Accelerating greedy coordinate descent methods

Accelerating Greedy Coordinate Descent Methods

Haihao Lu* Robert M. Freund† Vahab Mirrokni‡

Revised June 6, 2018 (original dated Feb. 9, 2018)

Abstract

We study ways to accelerate greedy coordinate descent in theory and in practice, where "accelerate" refers either to $O(1/k^2)$ convergence in theory, in practice, or both. We introduce and study two algorithms: Accelerated Semi-Greedy Coordinate Descent (ASCD) and Accelerated Greedy Coordinate Descent (AGCD). While ASCD takes greedy steps in the x-updates and randomized steps in the z-updates, AGCD is a straightforward extension of standard greedy coordinate descent that only takes greedy steps. On the theory side, our main results are for ASCD: we show that ASCD achieves $O(1/k^2)$ convergence, and it also achieves accelerated linear convergence for strongly convex functions. On the empirical side, we observe that both AGCD and ASCD outperform Accelerated Randomized Coordinate Descent on a variety of instances. In particular, we note that AGCD significantly outperforms the other accelerated coordinate descent methods in numerical tests, in spite of a lack of theoretical guarantees for this method. To complement the empirical study of AGCD, we present a Lyapunov energy function argument that points to an explanation for why a direct extension of the acceleration proof for AGCD does not work; and we also introduce a technical condition under which AGCD is guaranteed to have accelerated convergence. Last of all, we confirm that this technical condition holds in our empirical study.

1 Introduction: Related Work and Accelerated Coordinate Descent Framework

Coordinate descent methods have received much-deserved attention recently due to their capability for solving large-scale optimization problems (with sparsity) that arise in machine learning applications and elsewhere. With inexpensive updates at each iteration, coordinate descent algorithms obtain faster running times than similar full gradient descent algorithms in order to reach the same near-optimality tolerance; indeed some of these algorithms now comprise the state-of-the-art in machine learning algorithms for loss minimization.

[*] MIT Department of Mathematics and MIT Operations Research Center, 77 Massachusetts Avenue, Cambridge, MA 02139 (mailto: haihao@mit.edu). This author's research is supported by AFOSR Grant No. FA9550-15-1-0276 and the MIT-Belgium Université Catholique de Louvain Fund.

[†] MIT Sloan School of Management, 77 Massachusetts Avenue, Cambridge, MA 02139 (mailto: rfreund@mit.edu). This author's research is supported by AFOSR Grant No. FA9550-15-1-0276 and the MIT-Belgium Université Catholique de Louvain Fund.

[‡] Google Research, 111 8th Avenue New York, NY 10011 (mailto: mirrokni@google.com).

Most recent research on coordinate descent has focused on versions of randomized coordinate descent, which can essentially recover the same results (in expectation) as full gradient descent, including obtaining "accelerated" (i.e., $O(1/k^2)$) convergence guarantees. On the other hand, in some important machine learning applications, greedy coordinate methods demonstrate superior numerical performance while also delivering much sparser solutions. For example, greedy coordinate descent is one of the fastest algorithms for the graphical LASSO implemented in DP-GLASSO [19]. And sequence minimization optimization (SMO) (a variant of greedy coordinate descent) is widely known as the best solver for kernel SVM [11][24] and is implemented in LIBSVM and SVMLight.

In general, for smooth convex optimization the standard first-order methods converge at a rate of $O(1/k)$ (including greedy coordinate descent). In 1983, Nesterov [22] proposed an algorithm that achieved a rate of $O(1/k^2)$ – which can be shown to be the optimal rate achievable by any first-order method [20]. This method (and other similar methods) is now referred to as Accelerated Gradient Descent (AGD).

However, there has not been much work on accelerating the standard Greedy Coordinate Descent (GCD) due to the inherent difficulty in demonstrating $O(1/k^2)$ computational guarantees (we discuss this difficulty further in Section 4.1). One work that might be close is [26], which updates the z-sequence using the full gradient and thus should not be considered as a coordinate descent method in the standard sense. There is a very related concurrent work [15] and we will discuss the connections to our results in Section 4.1.

In this paper, we study ways to accelerate greedy coordinate descent in theory and in practice. We introduce and study two algorithms: Accelerated Semi-Greedy Coordinate Descent (ASCD) and Accelerated Greedy Coordinate Descent (AGCD). While ASCD takes greedy steps in the x-updates and randomized steps in the z-updates, AGCD is a straightforward extension of GCD that only takes greedy steps. On the theory side, our main results are for ASCD: we show that ASCD achieves $O(1/k^2)$ convergence, and it also achieves accelerated linear convergence when the objective function is furthermore strongly convex. However, a direct extension of convergence proofs for ARCD does not work for ASCD due to the different coordinates we use to update x-sequence and z-sequence. Thus, we present a new proof technique – which shows that a greedy coordinate step yields better objective function value than a full gradient step with a modified smoothness condition.

On the empirical side, we first note that in most of our experiments ASCD outperforms Accelerated Randomized Coordinate Descent (ARCD) in terms of running time. On the other hand, we note that AGCD significantly outperforms the other accelerated coordinate descent methods in all instances, in spite of a lack of theoretical guarantees for this method. To complement the empirical study of AGCD, we present a Lyapunov energy function argument that points to an explanation for why a direct extension of the proof for AGCD does not work. This argument inspires us to introduce a technical condition under which AGCD is guaranteed to converge at an accelerated rate. Interestingly, we confirm that technical condition holds in a variety of instances in our empirical study, which in turn justifies our empirical observation that AGCD works very well in practice.

1.1 Related Work

Coordinate Descent. Coordinate descent methods have a long history in optimization, and convergence of these methods has been extensively studied in the optimization community in the 1980s-90s, see [4], [17], and [18]. There are roughly three types of coordinate descent methods depending on how the coordinate is chosen: randomized coordinate descent (RCD), cyclic coordinate descent (CCD), and greedy coordinate descent (GCD). RCD has received much attention since the seminal paper of Nesterov [21]. In RCD, the coordinate is chosen randomly from a certain fixed distribution. [25] provides an excellent review of theoretical results for RCD. CCD chooses the coordinate in a cyclic order, see [3] for basic convergence results. More recent results show that CCD is inferior to RCD in the worst case [28], while it is better than RCD in certain situations [9]. In GCD, we select the coordinate yielding the largest reduction in the objective function value. GCD usually delivers better function values at each iteration in practice, though this comes at the expense of having to compute the full gradient in order to select the gradient coordinate with largest magnitude. The recent work [23] shows that GCD has faster convergence than RCD in theory, and also provides several applications in machine learning where the full gradient can be computed cheaply. A parallel GCD method is proposed in [31] and numerical results show its advantage in practice.

Accelerated Randomized Coordinate Descent. Since Nesterov's paper on RCD [21] there has been significant focus on accelerated versions of RCD. In particular, [21] developed the first accelerated randomized coordinate gradient method for minimizing unconstrained smooth functions. [16] present a sharper convergence analysis of Nesterov's method using a randomized estimate sequence framework. [7] proposed the APPROX (Accelerated, Parallel and PROXimal) coordinate descent method and obtained an accelerated sublinear convergence rate, and [12] developed an efficient implementation of ARCD.

1.2 Accelerated Coordinate Descent Framework

Our optimization problem of interest is:

$$P : f^* := \quad \text{minimum}_x \quad f(x) \ , \tag{1}$$

where $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is a differentiable convex function.

Definition 1.1. *$f(\cdot)$ is coordinate-wise L-smooth for the vector of parameters $L := (L_1, L_2, \ldots, L_n)$ if $\nabla f(\cdot)$ is coordinate-wise Lipschitz continuous for the corresponding coefficients of L, i.e., for all $x \in \mathbb{R}^n$ and $h \in \mathbb{R}$ it holds that:*

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \le L_i |h| \ , \ i = 1, \ldots, n \ , \tag{2}$$

where $\nabla_i f(\cdot)$ denotes the i^{th} coordinate of $\nabla f(\cdot)$ and e_i is i^{th} unit coordinate vector, for $i = 1, \ldots, n$.

