Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 21, No. 2, pp. 327–338, 2025

AN ADAPTIVE ACCELERATED COORDINATE DESCENT METHOD WITH NON-UNIFORM SAMPLING

Yu You and Jirui Ma

Abstract: We consider the optimization problem of minimizing a smooth and convex function. Based on the accelerated coordinate descent method (ACDM) using probabilities $L_i^{1/2}[\sum_{k=1}^n L_k^{1/2}]^{-1}$ for non-uniform sampling (Nesterov Yu. et al., SIAM J. Optim., 110–123, 2017 [15]), we propose an adaptive accelerated coordinate descent method (AACDM) with the same probability distribution determined by L_i as in ACDM. However, AACDM dynamically backtracks the coordinate Lipschitz constants of the problem to approximate of local smoothness of the problem better. Not only the non-monotone version of AACDM but also the monotone version are presented in the paper. The convergence rate of the proposed method is studied. Numerical results on some classic problems show the efficiency of the adaptive scheme.

Key words: *coordinate decent method, non-uniform sampling, non-monotone backtracking, convex optimization*

Mathematics Subject Classification: *65K05, 90C25*

1 Introduction

Coordinate descent-type methods perform exact or approximate minimization along (block) coordinate directions. They get more and more popular in machine learning and large-scale optimization due to their low computational cost and favorable performance [9, 18, 1]. Coordinate descent-type methods can be classified by the update rule of the working coordinates set, such as an exact minimization of the working set as in alternative minimization [16], the gradient or proximal gradient step as in block gradient methods [12, 6, 14, 15, 2, 17], and the conditional gradient step as in block conditional gradient methods [4, 10].

In this paper, we consider the unconstrained optimization problem in the form of

$$\min_{\mathbf{x}\in\mathbb{R}^n} H(\mathbf{x}), \tag{1.1}$$

where the following are assumed throughout the paper:

Assumption 1.1. (a) H has componentwise Lipschitz continuous gradient, i.e., for $i = 1, 2, \ldots, n$,

$$|\nabla_i H(\mathbf{x} + t\mathbf{e}_i) - \nabla_i H(\mathbf{x})| \le L_i|t|, \quad \forall \mathbf{x} \in \mathbb{R}^n, \quad \forall t \in \mathbb{R}, \tag{1.2}$$

where $\mathbf{e}_i$ is the i-th coordinate vector in $\mathbb{R}^n$, $L_i > 0$ and $\nabla_i H(\mathbf{x})$ is the i-th coordinate gradient.

DOI: https:// doi.org/10.61208/pjo-2024-011

(b) H is σ-strongly ($\sigma \geq 0$) convex, i.e.,

$$H(\mathbf{y}) \geq H(\mathbf{x}) + \langle \nabla H(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{\sigma}{2}\|\mathbf{x} - \mathbf{y}\|^2, \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \tag{1.3}$$

where $\nabla H(\mathbf{x})$ is the gradient of H at $\mathbf{x}$.

(c) The optimal solution set of (1.1) is nonempty.

A classical method for solving (1.1) when $\sigma = 0$ is the iterative scheme

$$\mathbf{x}^0 \in \mathbb{R}^n, \text{ and for } k \geq 0, \ \mathbf{x}^{k+1} = \mathbf{x}^k - \frac{1}{L_{\max}} \nabla_{i_k} H(\mathbf{x}^k)\mathbf{e}_{ik}, \tag{1.4}$$

where $L_{\max} = \max\{L_1, L_2, \ldots, L_n\}$ and $i_k \in \arg\max_{1 \leq i \leq n} |\nabla_i H(\mathbf{x}^k)|$. It enjoys an $O(nL_{\max}/k)$ rate of convergence in terms of function values [14]. Note that at each iteration, (1.4) requires to compute the full gradient vector, which could be expensive or intractable for large-scale problems. To address this issue, some random coordinate gradient descent methods were developed. For example, the randomized coordinate descent method (RCDM) with different sampling strategies was proposed and an accelerated variant of RCDM with uniform sampling achieved an $O((nL_{\max})^2/k^2)$ rate of convergence [14]. In [11], another accelerated variant of RCDM was presented and attained the convergence rate of $O((n\sum_{i=1}^n L_i)/k^2)$ by using probabilities $L_i[\sum_{k=1}^n L_k]^{-1}$ for selecting active coordinates. Next, the accelerated coordinate descent method (ACDM) proposed in [2, 15] further improved the convergence rate to $O((\sum_{i=1}^n L_i^{1/2})^2/k^2)$, which does not depend on the number of variables, by using probabilities $L_i^{1/2}[\sum_{k=1}^n L_k^{1/2}]^{-1}$ for non-uniform sampling.

In [13], some greedy strategies for selecting the active coordinate were proposed, which require to compute the full gradient vector and can make better performance in practice when the cost of full gradient vector is cheap. In [17], an optimal sampling strategy and a suboptimal method with cheap implementation were given. Some other random coordinate descent methods were developed for solving the composite convex optimization model, i.e., the sum of a smooth convex function and a proper closed convex function with easily computable proximal term of the latter. Interested readers are referred to [12, 11].

