \right)^2 \right] - 2(\nabla f(x_k) - \nabla f(x_\ast))^2, $$

(21)

where the last equality follows from the fact that

$$ \mathbb{E}_i \left[ \frac{\nabla f_i(x_k) - \nabla f_i(x_\ast)}{nq_{ij}} \right] = \sum_{i=1}^{n} q_{ij} \frac{\nabla f_i(x_k) - \nabla f_i(x_\ast)}{nq_{ij}} = \nabla f(x_k) - \nabla f(x_\ast). $$

**STEP 3.** In this step we bound the first two terms in the right-hand side of inequality (21). It will now be useful to introduce the following notation:

$$ Q_j := \{ i : L_{ij} \neq 0 \}, \quad j = 1, \ldots, d, $$

(22)

and

$$ 1_{ij} := \begin{cases} 1 & \text{if } L_{ij} \neq 0, \\ 0 & \text{otherwise}, \end{cases} \quad i = 1, \ldots, n, \quad j = 1, \ldots, d. $$
Let us first examine the first term in the right-hand side of (21). Using the coordinate co-coercivity lemma (Lemma 5.1) with $y = x_*$, we obtain the inequality

$$(\nabla f_i(x) - \nabla f_i(x_*))^2 \leq 2L_i(f_i(x) - f_i(x_*) - \langle \nabla f_i(x_*) , x - x_* \rangle),$$

using which we get the bound:

$$2 \sum_{s=1}^{d} p_s^{-1} E_i \left[ \left( \frac{1}{n q_{i,s}} (\nabla f_i(y_{k,t}) - \nabla f_i(x_*)) \right) ^2 \right]$$

$$= 2 \sum_{s=1}^{d} \frac{1}{n q_{i,s}} (\nabla f_i(y_{k,t}) - \nabla f_i(x_*))^2$$

$$\leq 4 \sum_{s=1}^{d} p_s^{-1} \sum_{i \in Q_s} \frac{L_i}{n^2 q_{i,s}} (f_i(y_{k,t}) - f_i(x_*) - \langle \nabla f_i(x_*), y_{k,t} - x_* \rangle)$$

$$= 4 \sum_{s=1}^{d} \sum_{i=1}^{n} p_s^{-1} \frac{v_s}{n^2 q_{i,s}} (f_i(y_{k,t}) - f_i(x_*) - \langle \nabla f_i(x_*), y_{k,t} - x_* \rangle).$$

(23)

Note that by (5) and (11), we have that for all $s = 1, 2, \ldots, d$,

$$p_s^{-1} v_s = n \hat{L}.$$

Continuing from (24), we can therefore further write

$$2 \sum_{s=1}^{d} p_s^{-1} E_i \left[ \left( \frac{1}{n q_{i,j}} (\nabla f_i(y_{k,t}) - \nabla f_i(x_*)) \right) ^2 \right]$$

$$\leq 4 \sum_{i=1}^{n} \sum_{s=1}^{d} \frac{\hat{L}}{n \omega_i} (f_i(y_{k,t}) - f_i(x_*) - \langle \nabla f_i(x_*), y_{k,t} - x_* \rangle)$$

$$= \frac{4 \hat{L}}{n} \sum_{i=1}^{n} (f_i(y_{k,t}) - f_i(x_*) - \langle \nabla f_i(x_*), y_{k,t} - x_* \rangle)$$

$$= 4 \hat{L}(f(y_{k,t}) - f(x_*)).$$

(24)

The same reasoning applies to the second term on the right-hand side of the inequality (21) and we have:

$$2 \sum_{s=1}^{d} p_s^{-1} E_i \left[ \left( \frac{1}{n q_{i,j}} (\nabla f_i(y_{k,t}) - \nabla f_i(x_*)) \right) ^2 \right] \leq 4 \hat{L}(f(y_{k,t}) - f(x_*)).$$

(25)

STEP 4. Next we bound the third term on the right-hand side of the inequality (21). First note that since $f$ is $\mu$-strongly convex (see (3)), for all $x \in \mathbb{R}^d$ we have:

$$\langle \nabla f(x), x - x_* \rangle \geq f(x) - f(x_*) + \frac{\mu}{2} \| x - x_* \|^2.$$
We can now write:

\[
2 \sum_{s=1}^{d} p_s^{-1} (\nabla f(x_k) - \nabla f(x_s))^2 \geq \frac{2}{\max_s p_s} \sum_{j=1}^{d} (\nabla f(x_k) - \nabla f(x_s))^2 \geq \frac{4\mu}{\max_s p_s} (f(x_k) - f(x_s)). \tag{28}
\]

**STEP 5.** We conclude by combining (20), (21), (25), (26) and (28).

5. **Proof of the main result**

In this section we provide the proof of our main result. In order to present the proof in an organized fashion, we first establish two technical lemmas.