We presume throughout that $L_i > 0$ for $i = 1, \ldots, n$. Let $\mathbf{L}$ denote the diagonal matrix whose diagonal coefficients correspond to the respective coefficients of L. Let $\langle \cdot, \cdot \rangle$ denote the standard coordinate inner product in $\mathbb{R}^n$, namely $\langle x, y \rangle = \sum_{i=1}^n x_i y_i$, and let $\| \cdot \|_p$ denote the ℓ_p norm for $1 \le p \le \infty$. Let $\langle x, y \rangle_L := \sum_{i=1}^n L_i x_i y_i = \langle x, \mathbf{L}y \rangle = \langle \mathbf{L}x, y \rangle$ denote the L-inner product. Define the

Algorithm 1 Accelerated Coordinate Descent Framework without Strong Convexity

```
Initialize. Initialize with $x^0$, set $z^0 \leftarrow x^0$. Assume $f(\cdot)$ is coordinate-wise $L$-smooth for known and given $L$. Define the sequence $\{\theta_k\}$ as follows: $\theta_0 = 1$, and define $\theta_k$ recursively by the relationship $\frac{1-\theta_k}{\theta_k^2} = \frac{1}{\theta_{k-1}^2}$ for $k = 1, 2, \ldots$.

At iteration $k$ :
Perform Updates.
Define $y^k := (1-\theta_k)x^k + \theta_k z^k$
Choose coordinate $j_k^1$ (by some rule)
Compute $x^{k+1} := y^k - \frac{1}{L_{j_k^1}} \nabla_{j_k^1} f(y^k) e_{j_k^1}$
Choose coordinate $j_k^2$ (by some rule)
Compute $z^{k+1} := z^k - \frac{1}{nL_{j_k^2}\theta_k} \nabla_{j_k^2} f(y^k) e_{j_k^2}$ .
```

norm $\|x\|_L := \sqrt{\langle x, \mathbf{L}x \rangle}$. Letting $\mathbf{L}^{-1}$ denote the inverse of $\mathbf{L}$, we will also use the norm $\|\cdot\|_{L^{-1}}$ defined by $\|v\|_{L^{-1}} := \sqrt{\langle v, \mathbf{L}^{-1}v \rangle} = \sqrt{\sum_{i=1}^n \mathbf{L}_i^{-1} v_i^2}$.

Algorithm 1 presents a generic framework for accelerated coordinate descent methods that is flexible enough to encompass deterministic as well as randomized methods. One specific case is the standard Accelerated Randomized Coordinate Descent (ARCD). In this paper we propose and study two other cases. The first is Accelerated Greedy Coordinate Descent (AGCD), which is a straightforward extension of greedy coordinate descent to the acceleration framework and which, surprisingly, has not been previously studied (that we are aware of). The second is a new algorithm which we call Accelerated Semi-Greedy Coordinate Descent (ASCD) that takes greedy steps in the x-updates and randomized steps in the z-update.

In the framework of Algorithm 1 we choose a coordinate j_k^1 of the gradient $\nabla f(y^k)$ to perform the update of the x-sequence, and we choose (a possibly different) coordinate j_k^2 of the gradient $\nabla f(y^k)$ to perform the update of the z-sequence. Herein we will study three different rules for choosing the coordinates j_k^1 and j_k^2 which then define three different specific algorithms:

- ARCD (Accelerated Randomized Coordinate Descent): use the rule

$$j_k^2 = j_k^1 :\sim \mathcal{U}[1, \cdots, n] \tag{3}$$

- AGCD (Accelerated Greedy Coordinate Descent): use the rule

$$j_k^2 = j_k^1 := \arg\max_i \frac{1}{\sqrt{L_i}} |\nabla_i f(y^k)| \tag{4}$$

- ASCD (Accelerated Semi-Greedy Coordinate Descent): use the rule

$$\begin{aligned} j_k^1 &:= \arg\max_i \frac{1}{\sqrt{L_i}} |\nabla_i f(y^k)| \\ j_k^2 &:\sim \mathcal{U}[1, \cdots, n] \ . \end{aligned} \tag{5}$$

In ARCD a random coordinate j_k^1 is chosen at each iteration k, and this coordinate is used to update both the x-sequence and the z-sequence. ARCD is well studied, and is known to have the following convergence guarantee in expectation (see [7] for details):

$$E\left[f(x^k) - f(x^*)\right] \le \tfrac{2n^2}{(k+1)^2}\|x^* - x^0\|_L^2 \ , \tag{6}$$

where the expectation is on the random variables used to define the first k iterations.

In AGCD we choose the coordinate j_k^1 in a "greedy" fashion, i.e., corresponding to the maximal (weighted) absolute value coordinate of the the gradient $\nabla f(y^k)$. This greedy coordinate is used to update both the x-sequence and the z-sequence. As far as we know AGCD has not appeared in the first-order method literature. One reason for this is that while AGCD is the natural accelerated version of greedy coordinate descent, the standard proof methodologies for establishing acceleration guarantees (i.e., $O(1/k^2)$ convergence) fail for AGCD. Despite this lack of worst-case guarantee, we show in Section 5 that AGCD is extremely effective in numerical experiments on synthetic linear regression problems as well as on practical logistic regression problems, and dominates other coordinate descent methods in terms of numerical performance. Furthermore, we observe that AGCD attains $O(1/k^2)$ convergence (or better) on these problems in practice. Thus AGCD is worthy of significant further study, both computationally as well as theoretically. Indeed, in Section 4 we will present a technical condition that implies $O(1/k^2)$ convergence when satisfied, and we will argue that this condition ought to be satisfied in many settings.

ASCD, which we consider to be the new theoretical contribution of this paper, combines the salient features of AGCD and ARCD. In ASCD we choose the greedy coordinate of the gradient to perform the x-update, while we choose a random coordinate to perform the z-update. In this way we achieve the practical advantage of greedy x-updates, while still guaranteeing $O(1/k^2)$ convergence in expectation by virtue of choosing the random coordinate used in the z-update, see Theorem 2.1. And under strong convexity, ASCD achieves linear convergence as will be shown in Section 3.

The paper is organized as follows. In Section 2 we present the $O(1/k^2)$ convergence guarantee (in expectation) for ASCD. In Section 3 we present an extension of the accelerated coordinate descent framework to the case of strongly convex functions, and we present the associated linear convergence guarantee for ASCD under strong convexity. In Section 4 we study AGCD; we present a Lyapunov energy function argument that points to why standard analysis of accelerated gradient descent methods fails in the analysis of AGCD. In Section 4.2 we present a technical condition under which AGCD will achieve $O(1/k^2)$ convergence. In Section 5, we present results of our numerical experiments using AGCD and ASCD on synthetic linear regression problems as well as practical logistic regression problems.

2 Accelerated Semi-Greedy Coordinate Descent (ASCD)

In this section we present our computational guarantee for the Accelerated Semi-Greedy Coordinate Descent (ASCD) method in the non-strongly convex case. Algorithm 1 with rule (5) presents the Accelerated Semi-Greedy Coordinate Descent method (ASCD) for the non-strongly convex case. At each iteration k the ASCD method choose the greedy coordinate j_k^1 to do the x-update, and chooses a randomized coordinate $j_k^2 \sim \mathcal{U}[1, \cdots, n]$ to do the z-update. Unlike ARCD where the same randomized coordinate is used in both the x-update and the z-update – in ASCD j_k^1 is chosen in a deterministic greedy way, j_k^1 and j_k^2 are likely to be different.

At each iteration k of ASCD the random variable j_k^2 is introduced, and therefore x^k depends on the realization of the random variable

$$\xi_k := \{j_0^2, \ldots, j_{k-1}^2\} \ .$$

For convenience we also define $\xi_0 := \emptyset$.