In [2, 15], the step sizes of their algorithms for (1.1) under Assumption 1.1 are fixed and determined by the (global) parameters $\{L_i\}$. Note that this may not be preferable for practical applications where the (local) parameter values differ from the global counterparts to some extent. This implies that methods which can be adaptive to the local parameters might improve the performance in practice. Motivated by this, in this paper we study the adaptive ACDM for (1.1), which still requires (global) Lipschitz constants for non-uniform sampling as a prior, while the (local) coordinate Lipschitz constants are determined by backtracking (not neceessarily monotone) to achieve better performance. Both the strongly and non-strongly cases are discussed in this paper. The non-monotone backtracking line search is included in our adaptive scheme, which performs better (compared with the monotone one) for applications whose local coordinate Lipschitz constants oscillate along the trajectory or become smaller when approaching the tail. The adaptive ACDM is indeed not a monotone method, meaning that the sequence of function values it produces is not necessarily nonincreasing. Since the monotone approach can be used to improve numerical stability (see monotone FISTA in [5]), we also propose an adaptive ACDM in monotone version.

The paper is organized as follows. In section 2 we present the adaptive ACDM in nonmonotone version as well as the one in monotone version for solving (1.1). In section 3 we establish the convergence rate of the adaptive ACDM in non-monotone version when

the objective function is σ-strongly convex ($\sigma \geq 0$). In section 4 we report the numerical experiments on some classical problems. Finally in section 5 we discuss the adaptive ACDMs for (1.1) when the objective function is non-strongly convex.

2 Adaptive ACDMs

We first present an adaptive ACDM in non-monotone version by using probabilities $L_i^{1/2}[\sum_{k=1}^n L_k^{1/2}]^{-1}$ for non-uniform sampling, then present the one in monotone version. Here, $\mathbb{N}_+$ and $\mathbb{N}$ denote the set of positive integers and the set of non-negative integers, respectively.

Algorithm 2.1. Adaptive ACDM in non-monotone version

Input: $S_{1/2} = \sum_{i=1}^n \sqrt{L_i}$, $p_i = \frac{\sqrt{L_i}}{S_{1/2}}$ for $i \in \{1, 2, \cdots, n\}$, $\delta > 1$, $t_0 \in \mathbb{N}_+$ such that $\sigma < S_{1/2}^2 \delta^{-t_0}$ ($t_0 = \infty$ when $\sigma = 0$), $\mathbf{x}^0 = \mathbf{v}^0 \in \mathbb{R}^n$, and $A_0 = 0, B_0 = 1$.

1. For $k = 0, 1, 2, \cdots$

2. Generate $i_k \in \{1, 2, \cdots, n\}$ via probability distribution $(p_i)_i$;

3. **Repeat** step 4-6

4. Choose $t_{k+1} \leq t_0, t_{k+1} \in \mathbb{N}$, and compute $a_{k+1} > 0$ by solving the equation

$$a_{k+1}^2 S_{1/2}^2 \delta^{-t_{k+1}} = A_{k+1} B_{k+1}, \tag{2.1}$$

where $A_{k+1} = A_k + a_{k+1}$ and $B_{k+1} = B_k + \sigma a_{k+1}$;

5. Define $\alpha_k = \frac{a_{k+1}}{A_{k+1}}, \beta_k = \frac{\sigma a_{k+1}}{B_{k+1}}$, and $\mathbf{y}^k = \frac{(1-\alpha_k)\mathbf{x}^k + \alpha_k(1-\beta_k)\mathbf{v}^k}{1-\alpha_k\beta_k}$;

6. Set $\mathbf{x}^{k+1} = \mathbf{y}^k - \frac{1}{\delta^{-t_{k+1}} L_{i_k}} \nabla_{i_k} H(\mathbf{y}^k) \mathbf{e}_{i_k}$;

7. **Until**

$$H(\mathbf{x}^{k+1}) \leq H(\mathbf{y}^k) - \frac{1}{2\delta^{-t_{k+1}} L_{i_k}} (\nabla_{i_k} H(\mathbf{y}^k))^2$$

8. Set $\mathbf{v}^{k+1} = (1-\beta_k)\mathbf{v}^k + \beta_k \mathbf{y}^k - \frac{a_{k+1}}{B_{k+1} p_{i_k}} \nabla_{i_k} H(\mathbf{y}^k) \mathbf{e}_{i_k}$.

9. End for

Remark 2.2. Algorithm 2.1 is non-monotone, hence $H(\mathbf{x}^{k+1}) \leq H(\mathbf{x}^k)$ may not hold for some k. Moreover, it follows from [3, Theorem 11.20] that $\sigma \leq \sum_{i=1}^n L_i$. Hence $\sigma < S_{1/2}^2$. So the choice of t_{k+1} implies that $\sigma < S_{1/2}^2 \delta^{-t_{k+1}}$ for all k.

Remark 2.3. If $H(x)$ is not strongly convex($\sigma = 0$), t_0 is not constrained by t_0 in step 4.

Algorithm 2.4. Adaptive ACDM in monotone version

Input: $S_{1/2} = \sum_{i=1}^n \sqrt{L_i}$, $p_i = \frac{\sqrt{L_i}}{S_{1/2}}$ for $i \in \{1, 2, \cdots, n\}$, $\delta > 1$, $t_0 \in \mathbb{N}_+$ such that $\sigma < S_{1/2}^2 \delta^{-t_0}$ ($t_0 = \infty$ when $\sigma = 0$), $\mathbf{x}^0 = \mathbf{v}^0 \in \mathbb{R}^n$, and $A_0 = 0, B_0 = 1$.