5.1 **Coordinate co-coercivity**

It is a well known and widely used fact (see, e.g. [9]) that for a continuously differentiable function \( \phi : \mathbb{R}^d \rightarrow \mathbb{R} \) and constant \( L_\phi > 0 \), the following two conditions are equivalent:

\[
\phi(x) \leq \phi(y) + \langle \nabla \phi(y), x - y \rangle + \frac{L_\phi}{2} \|x - y\|^2, \quad \forall x, y \in \mathbb{R}^d
\]

and

\[
\|\nabla \phi(x) - \nabla \phi(y)\|^2 \leq 2L_\phi (\phi(x) - \phi(y) - \langle \nabla \phi(y), x - y \rangle), \quad \forall x, y \in \mathbb{R}^d.
\]

The second condition is often referred to by the name co-coercivity. Note that our assumption (2) on \( f_i \) is similar to the first inequality. In our first lemma we establish a coordinate-based co-coercivity result which applies to functions \( f_i \) satisfying (2).

**Lemma 5.1 (Coordinate co-coercivity)** For all \( x, y \in \mathbb{R}^d \) and \( i = 1, \ldots, n \), \( j = 1, \ldots, d \), we have:

\[
(\nabla f_i(x) - \nabla f_i(y))^2 \leq 2L_{ij}(f_i(x) - f_i(y) - \langle \nabla f_i(y), x - y \rangle). \tag{29}
\]

**Proof** Fix any \( i, j \) and \( y \in \mathbb{R}^d \). Consider the function \( g_i : \mathbb{R}^d \rightarrow \mathbb{R} \) defined by:

\[
g_i(x) := f_i(x) - f_i(y) - \langle \nabla f_i(y), x - y \rangle. \tag{30}
\]

Then since \( f_i \) is convex, we know that \( g_i(x) \geq 0 \) for all \( x \), with \( g_i(y) = 0 \). Hence, \( y \) minimizes \( g_i \). We also know that for any \( x \in \mathbb{R}^d \):

\[
\nabla_j g_i(x) = \nabla f_i(x) - \nabla f_i(y). \tag{31}
\]

Since \( f_i \) satisfies (2), so does \( g_i \), and hence for all \( x \in \mathbb{R}^d \) and \( h \in \mathbb{R} \), we have

\[
g_i(x + he_j) \leq g_i(x) + \langle \nabla g_i(x), he_j \rangle + \frac{L_{ij}}{2} h^2.
\]

Minimizing both sides in \( h \), we obtain

\[
g_i(y) \leq \min_h g_i(x + he_j) \leq g_i(x) - \frac{1}{2L_{ij}} (\nabla_j g_i(x))^2,
\]

which together with (30) yields the result. 
5.2 Recursion

We now proceed to the final lemma, establishing a key recursion which ultimately yields the proof of the main theorem, which we present in Section 5.3.

Lemma 5.2 (Recursion) The iterates of S2CD satisfy the following recursion:

\[
\begin{align*}
\frac{1}{2} \mathbb{E}[\|y_{k+1} - x_*\|^2] + h(1 - 2h\hat{L})(f(y_{k+1}) - f(x_*)) \\
\leq (1 - \mu h) \frac{1}{2} \|y_k - x_*\|^2 + 2h^2\hat{L}(f(x_k) - f(x_*)).
\end{align*}
\] (32)

Proof

\[
\begin{align*}
\frac{1}{2} \mathbb{E}[\|y_{k+1} - x_*\|^2] &\equiv \frac{1}{2} \mathbb{E}[\|y_k - x_*\|^2 - \mathbb{E}[hp_j^{-1}G_{kj}^i e_j | y_k, t, x_*]] + \frac{1}{2} \mathbb{E}[\|hp_j^{-1}G_{kj}^i e_j\|^2] \\
&\equiv \frac{1}{2} \|y_k - x_*\|^2 - h\mathbb{E}[(\nabla f(y_k), y_k - x_*)] + \frac{h^2}{2} \mathbb{E}[\|s_{kj}\|^2] \\
&\leq \frac{1}{2} \|y_k - x_*\|^2 - h\left(f(y_k) - f(x_*) + \frac{\mu}{2} \|y_k - x_*\|^2\right) + \frac{h^2}{2} \mathbb{E}[\|s_{kj}\|^2] \\
&\leq \frac{1}{2} \|y_k - x_*\|^2 - h\left(f(y_k) - f(x_*) + \frac{\mu}{2} \|y_k - x_*\|^2\right) + 2h^2\hat{L}(f(y_k) - f(x_*)) + 2h^2\hat{L}(f(x_k) - f(x_*)) \\
&= (1 - \mu h) \frac{1}{2} \|y_k - x_*\|^2 - h(1 - 2h\hat{L})(f(y_k) - f(x_*)) \\
&\quad + 2h^2\hat{L}(f(x_k) - f(x_*)).
\end{align*}
\]