The following theorem presents our computational guarantee for ASCD for the non-strongly convex case:

Theorem 2.1. *Consider the Accelerated Semi-Greedy Coordinate Descent method (Algorithm 1 with rule* (5)*). If $f(\cdot)$ is coordinate-wise L-smooth, it holds for all $k \geq 1$ that:*

$$E_{\xi_k}\left[f(x^k) - f(x^*)\right] \leq \tfrac{n^2\theta_{k-1}^2}{2}\|x^* - x^0\|_L^2 \leq \tfrac{2n^2}{(k+1)^2}\|x^* - x^0\|_L^2 \ . \tag{7}$$

□

In the interest of both clarity and a desire to convey some intuition on proofs of accelerated methods in general, we will present the proof of Theorem 2.1 after first establishing some intermediary results along with some explanatory comments. We start with the "Three-Point Property" of Tseng [29]. Given a differentiable convex function $h(\cdot)$, the Bregman distance for $h(\cdot)$ is $D_h(y,x) := h(y) - h(x) - \langle \nabla h(x), y - x\rangle$. The Three-Point property can be stated as follows:

Lemma 2.1. (Three-Point Property (Tseng [29])) *Let $\phi(\cdot)$ be a convex function, and let $D_h(\cdot,\cdot)$ be the Bregman distance for $h(\cdot)$. For a given vector z, let*

$$z^+ := \arg\min_{x\in\mathbb{R}^n} \{\phi(x) + D_h(x,z)\} \ .$$

Then

$$\phi(x) + D_h(x,z) \geq \phi(z^+) + D_h(z^+,z) + D_h(x,z^+) \quad \text{for all } \ x \in \mathbb{R}^n \ ,$$

with equality holding in the case when $\phi(\cdot)$ is a linear function and $h(\cdot)$ is a quadratic function. □

Also, it follows from elementary integration and the coordinate-wise Lipschitz condition (2) that

$$f(x + he_i) \leq f(x) + h \cdot \nabla_i f(x) + \tfrac{h^2 L_i}{2} \ \text{ for all } x \in \mathbb{R}^n \text{ and } h \in \mathbb{R} \ . \tag{8}$$

At each iteration $k = 0, 1, \ldots$ of ASCD, notice that x^{k+1} is one step of greedy coordinate descent from y^k in the norm $\|\cdot\|_L$. Now define $s^{k+1} := y^k - \frac{1}{n}\mathbf{L}^{-1}\nabla f(y^k)$, which is a full steepest-descent step from y^k in the norm $\|\cdot\|_{nL}$. We first show that the greedy coordinate descent step yields a good objective function value as compared to the quadratic model that yields s^{k+1}.

Lemma 2.2.

$$f(x^{k+1}) \leq f(y^k) + \langle \nabla f(y^k), s^{k+1} - y^k\rangle + \tfrac{n}{2}\|s^{k+1} - y^k\|_L^2 \ .$$

Proof:

$$\begin{aligned} f(x^{k+1}) &\leq f(y^k) - \tfrac{1}{2L_{j_k^1}} \left(\nabla_{j_k^1} f(y^k)\right)^2 \\ &\leq f(y^k) - \tfrac{1}{2n}\|\nabla f(y^k)\|^2_{L^{-1}} \\ &= f(y^k) + \langle \nabla f(y^k), s^{k+1} - y^k\rangle + \tfrac{n}{2}\|s^{k+1} - y^k\|^2_L \ , \end{aligned} \tag{9}$$

where the first inequality of (9) derives from the smoothness of $f(\cdot)$, and is a simple instance of (8) using $x = y^k$, $i = j_k^1$, and $h = -\frac{1}{L_{j_k^1}}\nabla_{j_k^1} f(y^k)$. The second inequality of (9) follows from the definition of j_k^1 which yields:

$$n\left[\tfrac{1}{L_{j_k^1}}\left(\nabla_{j_k^1} f(y^k)\right)^2\right] \geq \sum_{i=1}^{n} \tfrac{1}{L_i}\left(\nabla_i f(y^k)\right)^2 = \left\|\nabla f(y^k)\right\|^2_{L^{-1}} \ .$$

The last equality of (9) follows by using the definition of s^{k+1} and rearranging terms. □

Utilizing the interpretation of s^{k+1} as a gradient descent step from y^k but with a larger smoothness descriptor (nL as opposed to L), we can invoke the standard proof for accelerated gradient descent derived in [29] for example. We define $t^{k+1} := z^k - \frac{1}{n\theta_k}\mathbf{L}^{-1}\nabla f(y^k)$, or equivalently we can define t^{k+1} by:

$$t^{k+1} = \arg\min_z \ \langle \nabla f(y^k), z - z^k\rangle + \tfrac{n\theta_k}{2}\|z - z^k\|^2_L \tag{10}$$

(which corresponds to z^{k+1} in [29] for standard accelerated gradient descent). Then we have:

Lemma 2.3.

$$f(x^{k+1}) \leq (1-\theta_k) f(x^k) + \theta_k f(x^*) + \tfrac{n\theta_k^2}{2}\|x^* - z^k\|^2_L - \tfrac{n\theta_k^2}{2}\|x^* - t^{k+1}\|^2_L \ . \tag{11}$$

Proof. Following from Lemma 2.2, we have

$$\begin{aligned} f(x^{k+1}) &\leq f(y^k) + \langle \nabla f(y^k), s^{k+1} - y^k\rangle + \tfrac{n}{2}\|s^{k+1} - y^k\|^2_L \\ &= f(y^k) + \theta_k\left(\langle \nabla f(y^k), t^{k+1} - z^k\rangle + \tfrac{n\theta_k}{2}\|t^{k+1} - z^k\|^2_L\right) \\ &= f(y^k) + \theta_k\left(\langle \nabla f(y^k), x^* - z^k\rangle + \tfrac{n\theta_k}{2}\|x^* - z^k\|^2_L - \tfrac{n\theta_k}{2}\|x^* - t^{k+1}\|^2_L\right) \\ &= (1-\theta_k)\left(f(y^k) + \langle \nabla f(y^k), x^k - y^k\rangle\right) + \theta_k\left(f(y^k) + \langle \nabla f(y^k), x^* - y^k\rangle\right) \\ &\quad + \tfrac{n\theta_k^2}{2}\|x^* - z^k\|^2_L - \tfrac{n\theta_k^2}{2}\|x^* - t^{k+1}\|^2_L \\ &\leq (1-\theta_k) f(x^k) + \theta_k f(x^*) + \tfrac{n\theta_k^2}{2}\|x^* - z^k\|^2_L - \tfrac{n\theta_k^2}{2}\|x^* - t^{k+1}\|^2_L \end{aligned} \tag{12}$$

where the first equality of (12) utilizes $s^{k+1} - y^k = \theta_k(t^{k+1} - z^k)$. The second equality of (12) follows as an application of the Three-Point-Property (Lemma 2.1) together with (10), where we set

$\phi(x) = \langle \nabla f(y^k), x - z^k \rangle$ and $h(x) = \frac{n\theta_k}{2}\|x\|_L^2$ (whereby $D_h(x,v) = \frac{n\theta_k}{2}\|x - v\|_L^2$). The third equality of (12) is derived from $y^k = (1-\theta_k)x^k + \theta_k z^k$ and rearranging the terms. And the last inequality of (12) is an application of the gradient inequality at y^k applied to x^k and also to x^*. □

Notice that t^{k+1} is an all-coordinate update of z^k, and computing t^{k+1} can be very expensive. Instead we will use z^{k+1} to replace t^{k+1} in (11) by using the equality in the next lemma.

Lemma 2.4.

$$\tfrac{n}{2}\|x^* - z^k\|_L^2 - \tfrac{n}{2}\|x^* - t^{k+1}\|_L^2 = \tfrac{n^2}{2}\|x^* - z^k\|_L^2 - \tfrac{n^2}{2}E_{j_k^2}\left[\|x^* - z^{k+1}\|_L^2\right] \ . \tag{13}$$

Proof:

$$\begin{aligned} \tfrac{n}{2}\|x^* - z^k\|_L^2 - \tfrac{n}{2}\|x^* - t^{k+1}\|_L^2 &= \tfrac{n}{2}\left\langle t^{k+1} - z^k, 2x^* - 2z^k \right\rangle_L - \tfrac{n}{2}\left\|t^{k+1} - z^k\right\|_L^2 \\ &= \tfrac{n^2}{2}E_{j_k^2}\left[\left\langle z^{k+1} - z^k, 2x^* - 2z^k \right\rangle_L - \left\|z^{k+1} - z^k\right\|_L^2\right] \qquad (14) \\ &= \tfrac{n^2}{2}\|x^* - z^k\|_L^2 - \tfrac{n^2}{2}E_{j_k^2}\left[\|x^* - z^{k+1}\|_L^2\right], \end{aligned}$$

where the first and third equations above are straightforward arithmetic rearrangements, and the second equation follows from the two easy-to-verify identities $t^{k+1} - z^k = nE_{j_k^2}\left[z^{k+1} - z^k\right]$ and $\left\|t^{k+1} - z^k\right\|_L^2 = nE_{j_k^2}\left[\left\|z^{k+1} - z^k\right\|_L^2\right]$. □

We now have all of the ingredients needed to prove Theorem 2.1.