1. For $k = 0, 1, 2, \cdots$

2. Generate $i_k \in \{1, 2, \cdots, n\}$ via probability distribution $(p_i)_i$;

3. **Repeat** step 4-6

4. Choose $t_{k+1} \leq t_0, t_{k+1} \in \mathbb{N}$, and compute $a_{k+1} > 0$ by solving the equation

$$a_{k+1}^2 S_{1/2}^2 \delta^{-t_{k+1}} = A_{k+1} B_{k+1},$$

where $A_{k+1} = A_k + a_{k+1}$ and $B_{k+1} = B_k + \sigma a_{k+1}$;

5. Define $\alpha_k = \frac{a_{k+1}}{A_{k+1}}, \beta_k = \frac{\sigma a_{k+1}}{B_{k+1}}$, and $\mathbf{y}^k = \frac{(1-\alpha_k)\mathbf{x}^k + \alpha_k(1-\beta_k)\mathbf{v}^k}{1-\alpha_k\beta_k}$;

6. Set $\tilde{\mathbf{x}}^{k+1} = \mathbf{y}^k - \frac{1}{\delta^{-t_{k+1}} L_{i_k}} \nabla_{i_k} H(\mathbf{y}^k) \mathbf{e}_{i_k}$;

7. **Until**

$$H(\tilde{\mathbf{x}}^{k+1}) \leq H(\mathbf{y}^k) - \frac{1}{\delta^{-t_{k+1}} 2L_{i_k}} (\nabla_{i_k} H(\mathbf{y}^k))^2$$

8. Choose $\mathbf{x}^{k+1} \in \mathbb{R}^n$ such that $H(\mathbf{x}^{k+1}) \leq \min\{H(\tilde{\mathbf{x}}^{k+1}), H(\mathbf{x}^k)\}$

9. $\mathbf{v}^{k+1} = (1-\beta_k)\mathbf{v}^k + \beta_k \mathbf{y}^k - \frac{a_{k+1}}{B_{k+1} p_{i_k}} \nabla_{i_k} H(\mathbf{y}^k) \mathbf{e}_{i_k}$.

10. End for

Remark 2.5. In step 4 of Algorithm 2.4, given an initial guess t_{k+1}, for example t_k, one may update t_{k+1} by $t_{k+1} - 1$ until step 7 is satisfied. The existence of $t_{k+1} \in \mathbb{N}$ can be verified by setting $t_{k+1} = 0$, and $H(y^k) - H(x^{k+1}) \geq \frac{1}{2L_{i_k}} (\nabla_{i_k} H(y^k))^2$ is satisfied as [3, Lemma 11.9]. So the backtracking procedure terminates within finite steps.

Remark 2.6. One may choose

$$\mathbf{x}^{k+1} = \begin{cases} \tilde{\mathbf{x}}^{k+1}, & \text{if } H(\tilde{\mathbf{x}}^{k+1}) \leq H(\mathbf{x}^k), \\ \mathbf{x}^k - \frac{1}{L_{i_k}} \nabla f(\mathbf{x}^k), & \text{if } H(\tilde{\mathbf{x}}^{k+1}) > H(\mathbf{x}^k) \end{cases} \tag{2.2}$$

as an implementation of monotone approach in step 8 of Algorithm 2.4.

3 Convergence Analysis

In this section, we study the convergence rate of Algorithm 2.1. We use the notation $\xi_k = \{i_0, i_1, \cdots, i_k\}$.

Theorem 3.1. *Suppose that Assumption 1.1 holds and* $\mathbf{x}^*$ *is an optimizer of* (1.1). *Let* $\{\mathbf{x}^k\}_{k=0}^{\infty}$ *and* $\{\mathbf{v}^k\}_{k=0}^{\infty}$ *be the sequences generated by Algorithm 2.1. Then, for any* $k \geq 0$,

(i)

$$\begin{aligned} &2A_{k+1} E_{\xi_k}[H(\mathbf{x}^{k+1}) - H(\mathbf{x}^*)] + B_{k+1} \mathbb{E}_{\xi_k}[\|\mathbf{v}^{k+1} - \mathbf{x}^*\|^2] \\ \leq &2A_k E_{\xi_{k-1}}[H(\mathbf{x}^k) - H(\mathbf{x}^*)] + B_k \mathbb{E}_{\xi_{k-1}}[\|\mathbf{v}^k - \mathbf{x}^*\|^2]; \end{aligned} \tag{3.1}$$

(ii) if $\sigma = 0$, *then*

$$E_{\xi_k}[H(\mathbf{x}^{k+1}) - H(\mathbf{x}^*)] \leq \frac{2S_{1/2}^2 \|\mathbf{x}^0 - \mathbf{x}^*\|^2}{(k+1)^2}; \tag{3.2}$$

if $\sigma > 0$, *then*

$$E_{\xi_k}[H(\mathbf{x}^{k+1}) - H(\mathbf{x}^*)] \leq S_{1/2}^2 (1 - \sqrt{\sigma}/S_{1/2})^{k+1} \|\mathbf{x}^0 - \mathbf{x}^*\|^2. \tag{3.3}$$