5.3 Proof of Theorem 3.2

For simplicity, let us denote:

\[
\eta_{k,t} := \frac{1}{2} \mathbb{E}[\|y_{k,t} - x_*\|^2], \quad \xi_{k,t} := \mathbb{E}[f(y_{k,t}) - f(x_*)],
\]

where the expectation now is with respect to the entire history. Notice that

\[
y_{k+1,0} = y_{k,t},
\]

where \(t_k = T \in \{1, \ldots, m\}\) with probability \((1 - \mu h)^{m-T}/\beta\) with \(\beta\) defined in (4). Conditioning on \(t_k\) we obtain that

\[
\xi_{k+1,0} = \frac{1}{\beta} \sum_{t=0}^{m-1} (1 - \mu h)\xi_{k,m-1-t}.
\] (33)
See also [6, Lemma 3] for a proof. By Lemma 5.2 we have the following m inequalities:

\[ \eta_{k,m} + h(1 - 2h\hat{L})\xi_{k,m-1} \leq (1 - \mu h)\eta_{k,m-1} + 2h^2\hat{L}\xi_{k,0}, \]

\[ (1 - \mu h)\eta_{k,m-1} + h(1 - 2h\hat{L})(1 - \mu h)\xi_{k,m-2} \leq (1 - \mu h)^2\eta_{k,m-2} + 2h^2\hat{L}(1 - \mu h)\xi_{k,0}, \]

\[ \vdots \]

\[ (1 - \mu h)^m\eta_{k,1} + \gamma(1 - 2h\hat{L})(1 - \mu h)^{m-1}\xi_{k,0} \leq (1 - \mu h)^m\eta_{k,0} + 2h^2\hat{L}(1 - \mu h)^{m-1}\xi_{k,0}. \]

By summing up the above m inequalities, we get:

\[ \eta_{k,m} + \gamma(1 - 2h\hat{L})\sum_{t=0}^{m-1} (1 - \mu h)^t\xi_{k,m-1-t} \leq (1 - \mu h)^m\eta_{k,0} + 2h^2\hat{L}\beta\xi_{k,0}. \]

It follows from the strong convexity assumption (3) that \( f(x_k) - f(x_*) \geq (\mu/2)\|x_k - x_*\|^2 \), that is, \( \xi_{k,0} \geq \mu\eta_{k,0} \). Therefore, together with (33) we get:

\[ h(1 - 2h\hat{L})\xi_{k+1,0} \leq \left( \frac{(1 - \mu h)^m}{\beta\mu} + 2h^2\hat{L} \right)\xi_{k,0}. \]

Hence if \( 0 < 2h\hat{L} < 1 \), then we obtain:

\[ \xi_{k+1,0} \leq \left( \frac{(1 - \mu h)^m}{(1 - (1 - \mu h)^m)(1 - 2h\hat{L})} + \frac{2h\hat{L}}{1 - 2h\hat{L}} \right)\xi_{k,0}, \]

which finishes the proof.

6. Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by Engineering and Physical Sciences Research Council [grant number EP/K02325X/1].

Notes

1. The complexity can be improved to \( O((da/\tau\mu)\log(1/\epsilon)) \) in the case when \( \tau \) coordinates are updated in each iteration, where \( \alpha \in [1, \tau] \) is a problem-dependent constant [12]. This has been further studied for non-smooth problems via smoothing [3], for arbitrary non-uniform distributions governing the selection of coordinates [11,14] and in the distributed setting [4,11,13]. Also, efficient accelerated variants with \( O(1/\sqrt{\epsilon}) \) rate were developed [2,4], capable of solving problems with 50 billion variables.

2. In S2CD, as presented, coordinates \( j \) is selected first, and then function \( i \) is selected, according to a distribution conditioned on the choice of \( j \). However, one could equivalently sample \( (i,j) \) with joint probability \( p_{ij} \). We opted for the sequential sampling for clarity of presentation purposes.

3. The quantity \( \omega := \max_{i,j} \omega_{ij} \) (degree of partial separability of \( f \)) was used in the analysis of a large class of randomized parallel coordinate descent methods in [12]. The more informative quantities \( \{\omega_{ij}\} \) appear in the analysis of parallel/distributed/mini-batch coordinate descent methods [2,4,13].
4. A parallel CD method in which every subset of coordinates can be assigned a different probability of being chosen/updated was analysed in [14].

5. It is possible to modify the argument slightly and replace the term $\hat{L}$ appearing in the numerator by $\hat{L} - \mu / \max_{p_s}$. However, as this does not bring any significant improvements, we decided to present the result in this simplified form.

References