Proof of Theorem 2.1 Substituting (13) into (11), we obtain:

$$f(x^{k+1}) \le (1-\theta_k)f(x^k) + \theta_k f(x^*) + \tfrac{n^2\theta_k^2}{2}\|x^* - z^k\|_L^2 - \tfrac{n^2\theta_k^2}{2}E_{j_k^2}\left[\|x^* - z^{k+1}\|_L^2\right] \ . \tag{15}$$

Rearranging and substituting $\frac{1-\theta_{k+1}}{\theta_{k+1}^2} = \frac{1}{\theta_k^2}$, we arrive at:

$$\tfrac{1-\theta_{k+1}}{\theta_{k+1}^2}\left(f(x^{k+1}) - f(x^*)\right) + \tfrac{n^2}{2}E_{j_k^2}\left\|x^* - z^{k+1}\right\|_L^2 \le \left[\tfrac{1-\theta_k}{\theta_k^2}\left(f(x^k) - f(x^*)\right) + \tfrac{n^2}{2}\left\|x^* - z^k\right\|_L^2\right] \ .$$

Taking the expectation over the random variables $j_1^2, j_2^2, \ldots, j_k^2$, it follows that:

$$E_{\xi_{k+1}}\left[\tfrac{1-\theta_{k+1}}{\theta_{k+1}^2}\left(f(x^{k+1}) - f(x^*)\right) + \tfrac{n^2}{2}\left\|x^* - z^{k+1}\right\|_L^2\right] \le E_{\xi_k}\left[\tfrac{1-\theta_k}{\theta_k^2}\left(f(x^k) - f(x^*)\right) + \tfrac{n^2}{2}\left\|x^* - z^k\right\|_L^2\right] \ .$$

Applying the above inequality in a telescoping manner for $k = 1, 2, \ldots$, yields:

$$\begin{aligned} E_{\xi_k}\left[\tfrac{1-\theta_k}{\theta_k^2}\left(f(x^k) - f(x^*)\right)\right] &\le E_{\xi_k}\left[\tfrac{1-\theta_k}{\theta_k^2}\left(f(x^k) - f(x^*)\right) + \tfrac{n^2}{2}\left\|x^* - z^k\right\|_L^2\right] \\ &\vdots \\ &\le E_{\xi_0}\left[\tfrac{1-\theta_0}{\theta_0^2}\left(f(x^0) - f(x^*)\right) + \tfrac{n^2}{2}\left\|x^* - z^0\right\|_L^2\right] \\ &= \tfrac{n^2}{2}\left\|x^* - x^0\right\|_L^2 \ . \end{aligned}$$

Note from an induction argument that $\theta_i \le \frac{2}{i+2}$ for all $= 0, 1, \ldots$, whereby the above inequality rearranges to:

$$E_{\xi_k}\left[\left(f(x^k) - f(x^*)\right)\right] \le \tfrac{\theta_k^2}{1-\theta_k}\tfrac{n^2}{2}\left\|x^* - x^0\right\|_L^2 = \tfrac{n^2\theta_{k-1}^2}{2}\left\|x^* - x^0\right\|_L^2 \le \tfrac{2n^2}{(k+1)^2}\left\|x^* - x^0\right\|_L^2 \ . \quad □$$

3 Accelerated Coordinate Descent Framework under Strong Convexity

We begin with the definition of strong convexity as developed in [16]:

Definition 3.1. *$f(\cdot)$ is μ-strongly convex with respect to $\|\cdot\|_L$ if for all $x, y \in \mathbb{R}^n$ it holds that:*

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

Note that μ can be viewed as an extension of the condition number of $f(\cdot)$ in the traditional sense since μ is defined relative to the coordinate smoothness coefficients through $\|\cdot\|_L$, see [16]. Algorithm 2 presents the generic framework for accelerated coordinate descent methods in the case when $f(\cdot)$ is μ-strongly convex for known μ.

Algorithm 2 Accelerated Coordinate Descent Framework (μ-strongly convex case)

```
  Initialize. Initialize with $z^0 = x^0$. Assume $f(\cdot)$ is coordinate-wise $L$-smooth and $\mu$-strongly convex for known and given $L$ and $\mu$, and define the parameters $a = \frac{\sqrt{\mu}}{n+\sqrt{\mu}}$ and $b = \frac{\mu a}{n^2}$ .

  At iteration $k$ :
  Perform Updates.
  Define $y^k = (1-a)x^k + az^k$
  Choose $j_k^1$ (by some rule)
  Compute $x^{k+1} = y^k - \frac{1}{L_{j_k^1}} \nabla_{j_k^1} f(y^k) e_{j_k^1}$
  Compute $u^k = \frac{a^2}{a^2+b} z^k + \frac{b}{a^2+b} y^k$
  Choose $j_k^2$ (by some rule)
  Compute $z^{k+1} = u^k - \frac{a}{a^2+b} \frac{1}{nL_{j_k^2}} \nabla f_{j_k^2}(y^k) e_{j_k^2}$
```

Just as in the non-strongly convex case, we extend the three algorithms ARCD, AGCD, and ASCD to the strongly convex case by using the rules (3), (4), and (5) in Algorithm 2. The following theorem presents our computational guarantee for ASCD for strongly convex case:

Theorem 3.1. *Consider the Accelerated Semi-Greedy Coordinate Descent method for strongly convex case (Algorithm 2 with rule (5)). If $f(\cdot)$ is coordinate-wise L-smooth and μ-strongly convex with respect to $\|\cdot\|_L$, it holds for all $k \geq 1$ that:*

$$E_{\xi_k}\left[f(x^k) - f^* + \tfrac{n^2}{2}(a^2+b)\|z^k - x^*\|_L^2\right] \leq \left(1 - \tfrac{\sqrt{\mu}}{n+\sqrt{\mu}}\right)^k \left(f(x^0) - f^* + \tfrac{n^2}{2}(a^2+b)\|x^0 - x^*\|_L^2\right) . \tag{16}$$

□

We provide a concise proof of Theorem 3.1 in the Appendix.

4 Accelerated Greedy Coordinate Descent

In this section we discuss accelerated greedy coordinate descent (AGCD), which is Algorithm 1 with rule (4). In the interest of clarity we limit our discussion to the non-strongly convex case. We present a Lyapunov function argument which shows why the standard type of proof of accelerated gradient methods fails for AGCD, and we propose a technical condition under which AGCD is guaranteed to have an $O(1/k^2)$ accelerated convergence rate. Although there are no guarantees that the technical condition will hold for a given function $f(\cdot)$, we provide intuition as to why the technical condition ought to hold in most cases.

4.1 Why AGCD fails (in theory)

The mainstream research community's interest in Nesterov's accelerated method [22] started around 15 years ago; and yet even today most researchers struggle to find basic intuition as to what is really going on in accelerated methods. Indeed, Nesterov's estimation sequence proof technique seems to work out arithmetically but with little fundamental intuition. There are many recent work trying to explain this acceleration phenomenon [27][30][10][13][8][1] [5]. A line of recent work has attempted to give a physical explanation of acceleration techniques by studying the continuous-time interpretation of accelerated gradient descent via dynamical systems, see [27], [30], and [10]. In particular, [27] introduced the continuous-time dynamical system model for accelerated gradient descent, and presented a convergence analysis using a Lyapunov energy function in the continuous-time setting. [30] studied discretizations of the continuous-time dynamical system, and also showed that Nesterov's estimation sequence analysis is equivalent to the Lyapunov energy function analysis in the dynamical system in the discrete-time setting. And [10] presented an energy dissipation argument from control theory for understanding accelerated gradient descent.

In the discrete-time setting, one can construct a Lyapunov energy function of the form [30]:

$$E_k = A_k(f(x^k) - f^*) + \tfrac{1}{2}\|x^* - z^k\|_L^2 \ , \tag{17}$$

where A_k is a parameter sequence with $A_k \sim O(k^2)$, and one shows that E_k is non-increasing in k, thereby yielding:

$$f(x^k) - f^* \leq \frac{E_k}{A_k} \leq \frac{E_0}{A_k} \sim O\left(\frac{1}{k^2}\right) \ .$$

The proof techniques of acceleration methods such as [22], [29] and [1], as well as the recent proof techniques for accelerated randomized coordinate descent (such as [21], [16], and [7]) can all be rewritten in the above form (up to expectation) each with slightly different parameter sequences $\{A_k\}$.