Proof. (i) Define $\mathbf{w}^k = (1-\beta_k)\mathbf{v}^k + \beta_k \mathbf{y}^k$. Then

$$\mathbf{y}^k = \frac{(1-\alpha_k)\mathbf{x}^k}{1-\alpha_k\beta_k} + \frac{\alpha_k(1-\beta_k)}{1-\alpha_k\beta_k} \frac{\mathbf{w}^k - \beta_k \mathbf{y}^k}{1-\beta_k} = \frac{(1-\alpha_k)\mathbf{x}^k + \alpha_k \mathbf{w}^k}{1-\alpha_k\beta_k} - \frac{\alpha_k\beta_k \mathbf{y}^k}{1-\alpha_k\beta_k},$$

thus $\mathbf{y}^k = (1-\alpha_k)\mathbf{x}^k + \alpha_k \mathbf{w}^k$. Denote $r_k^2 = \|\mathbf{v}^k - \mathbf{x}^*\|^2$, then

$$\begin{aligned}\|\mathbf{v}^{k+1} - \mathbf{x}^*\|^2 =& \left\|\mathbf{w}^k - \frac{a_{k+1}}{B_{k+1}p_{i_k}}\nabla_{i_k}H(\mathbf{y}^k)\mathbf{e}_{i_k} - \mathbf{x}^*\right\|^2 \\ =&\|\mathbf{w}^k - \mathbf{x}^*\|^2 - \frac{2a_{k+1}}{B_{k+1}p_{i_k}}\langle \nabla_{i_k}H(\mathbf{y}^k), \mathbf{w}_{i_k}^k - \mathbf{x}_{i_k}^*\rangle \\ &+ \frac{a_{k+1}^2}{B_{k+1}^2 p_{i_k}^2}(\nabla_{i_k}H(\mathbf{y}^k))^2. \end{aligned} \tag{3.4}$$

On the other hand

$$\begin{aligned}\|\mathbf{w}^k - \mathbf{x}^*\|^2 &= \|(1-\beta_k)(\mathbf{v}^k - \mathbf{x}^*) + \beta_k(\mathbf{y}^k - \mathbf{x}^*)\|^2 \\ &\le (1-\beta_k)r_k^2 + \beta_k\|\mathbf{y}^k - \mathbf{x}^*\|^2. \end{aligned} \tag{3.5}$$

Hence, by (2.1), (3.4) and (3.5),

$$\begin{aligned}B_{k+1}r_{k+1}^2 \le& B_k r_k^2 + \beta_k B_{k+1}\|\mathbf{y}^k - \mathbf{x}^*\|^2 - \frac{2a_{k+1}}{p_{i_k}}\langle \nabla_{i_k}H(\mathbf{y}^k), \mathbf{w}_{i_k}^k - \mathbf{x}_{i_k}^*\rangle \\ &+ \frac{2a_{k+1}^2 L_{i_k}\delta^{-t_{k+1}}}{B_{k+1}p_{i_k}^2}(H(\mathbf{y}^k) - H(\mathbf{x}^{k+1})) \\ =& B_k r_k^2 + \beta_k B_{k+1}\|\mathbf{y}^k - \mathbf{x}^*\|^2 - \frac{2a_{k+1}}{p_{i_k}}\langle \nabla_{i_k}H(\mathbf{y}^k), \mathbf{w}_{i_k}^k - \mathbf{x}_{i_k}^*\rangle \\ &+ 2A_{k+1}(H(\mathbf{y}^k) - H(\mathbf{x}^{k+1})).\end{aligned}$$

Taking expectation with i_k, we obtain

$$\begin{aligned}B_{k+1}\mathbb{E}_{i_k}(r_{k+1}^2) \le& B_k r_k^2 + \sigma a_{k+1}\|\mathbf{y}^k - \mathbf{x}^*\|^2 + 2a_{k+1}\langle \nabla H(\mathbf{y}^k), \mathbf{x}^* - \mathbf{w}^k\rangle \\ &+ 2A_{k+1}(H(\mathbf{y}^k) - \mathbb{E}_{i_k}(H(\mathbf{x}^{k+1}))).\end{aligned} \tag{3.6}$$

It then follows from $\mathbf{w}^k = \mathbf{y}^k + \frac{1-\alpha_k}{\alpha_k}(\mathbf{y}^k - \mathbf{x}^k)$, $\alpha_k = \frac{a_{k+1}}{A_{k+1}}$, $A_{k+1} = A_k + a_{k+1}$ and (1.3) that

$$\begin{aligned}a_{k+1}\langle \nabla H(\mathbf{y}^k), \mathbf{x}^* - \mathbf{w}^k\rangle &= a_{k+1}\langle \nabla H(\mathbf{y}^k), \mathbf{x}^* - \mathbf{y}^k + \frac{1-\alpha_k}{\alpha_k}(\mathbf{x}^k - \mathbf{y}^k)\rangle \\ \le& a_{k+1}(H(\mathbf{x}^*) - H(\mathbf{y}^k)) - \frac{1}{2}a_{k+1}\sigma\|\mathbf{y}^k - \mathbf{x}^*\|^2 + a_{k+1}\frac{1-\alpha_k}{\alpha_k}(H(\mathbf{x}^k) - H(\mathbf{y}^k)) \\ =& a_{k+1}H(\mathbf{x}^*) - A_{k+1}H(\mathbf{y}^k) + A_k H(\mathbf{x}^k) - \frac{1}{2}a_{k+1}\sigma\|\mathbf{y}^k - \mathbf{x}^*\|^2. \end{aligned} \tag{3.7}$$

Substituting (3.7) into (3.6), we have

$$B_{k+1}\mathbb{E}_{i_k}(r_{k+1}^2) \le B_k r_k^2 + 2A_k(H(\mathbf{x}^k) - H(\mathbf{x}^*)) - 2A_{k+1}(\mathbb{E}_{i_k}(H(\mathbf{x}^{k+1}) - H(\mathbf{x}^*))).$$

Taking expectation over ξ_{k-1} yields (3.1).

(*ii*) Inequality (3.1) implies that

$$E_{\xi_k}[H(\mathbf{x}^{k+1}) - H(\mathbf{x}^*)] \le \frac{\|\mathbf{x}^0 - \mathbf{x}^*\|^2}{2A_{k+1}}. \tag{3.8}$$

It suffices to estimate the growth of A_k. It can be prove by induction that $B_k = 1 + \sigma A_k$. Hence, by (2.1),

$$(A_{k+1} - A_k)^2 S_{1/2}^2 \delta^{-t_{k+1}} = A_{k+1}(1 + \sigma A_{k+1}). \tag{3.9}$$

For $\sigma = 0$, we prove by induction that

$$A_k \geq \frac{k^2}{4S_{1/2}^2}, \quad \forall k \geq 0. \tag{3.10}$$

Obviously, (3.10) holds for $k = 0$. Suppose it holds for k. By (3.9),

$$\begin{aligned} A_{k+1} &= A_k + \frac{1/(S_{1/2}^2\delta^{-t_{k+1}}) + \sqrt{4A_k/(S_{1/2}^2\delta^{-t_{k+1}}) + 1/(S_{1/2}^4\delta^{-2t_{k+1}})}}{2} \\ &\geq A_k + 1/(2S_{1/2}^2) + \sqrt{A_k}/S_{1/2} \\ &\geq \frac{k^2}{4S_{1/2}^2} + 1/(2S_{1/2}^2) + \frac{k}{2S_{1/2}^2} = \frac{1}{4S_{1/2}^2}(k^2 + 2k + 2) \\ &\geq \frac{(k+1)^2}{4S_{1/2}^2}. \end{aligned}$$

Thus (3.10) holds for all $k \geq 0$. This, together with (3.8), yields (3.2).

For $\sigma > 0$, by (3.9),

$$\begin{aligned} A_{k+1} &= \frac{2A_k + 1/(S_{1/2}^2\delta^{-t_{k+1}}) + \sqrt{4A_k^2\frac{\sigma}{S_{1/2}^2\delta^{-t_{k+1}}} + \frac{4A_k}{S_{1/2}^2\delta^{-t_{k+1}}} + \frac{1}{S_{1/2}^4\delta^{-2t_{k+1}}}}}{2(1 - \sigma/(S_{1/2}^2\delta^{-t_{k+1}}))} \\ &\geq \frac{2A_k + 2A_k\frac{\sqrt{\sigma_0}}{S_{1/2}}}{2(1 - \sigma/S_{1/2}^2)} = \frac{A_k}{1 - \frac{\sqrt{\sigma}}{S_{1/2}}}. \end{aligned}$$

Since $A_1 = \frac{1/(S_{1/2}^2\delta^{-t_1})}{1-\sigma/(S_{1/2}^2\delta^{-t_1})} \geq \frac{1/(2S_{1/2}^2)}{1-\sqrt{\sigma}/S_{1/2}}$, we have

$$A_{k+1} \geq \frac{1/(2S_{1/2}^2)}{(1 - \sqrt{\sigma}/S_{1/2})^{k+1}}, \quad \forall k \geq 0, \tag{3.11}$$

which, together with (3.8), implies (3.3). □ □

The proof of (i) of Theorem 3.1 is similar to that in [15]. In fact, Algorithm 2.4 has the same convergence rate as Algorithm 2.1. Since the analysis is almost the same, we omit the proof here. Interested readers are referred to [8, 1] for more details.

4 Numerical Results

In this section, we test Algorithm 2.1 (the adaptive variant of ACDM proposed in [15], denoted as AACDM) and Algorithm 2.4 (the adaptive ACDM in monotone version, denoted as AACDM_M) on some classic problems, and compare them with ACDM and ACDM in monotone version (denoted as ACDM_M). The specific adaptive method in AACDM and AACDM_M is that at iterate k we set t_{k+1} as the previously returned t_k if k is not divisible by 5 and set it as $t_k + 1$ otherwise; moreover, δ is fixed as 2 in our test. All experiments are implemented in MATLAB 2022a on a 64-bit PC with an AMD Ryzen 5600U CPU (2.30GHz) and 16GB of RAM.