Now let us return to accelerated greedy coordinate descent. Let us assume for simplicity that $L_1 = \cdots = L_n$ (as we can always do rescaling to achieve this condition). Then the greedy coordinate j_k^1 is chosen as the coordinate of the gradient with the largest magnitude, which corresponds to the coordinate yielding the greatest guaranteed decrease in the objective function value. However, in the proof of acceleration using the Lyapunov energy function, one needs to prove a decrease in E_k (17) instead of a decrease in the objective function value $f(x^k)$. The coordinate j_k^1 is not necessarily the greedy coordinate for decreasing the energy function E_k due to the presence of the second term $\|x^* - z^k\|_L^2$ in (17). This explains why the greedy coordinate can fail to decrease E_k, at least in theory. And because x^* is not known when running AGCD, there does not seem to be any way to find the greedy descent coordinate for the energy function E_k.

That is why in ASCD we use the greedy coordinate to perform the x-update (which corresponds to the fastest coordinate-wise decrease for the first term in energy function), while we choose a random coordinate to perform the z-update (which corresponds to the second term in the energy function); thereby mitigating the above problem in the case of ASCD.

In a concurrent paper [15], the authors develop computational theory for matching pursuit algorithms, which can be viewed as a generalized version of greedy coordinate descent where the directions do not need to form an orthogonal basis. The paper also develops an accelerated version of the matching pursuit algorithms, which turns out to be equivalent to the algorithm ASCD discussed here in the special case where the chosen directions are orthogonal. Although the focus in [15] and in our paper are different – [15] is more focused on (accelerated) greedy direction updates along a certain linear subspace whereas our focus is on when and how one can accelerate greedy coordinate updates – both of the works share a similar spirit and similar approaches in developing accelerated greedy methods. Moreover, both works use a decoupling of the coordinate update for the $\{x^k\}$ sequence (with a greedy rule) and the $\{z^k\}$ sequence (with a randomized rule). In fact, [15] is consistent with the argument in our paper as to why one cannot accelerate greedy coordinate descent in general.

4.2 How to make AGCD work (in theory)

Here we propose the following technical condition under which the proof of acceleration of AGCD can be made to work.

Technical Condition 4.1. *There exists a positive constant γ and an iteration number K such that for all $k \geq K$ it holds that:*

$$\sum_{i=0}^{k} \frac{1}{\theta_i} \langle \nabla f(y^i), z^i - x^* \rangle \leq \sum_{i=0}^{k} \frac{n\gamma}{\theta_i} \nabla_{j_i} f(y^i)(z^i_{j_i} - x^*_{j_i}) \ , \tag{18}$$

where $j_i = \arg\max_i \frac{1}{\sqrt{L_i}} |\nabla_i f(y^k)|$ is the greedy coordinate at iteration i.

One can show that this condition is sufficient to prove an accelerated convergence rate $O(1/k^2)$ for AGCD. Therefore let us take a close look at Technical Condition 4.1. The condition considers the weighted sum (with weights $\frac{1}{\theta_i} \sim O(i^2)$) of the inner product of $\nabla f(y^k)$ and $z^k - x^*$, and the condition states that the inner product corresponding to the greedy coordinate (the right side above) is larger than the average of all coordinates in the inner product, by a factor of γ. In the case of ARCD and ASCD, it is easy to show that Technical Condition 4.1 holds automatically up to expectation, with $\gamma = 1$.

Here is an informal explanation of why Technical Condition 4.1 ought to hold for most convex functions and most iterations of AGCD. When k is sufficiently large, the three sequence $\{x^k\}$, $\{y^k\}$ and $\{z^k\}$ ought to all converge to x^* (which always happens in practice though lack of theoretical justification), whereby z^k is close to y^k. Thus we can instead consider the inner product

$\langle \nabla f(y^k), y^k - x^* \rangle$ in (18). Notice that for any coordinate j it holds that $|y_j^k - x_j^*| \geq \frac{1}{L_j}|\nabla_j f(y^k)|$, and therefore $|\nabla_j f(y^k) \cdot (y_j^k - x_j^*)| \geq \frac{1}{L_j}|\nabla_j f(j^k)|^2$. Now the greedy coordinate is chosen by $j_i := \arg\max_j \frac{1}{L_j}|\nabla_j f(j^k)|^2$, and therefore it is reasonably likely that in most cases the greedy coordinate will yield a better product than the average of the components of the inner product.

The above is not a rigorous argument, and we can likely design some worst-case functions for which Technical Condition 4.1 fails. But the above argument provides some intuition as to why the condition ought to hold in most cases, thereby yielding the observed improvement of AGCD as compared with ARCD that we will shortly present in Section 5, where we also observe that Technical Condition 4.1 holds empirically on all of our problem instances.

With a slight change in the proof of Theorem 2.1, we can show the following result:

Theorem 4.1. *Consider the Accelerated Greedy Coordinate Descent (Algorithm 1 with rule* (4)*). If $f(\cdot)$ is coordinate-wise L-smooth and satisfies Technical Condition 4.1 with constant $\gamma \leq 1$ and iteration number K, then it holds for all $k \geq K$ that:*

$$f(x^k) - f(x^*) \leq \tfrac{2n^2\gamma}{(k+1)^2}\|x^* - x^0\|_L^2 \ . \tag{19}$$

We note that if $\gamma < 1$ (which we always observe in practice), then AGCD will have a better convergence guarantee than ARCD.

Remark 4.1. *The arguments in Section 4.1 and Section 4.2 also work for strongly convex case, albeit with suitable minor modifications.*

5 Numerical Experiments

5.1 Linear Regression

We consider solving synthetic instances of the linear regression model with least-squares objective function:

$$f^* \ := \ \min_{\beta \in \mathbb{R}^p} f(\beta) := \|y - X\beta\|_2^2$$

using ASCD, ARCD and AGCD, where the mechanism for generating the data (y, X) and the algorithm implementation details are described in the supplementary materials. Figure 1 shows the optimality gap versus time (in seconds) for solving different instances of linear regression with different condition numbers of the matrix X^TX using ASCD, ARCD and AGCD. In each plot, the vertical axis is the objective value optimality gap $f(\beta^k) - f^*$ in log scale, and the horizontal axis is the running time in seconds. Each column corresponds to an instance with the prescribed condition number κ of X^TX, where $\kappa = \infty$ means that the minimum eigenvalue of X^TX is 0. The first row of plots is for Algorithm Framework 1 which is ignorant of any strong convexity information. The second row of plots is for Algorithm Framework 2, which uses given strong convexity information. And because the linear regression optimization problem is quadratic, it is straightforward to compute κ as well as the true parameter μ for the instances where $\kappa > 0$. The last column of the figure corresponds to $\kappa = \infty$, and in this instance we set μ using the smallest

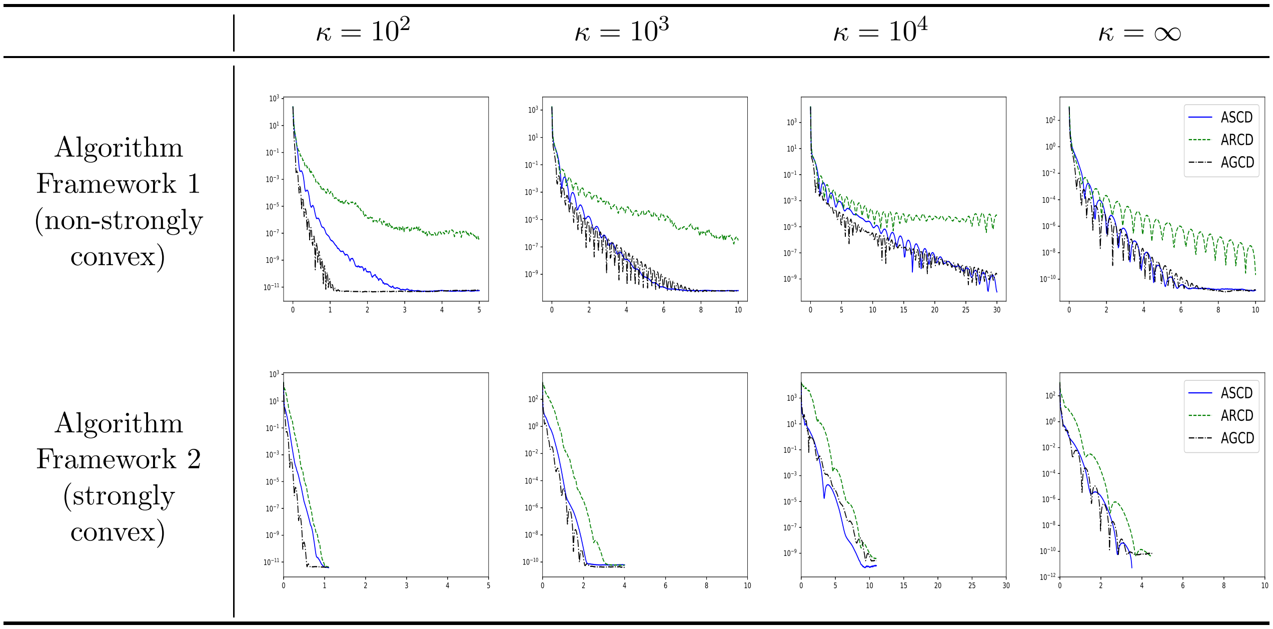

Figure 1: Plots showing the optimality gap versus run-time (in seconds) for synthetic linear regression problems solved by ASCD, ARCD and AGCD.

positive eigenvalue of $X^T X$, which can be shown to work in theory since all relevant problem computations are invariant in the nullspace of X.

Here we see in Figure 1 that AGCD and ASCD consistently have superior performance over ARCD for both the non-strongly convex case and the strongly convex case, with ASCD performing almost as well as AGCD in most instances.

We remark that the behavior of any convex function near the optimal solution is similar to the quadratic function defined by the Hessian at the optimum, and therefore the above numerical experiments show promise that AGCD and ASCD are likely to outperform ARCD asymptotically for any twice-differentiable convex function.

5.2 Logistic Regression

Here we consider solving instances of the logistic regression loss minimization problem:

$$f^* \ := \ \min_{\beta \in \mathbb{R}^p} f(\beta) := \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i \beta^T x_i)) \ , \tag{20}$$

using ASCD, ARCD and AGCD, where $\{x_i, y_i\}$ is the feature-response pair for the i-th data point and $y_i \in \{-1, 1\}$. Although the loss function $f(\beta)$ is not in general strongly convex, it is essentially locally strongly convex around the optimum but with unknown strong convexity parameter $\bar{\mu}$. And although we do not know the local strong convexity parameter $\bar{\mu}$, we can still run the strongly convex algorithm (Algorithm Framework 2) by assigning a value of $\bar{\mu}$ that is hopefully close to the actual value. Using this strategy, we solved a large number of logistic regression instances from

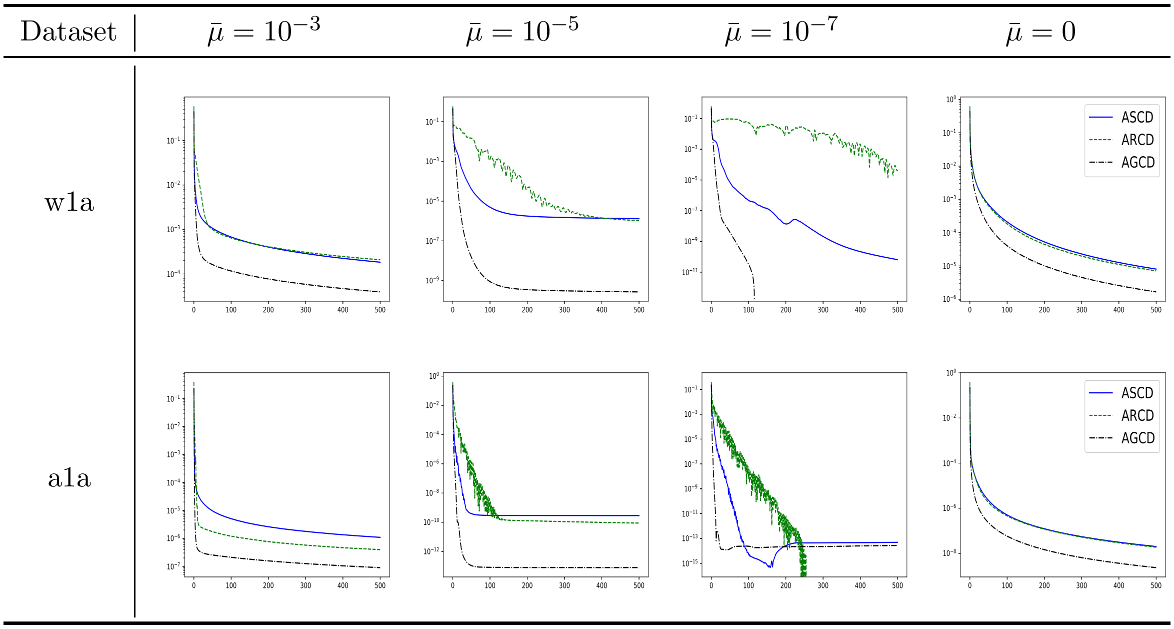

Figure 2: Plots showing the optimality gap versus run-time (in seconds) for the logistic regression instances w1a and a1a in LIBSVM, solved by ASCD, ARCD and AGCD.

LIBSVM [6]. Figure 2 shows the optimality gap versus time (in seconds) for solving two of these instances, namely w1a and a1a, which were chosen here because the performance of the algorithms on these two instances is representative of others in LIBSVM. In each plot, the vertical axis is the objective value optimality gap $f(\beta^k) - f^*$ in log scale, and the horizontal axis is the running time in seconds. Each column corresponds to a different assigned value of the local strong convexity parameter $\bar{\mu}$. The right-most column in the figure uses the assignment $\bar{\mu} = 0$, in which case the algorithms are implemented as in the non-strongly convex case (Algorithm Framework 1).

Here we see in Figure 2 that AGCD always has superior performance as compared to either ASCD and ARCD. In the relevant range of optimality gaps ($\leq 10^{-9}$), ASCD typically outperforms ARCD for smaller values of the assigned strong convexity parameter $\bar{\mu}$. However, the performance of ASCD and ARCD are essentially the same when no strong convexity is presumed.

Last of all, we attempt to estimate the parameter γ that arises in Technical Condition 4.1 for AGCD in several of the datasets in SVMLIB. Although for small k, the ratio between $\sum_{i=0}^{k} \frac{1}{\theta_i} \langle \nabla f(y^i), z^i - x^* \rangle$ and $\sum_{i=0}^{k} \frac{n}{\theta_i} \nabla_{j_i} f(y^i)(z^i_{j_i} - x^*_{j_i})$ can fluctuate widely, this ratio stabilizes after a number of iterations in all of our numerical tests. From Technical Condition 4.1, we know that γ is the upper bound of such ratio for all $k \geq K$ for some large enough value of K. Table 1 presents the observed values of γ for all $K \geq \bar{K} := 5{,}000$. Recalling from Theorem 4.1 that the γ value represents how much better AGCD can perform compared with ARCD in terms of computational guarantees, we see from Table 1 that AGCD should outperform ARCD for these representative instances – and indeed this is what we observe in practice in our tests.

Table 1: Largest observed values of γ for five different datasets in LIBSVM for $k \geq \bar{K} := 5000$.