4.1 Test problem given in [15]

Considered the test problem with randomly generated data given in [15]:

$$\min_{\mathbf{x}\in\mathbb{R}^M} f_\mu(\mathbf{x}) = \sum_{i=1}^{N} \phi_\mu(\langle \mathbf{a}^i, \mathbf{x}\rangle - \mathbf{c}_i), \tag{4.1}$$

where

$$\phi_\mu(\tau) = \begin{cases} \frac{\tau^2}{2\mu}, & \text{if } |\tau| \le \mu, \\ |\tau| - \frac{1}{2}\mu, & \text{if } |\tau| > \mu. \end{cases} \tag{4.2}$$

Coefficients of dense vector $\mathbf{a}^i$ are uniformly distributed in the interval $[1, 2]$. Coefficients of vector $\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_N)^T \in \mathbb{R}^N$ are chosen as $\mathbf{c}_i = \langle \mathbf{a}^i, \bar{\mathbf{y}}\rangle$ where the entries of vector $\bar{\mathbf{y}} \in \mathbb{R}^M$ are uniformly distributed in the interval $[-1, 1]$.

As the optimal value of (4.1) is zero, we use $f_\mu(\mathbf{x}) \le \varepsilon$ as the termination criterion with $\varepsilon = 10^{-2}$. We also choose $\mu = \varepsilon$.

For each problem, ACDM, ACDM_M, AACDM and AACDM_M are run 10 times with the uniformly randomized initial point $\mathbf{x}^0 \in [-10, 10]^M$ and we report the average running time in Table 1.

Problem size		CPU time(s)			
N	M	ACDM	AACDM	ACDM_M	AACDM_M
50	100	3.2794	2.6762	2.6966	**2.0354**
100	50	3.1576	2.7000	2.0285	**1.2460**
100	200	11.9336	10.0678	9.3187	**7.2214**
200	100	10.8258	8.6158	6.0377	**3.5905**
200	400	39.2334	34.9947	30.5583	**24.1483**
400	200	42.5816	39.8447	19.2615	**10.6929**
400	800	139.1519	133.1698	108.7773	**90.9265**
800	400	204.8777	165.0772	65.0083	**33.9478**
800	1600	564.3874	513.7417	467.3034	**390.5520**
1600	800	886.2508	857.4967	246.4927	**121.4863**

Table 1: Average running time in seconds on problem (4.1)

4.2 $l_2 - l_1$ penalty functions

Considered the dual $l_2 - l_1$ penalty functions given in [2]:

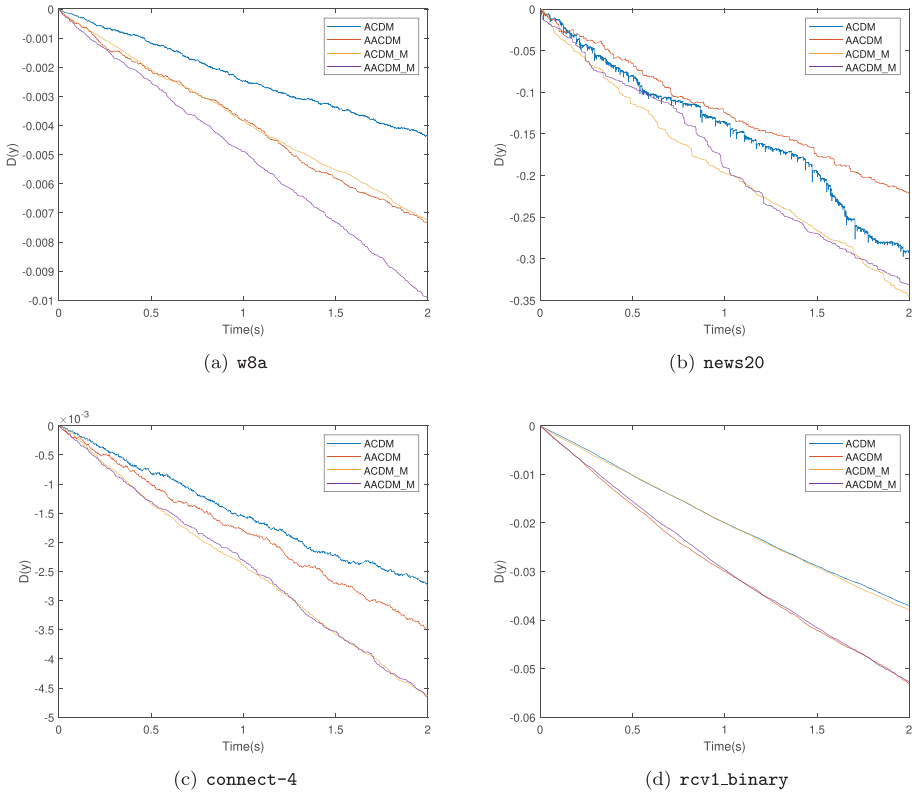
$$\min_{\mathbf{w}\in\mathbb{R}^M} P(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \phi_i(\langle \mathbf{a}^i, \mathbf{w}\rangle) + r(\mathbf{w}), \tag{4.3}$$

where $r(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2$ and $\phi_i(\alpha) = \frac{1}{2}(\alpha - l_i)^2 + |\alpha - l_i|$ with $\{(\mathbf{a}^1, l_1), \cdots, (\mathbf{a}^N, l_N)\}$ being a training set. The dual form of (4.3) is

$$\min_{\mathbf{y}\in\mathbb{R}^N} D(\mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N} \phi_i^*(\mathbf{y}_i) + \frac{1}{2\lambda N^2}\|\sum_{i=1}^{N} \mathbf{y}_i \mathbf{a}_i\|^2, \tag{4.4}$$

where the coordinate Lipshitz constants of D are $L_i = \frac{1}{N} + \frac{1}{\lambda N^2}\|\mathbf{a}^i\|^2$ for all i.