Dataset:	w1a	a1a	heart	madelon	rcv1
γ:	0.25	0.17	0.413	0.24	0.016

A Appendix

A.1 Proof of Theorem 3.1

In order to prove Theorem 3.1, we first prove the following three lemmas:

Lemma A.1.

$$a^2\|x^* - z^k\|_L^2 + b\|x^* - y^k\|_L^2 = (a^2+b)\|x^* - u^k\|_L^2 + \tfrac{a^2 b}{a^2+b}\|y^k - z^k\|_L^2 \ .$$

Proof:

$$\begin{aligned}
& (a^2+b)\|x^* - u^k\|_L^2 + \tfrac{a^2 b}{a^2+b}\|y^k - z^k\|_L^2 \\
= \quad & (a^2+b)\left(\|x^*\|_L^2 - 2\langle x^*, u^k\rangle_L + \|u^k\|_L^2\right) + \tfrac{a^2 b}{a^2+b}\|y^k - z^k\|_L^2 \\
= \quad & (a^2+b)\|x^*\|_L^2 - 2\langle x^*, a^2 z^k + b y^k\rangle_L + \tfrac{1}{a^2+b}\|a^2 z^k + b y^k\|_L^2 + \tfrac{a^2 b}{a^2+b}\|y^k - z^k\|_L^2 \qquad (21) \\
= \quad & (a^2+b)\|x^*\|_L^2 - 2\langle x^*, a^2 z^k + b y^k\rangle_L + a^2\|z^k\|_L^2 + b\|y^k\|_L^2 \\
= \quad & a^2\|x^* - z^k\|_L^2 + b\|x^* - y^k\|_L^2 \ ,
\end{aligned}$$

where the second equality utilizes $u^k = \frac{a^2}{a^2+b} z^k + \frac{b}{a^2+b} y^k$ and the other equalities are just mathematical manipulations. □

Lemma A.2. *Define* $t^{k+1} := u^k - \frac{a}{a^2+b}\frac{1}{n}\mathbf{L}^{-1}\nabla f(y^k)$*, then*

$$\|x^* - t^{k+1}\|_L^2 - \|x^* - u^k\|_L^2 = nE_{j_k^2}\left[\|x^* - z^{k+1}\|_L^2 - \|x^* - u^k\|_L^2\right]$$

Proof:

$$\begin{aligned}
\|x^* - t^{k+1}\|_L^2 - \|x^* - u^k\|_L^2 \quad &= \quad 2\langle x^* - u^k, u^k - t^{k+1}\rangle_L + \|u^k - t^{k+1}\|_L^2 \\
&= \quad 2nE_{j_k^2}\left[\langle x^* - u^k, u^k - z^{k+1}\rangle_L + \|u^k - z^{k+1}\|_L^2\right] \qquad (22) \\
&= \quad nE_{j_k^2}\left[\|x^* - z^{k+1}\|_L^2 - \|x^* - u^k\|_L^2\right] ,
\end{aligned}$$

where the second equality is from the relationship of $t^{k+1} = u^k - \frac{a}{a^2+b}\frac{1}{n}\mathbf{L}^{-1}\nabla f(y^k)$ and $z^{k+1} = u^k - \frac{a}{a^2+b}\frac{1}{nL_{j_k^2}}\nabla_{j_k^2} f(y^k) e_{j_k^2}$, and the first and third equations are just rearrangement. □

Lemma A.3.

$$a^2 \le (1-a)(a^2+b)\ .$$

Proof: Remember that $b = \frac{\mu a}{n^2}$ and $a > 0$, thus the above inequality is equivalent to

$$a \le (1-a)(a + \tfrac{\mu}{n^2})\ .$$

Substituting $a = \frac{\sqrt{\mu}}{n+\sqrt{\mu}}$, the above inequality becomes

$$\tfrac{\sqrt{\mu}}{n} \le \tfrac{\sqrt{\mu}}{n+\sqrt{\mu}} + \tfrac{\mu}{n^2}\ .$$

We furnish the proof by noting $\frac{\sqrt{\mu}}{n} - \frac{\sqrt{\mu}}{n+\sqrt{\mu}} = \frac{\mu}{n(n+\sqrt{\mu})} \le \frac{\mu}{n^2}$. □

Proof of Theorem 3.1: Recall that $t^{k+1} = u^k - \frac{a}{a^2+b}\frac{1}{n}\mathbf{L}^{-1}\nabla f(y^k)$, then it is easy to check that

$$t^{k+1} = \arg\min_z\ a\langle \nabla f(y^k), z - z^k\rangle + \frac{n}{2}a^2\|z - z^k\|_L^2 + \frac{n}{2}b\|z - y^k\|_L^2$$

by writing the optimality conditions of the right-hand side.

We have

$$\begin{aligned}
& f(x^{k+1}) - f(y^k) \\
\le\ & \langle \nabla f(y^k), x^{k+1} - y^k\rangle + \tfrac{1}{2}\|x^{k+1} - y^k\|_L^2 \\
=\ & -\tfrac{1}{2L_{j_k^1}}\left(\nabla_{j_k^1} f(y^k)\right)^2 \\
\le\ & -\tfrac{1}{2n}\|\nabla f(y^k)\|_{L^{-1}}^2 \\
\le\ & a\langle \nabla f(y^k), t^{k+1} - z^k\rangle + \tfrac{n}{2}a^2\|t^{k+1} - z^k\|_L^2 \\
=\ & a\langle \nabla f(y^k), x^* - z^k\rangle + \tfrac{n}{2}a^2\|x^* - z^k\|_L^2 - \tfrac{n}{2}a^2\|x^* - t^{k+1}\|_L^2 + \tfrac{n}{2}b\|x^* - y^k\|_L^2 \\
& -\tfrac{n}{2}b\|y^k - t^{k+1}\|_L^2 - \tfrac{n}{2}b\|t^{k+1} - x^*\|_L^2 \\
\le\ & a\langle \nabla f(y^k), x^* - z^k\rangle + \tfrac{n}{2}a^2\|x^* - z^k\|_L^2 - \tfrac{n}{2}a^2\|x^* - t^{k+1}\|_L^2 + \tfrac{n}{2}b\|x^* - y^k\|_L^2 - \tfrac{n}{2}b\|t^{k+1} - x^*\|_L^2 \\
=\ & a\langle \nabla f(y^k), x^* - z^k\rangle + \tfrac{n}{2}(a^2+b)\left(\|x^* - u^k\|_L^2 - \|x^* - t^{k+1}\|_L^2\right) + \tfrac{n}{2}\tfrac{a^2 b}{a^2+b}\|y^k - z^k\|_L^2 \\
=\ & a\langle \nabla f(y^k), x^* - z^k\rangle + \tfrac{n^2}{2}(a^2+b)E_{j_k^2}\left[\|x^* - u^k\|_L^2 - \|x^* - z^{k+1}\|_L^2\right] + \tfrac{n}{2}\tfrac{a^2 b}{a^2+b}\|y^k - z^k\|_L^2 \\
\le\ & a\langle \nabla f(y^k), x^* - z^k\rangle + \tfrac{n^2}{2}(a^2+b)E_{j_k^2}\left[\|x^* - u^k\|_L^2 - \|x^* - z^{k+1}\|_L^2\right] + \tfrac{n^2}{2}\tfrac{a^2 b}{a^2+b}\|y^k - z^k\|_L^2 \\
=\ & a\langle \nabla f(y^k), x^* - z^k\rangle + \tfrac{n^2}{2}\left((a^2+b)\|x^* - u^k\|_L^2 + \tfrac{a^2 b}{a^2+b}\|y^k - z^k\|_L^2\right) - \tfrac{n^2}{2}(a^2+b)\mathbb{E}_{j_k^2}\left[\|x^* - z^{k+1}\|_L^2\right] \\
=\ & a\langle \nabla f(y^k), x^* - z^k\rangle + \tfrac{n^2}{2}\left(a^2\|x^* - z^k\|_L^2 + b\|x^* - y^k\|_L^2\right) - \tfrac{n^2}{2}(a^2+b)E_{j_k^2}\left[\|x^* - z^{k+1}\|_L^2\right],
\end{aligned} \tag{23}$$

where the first inequality is due to coordinate-wise smoothness, the first equality utilizes $x^{k+1} = y^k - \frac{1}{L_{j_k^1}}\nabla_{j_k^1} f(y^k) e_{j_k^1}$, the second inequality follows from the fact that j_k^1 is the greedy coordinate of $\nabla f(y^k)$ in the $\|\cdot\|_{L^{-1}}$ norm, the third inequality follows from the basic inequality $\|v\|_L^2 + \|w\|_{L^{-1}}^2 \geq 2\langle v, w\rangle$ for all v, w, the second equality is from Three Point Property by noticing

$$t^{k+1} = \arg\min_z \; a\langle \nabla f(y^k), z - z^k\rangle + \frac{n}{2}a^2\|z - z^k\|_L^2 + \frac{n}{2}b\|z - y^k\|_L^2 \ ,$$

the third equality follows from Lemma 3.1, and the fourth and sixth equalities each utilize Lemma 3.2.