We apply ACDM, ACDM_M, AACDM and AACDM_M to the dual problem (4.4) on the datasets `w8a`, `news20`, `connect-4` and `rcv1_binary` from the LibSVM [7]. The numerical results are given in Figure 1.



(a) `w8a` (b) `news20`

(c) `connect-4` (d) `rcv1_binary`

Figure 1: Numerical results on $l_2 - l_1$ penalty functions

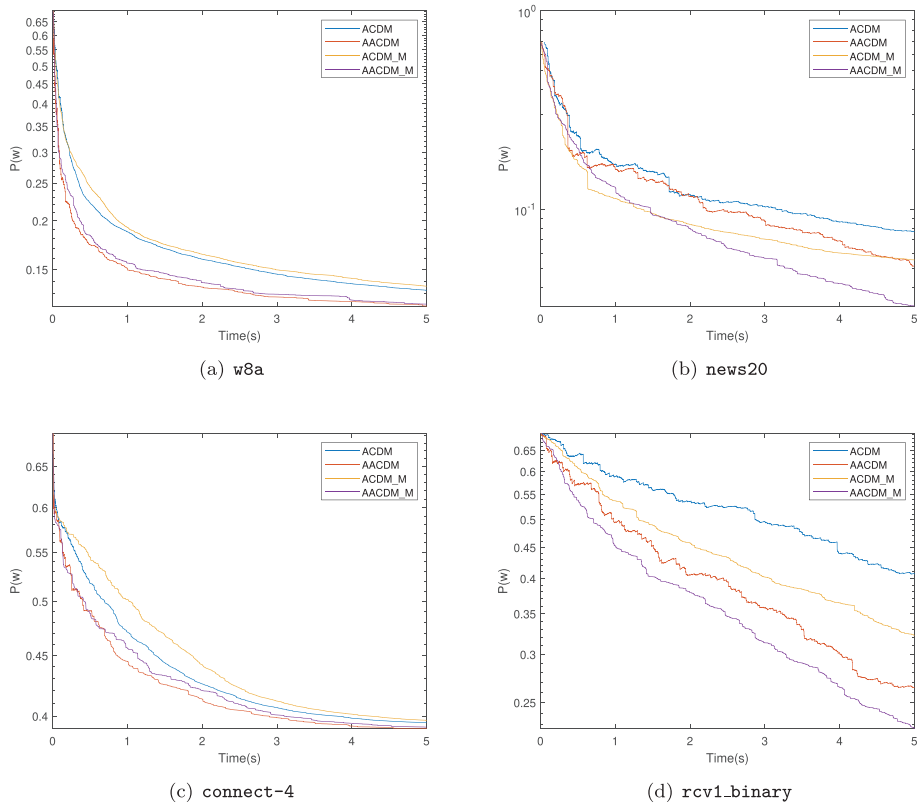
4.3 Logistic regression

Logistic regression is a widely used model for classification in machine learning. Consider the problem (4.3) with

$$\phi_i(\alpha) = \log(1 + \exp^{-\alpha l_i}) \quad \text{and} \quad r(\mathbf{w}) = 0. \tag{4.5}$$

We apply ACDM, ACDM_M, AACDM and AACDM_M to the original problem (4.3) on the datasets `w8a`, `news20`, `connect-4` and `rcv1_binary` too. The numerical results are given in Figure 2.

Note that in contrast to ACDM in [15], Algorithm 2.1 and Algorithm 2.4 spend more time in picking t_{k+1}, however with our adaptive choice of parameter t_{k+1}, the step sizes may be generally larger in outer loop. When the benefit of the latter can offset the expense caused by the former, the overall overhead will be reduced. The numerical results validate the improvement.



(a) `w8a` (b) `news20`

(c) `connect-4` (d) `rcv1_binary`

Figure 2: Numerical results on logistic regression

5 Discussions

We proposed two adaptive ACDMs for solving the problem (1.1) both in non-monotone version and in monotone version. Compared to the ACDMs, they can dynamically backtrack the coordinate Lipschitz constants of the problem. Convergence rate of the algorithms are given for σ-strongly convex function ($\sigma \geq 0$). Numerical results show that the adaptive strategy improve the performance a lot.

In [2], a counterpart of the ACDM was proposed for (1.1) when the objective function is non-strongly convex. Similarly, we can develop the adaptive ACDM for non-strongly convex problems using probabilities $L_i^{1/2}[\sum_{k=1}^n L_k^{1/2}]^{-1}$ for non-uniform sampling. It can be done as follows.

At the k-th iteration, first generate $i_k \in \{1, 2, \cdots, n\}$ via probability distribution $(p_i)_i$, choose $t_{k+1} \in \mathbb{N}$, and set

$$\tau_k = \frac{2}{1 + \sqrt{1 + 4\frac{\delta^{t_k - t_{k+1}}}{\tau_{k-1}^2}}} \in (0, 1]; \tag{5.1}$$

then repeat

$$\mathbf{x}^{k+1} = \tau_k \mathbf{z}^k + (1-\tau_k)\mathbf{y}^k, \tag{5.2}$$

$$\mathbf{y}^{k+1} = \mathbf{x}^{k+1} - \frac{1}{\delta^{-t_{k+1}} L_{i_k}} \nabla_{i_k} H(\mathbf{x}^{k+1}) \mathbf{e}_{i_k} \tag{5.3}$$

until

$$H(\mathbf{y}^{k+1}) \leq H(\mathbf{x}^{k+1}) - \frac{1}{\delta^{-t_{k+1}} 2L_{i_k}} (\nabla_{i_k} H(\mathbf{x}^{k+1}))^2 \tag{5.4}$$

is satisfied; finally, compute

$$\eta_{k+1} = \frac{1}{\delta^{-t_{k+1}} \tau_k S_{1/2}^2}, \tag{5.5}$$

$$\mathbf{z}^{k+1} = \mathbf{z}^k - \frac{\eta_{k+1}}{p_{i_k}} \nabla_{i_k} H(\mathbf{x}^{k+1}) \mathbf{e}_{i_k}. \tag{5.6}$$

It can be proved that for $k \geq 0$

$$\mathbb{E}_{\xi_k}[(H(\mathbf{y}^{k+1}) - H(\mathbf{x}^*))] \leq \frac{2S_{1/2}^2}{(k+2)^2} \|\mathbf{y}^0 - \mathbf{x}^*\|^2. \tag{5.7}$$

The monotone strategy similar to (2.2) can also be used to improve the performance for the non-strongly convex problems.

References

[1] A. Aberdam and A. Beck, An accelerated coordinate gradient descent algorithm for non-separable composite optimization, *J. Optim. Theory Appl.* 193 (2022), 219–246.

[2] Z.Y. Allen-Zhu, Z. Qu, P. Richtárik and Y. Yuan, Even faster accelerated coordinate descent using non-uniform sampling, *Proc. Int. Conf. Mach. Learn.*, (2016), 1110–1119.

[3] A. Beck, *First-Order Methods in Optimization*, SIAM, Philadelphia, 2017

[4] A. Beck, E. Pauwels and S. Sabach, The cyclic block conditional gradient method for convex optimization problems, *SIAM J. Optim.* 25 (2015), 2024–2049.

[5] A. Beck and M. Teboulle, Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems, *IEEE Trans. Image Process.* 18 (2009), 2419–2434.

[6] A. Beck and L. Tetruashvili, On the convergence of block coordinate descent type methods, *SIAM J. Optim.* 23 (2013), 2037–2060.

[7] C. Chang and C. Lin, *LIBSVM: A Library for Support Vector Machines*, Association for Computing Machinery, New York, 2011.

[8] A. d ' Aspremont, D. Scieur and A. Taylor, Acceleration methods, *Foundations and Trends® in Optimization* 5 (2021), 1–245.

[9] J. Friedman, T. Hastie, H. Höfling and R. Tibshirani, Pathwise coordinate optimization, *Ann. Appl. Stat.* 1 (2007), 302–332.

[10] S. Lacoste-Julien, M. Jaggi, M. Schmidt and P. Pletscher, Block-coordinate Frank-Wolfe optimization for structural SVMs, in: *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 53–61.

[11] Y.T. Lee and A. Sidford, Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems, in: *Proc. Annu. IEEE Symp. Found. Comput. Sci.*, 2013, pp. 147–156.

[12] Q.H. Lin, Z.S. Lu and L. Xiao, An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization, *SIAM J. Optim.* 25 (2015), 2244–2273.

[13] H.H. Lu, R. Freund and V. Mirrokni, Accelerating greedy coordinate descent methods, in: *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3257–3266.

[14] Y. Nesterov, Efficiency of coordinate descent methods on huge-scale optimization problems, *SIAM J. Optim.* 22 (2012), 341–362.

[15] Y. Nesterov and S. U. Stich, Efficiency of the accelerated coordinate descent method on structured optimization problems, *SIAM J. Optim.* 27 (2017), 110–123.

[16] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, SIAM, Philadelphia, 2000.

[17] S.U. Stich, A. Raj and M. Jaggi, Safe adaptive importance sampling, *Proc. Adv. Neural Inf. Process. Syst.* 30 (2017), 4382–4392.

[18] S.J. Wright Coordinate descent algorithms, *Math. Program.* 151 (2015), 3–34.

Yu You
School of Mathematical Sciences, Shanghai Jiao Tong University
Shanghai, China
Email: youyu0828sjtu@163.com

Jirui Ma
Beijing International Center for Mathematical Research, Shanghai Peking University
Beijing 100871, P.R. China.
Email: majirui@bicmr.pku.edu.cn

Manuscript received 2 December 2022
revised 21 November 2023
accepted for publication 26 February 2024

Yu You
School of Mathematical Sciences
Shanghai Jiao Tong University
Shanghai, China
E-mail address: youyu0828sjtu@163.com

Jirui Ma
Beijing International Center for Mathematical Research
Shanghai Peking University
Beijing 100871, P.R. China
E-mail address: majirui@bicmr.pku.edu.cn