On the other hand, by strong convexity we have

$$\begin{aligned} f(y^k) - f(x^*) &\leq \langle \nabla f(y^k), y^k - x^*\rangle - \tfrac{1}{2}\mu\|y^k - x^*\|_L^2 \\ &= \langle \nabla f(y^k), y^k - z^k\rangle + \langle \nabla f(y^k), z^k - x^*\rangle - \tfrac{1}{2}\mu\|y^k - x^*\|_L^2 \\ &= \tfrac{1-a}{a}\langle \nabla f(y^k), x^k - y^k\rangle + \langle \nabla f(y^k), z^k - x^*\rangle - \tfrac{1}{2}\mu\|y^k - x^*\|_L^2 \\ &\leq \tfrac{1-a}{a}\big(f(x^k) - f(y^k)\big) + \langle \nabla f(y^k), z^k - x^*\rangle - \tfrac{1}{2}\mu\|y^k - x^*\|_L^2, \end{aligned} \tag{24}$$

where the second equality uses the fact that $y^k = (1-a)x^k + az^k$ and the last inequality is from the gradient inequality.

By rearranging (24), we obtain

$$f(y^k) - f(x^*) \leq (1-a)\Big(f(x^k) - f(x^*)\Big) + a\langle \nabla f(y^k), z^k - x^*\rangle - \frac{1}{2}\mu a\|y^k - x^*\|_L^2 \ . \tag{25}$$

Notice that $b = \frac{\mu a}{n^2}$ and $a^2 \leq (1-a)(a^2 + b)$ following from Lemma 3.3. Thus summing up (23) and (25) leads to

$$\begin{aligned} &E_{j_k^2}\Big[f(x^{k+1}) - f(x^*) + \tfrac{n^2}{2}(a^2 + b)\|z^{k+1} - x^*\|_L^2\Big] \\ \leq\ & (1-a)\big(f(x^k) - f(x^*)\big) + \tfrac{n^2}{2}a^2\|z^k - x^*\|_L^2 \\ \leq\ & (1-a)\Big(f(x^k) - f(x^*) + \tfrac{n^2}{2}(a^2 + b)\|z^k - x^*\|_L^2\Big) \ , \end{aligned} \tag{26}$$

which furnishes the proof using a telescoping series. □

B More Material on the Numerical Experiments

B.1 Implementation Detail

To be consistent with the notation in statistics and machine learning we use p to denote the dimension of the variables in the optimization problems describing linear and logistic regression. Then the per-iteration computation cost of AGCD and ASCD is dominated by three computations:

(i) p-dimensional vector operations (such as in computing y^k using x^k and z^k), (ii) computation of the gradient $\nabla f(\cdot)$, and (iii) computation of the maximum (weighted) magnitude coordinate of the gradient $\nabla f(\cdot)$. [12] proposed an efficient way to avoid (i) by changing variables. Distinct from the dual approaches discussed in [12], [14], and [2], here we only consider the primal problem in the regime $n > p$, and therefore the cost of (ii) dominates the cost of (i) in these cases. For this reason in our numerical experiments we use the simple implementation of ARCD proposed by Nesterov [21] and which we adopt for AGCD and ASCD as well. We note that both the randomized methods and the greedy methods can take advantage of the efficient calculations proposed in [12] as well.

For the linear regression experiments we focused on synthetic problem instances with different condition numbers κ of the matrix $X^T X$ and where X is dense. In this case the cost for computation (i) is $O(p)$. And by taking advantage of the coordinate update structure, we can implement (ii) in $O(p)$ operations by pre-computing and storing $X^T X$ in memory, see [21] and [14] for further details. The cost of (iii) is simply $O(p)$.

The data (X, y) for the linear regression problems is generated as follows. For a given number of samples n and problem dimension p (in the experiments we used $n = 200$ and $p = 100$), we generate a standard Gaussian random matrix $\bar{X} \in \mathbb{R}^{p \times n}$ with each entry drawn $\sim N(0, 1)$. In order to generate the design matrix X with fixed condition number κ, we first decompose of $\bar{X}$ as $\bar{X} = U^T \bar{D} V$. Then we rescale the diagonal matrix $\bar{D}$ of singular values linearly to D such that the smallest singular value of D is $\frac{1}{\sqrt{\kappa}}$ and the largest singular value of D is 1. We then compute the final design matrix $X = U^T D V$ and therefore the condition number of $X^T X$ becomes κ. We generate the response vector y using the linear model $y \sim N(X\beta^*, \sigma^2)$, with true model β^* chosen randomly by a Gaussian distribution as well. For the cases with finite κ, we are able to compute the strong convexity parameter μ exactly because the objective function is quadratic, and we use that μ to implement our Algorithm Framework for strongly convex problems (Algorithm Framework 2). When $\kappa = \infty$, we instead use the smallest positive eigenvalue of $X^T X$ to compute μ.

For the logistic regression experiments, the cost of (ii) at each iteration of AGCD and ASCD can be much larger than $O(p)$ because there is no easy way to update the full gradient $\nabla f(\cdot)$. For these problems we have

$$\nabla f(\beta) = -\frac{1}{n} X^T w(\beta) \tag{27}$$

where X is the sample matrix with x_i composing the i-th row, and $w(\beta)_i := \frac{1}{1+\exp(y_i \beta^T x_i)}$. Notice that calculating $w(\beta)$ can be done using a rank 1-update with cost $O(n)$. But calculating the matrix-vector product $X^T w(\beta)$ will cost $O(np)$, which dominates the cost of (i) and/or (iii). However, in the case when X is a sparse matrix with density ρ, the cost can be decreased to $O(\rho np)$.

B.2 Comparing the Algorithms using Running Time and the Number of Iterations

Figure 3 shows the optimality gap versus running time (seconds) in the left plot and and versus the number of iterations in the right plot, logistic regression problem using the dataset madelon in LIBSVM [6], with $\bar{\mu} = 10^{-7}$. Here we see that AGCD and ASCD are vastly superior to ARCD in

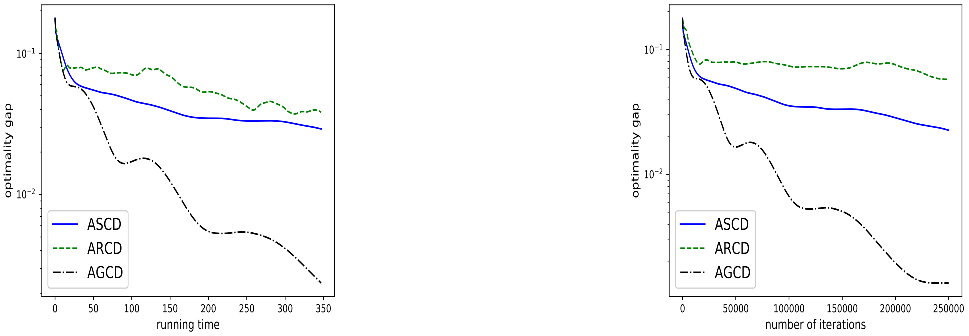

Figure 3: Plots showing the optimality gap versus run-time (in seconds) on the left and versus the number of iterations on the right, for the logistic regression instance madelon with $\bar{\mu} = 10^{-7}$, solved by ASCD, ARCD and AGCD.

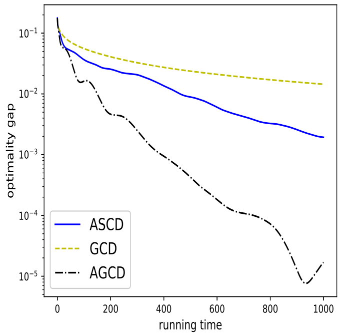

Figure 4: Plots showing the optimality gap versus run-time (in seconds) for the logistic regression instance madelon with $\bar{\mu} = 10^{-6}$, solved by ASCD, GCD and AGCD.

term of the number of iterations, but not nearly as much in terms of running time, because one iteration of AGCD or ASCD can be more expensive than an iteration of ARCD.

B.3 Comparing Accelerated Method with Non-Accelerated Method

Figure 4 shows the optimality gap versus running time (seconds) with GCD, ASCD and AGCD for logistic regression problem using the dataset madelon in LIBSVM [6], with $\bar{\mu} = 10^{-6}$. Here we see that ASCD and AGCD are superior to non-accelerated GCD.

B.4 Numerical Results for Logistic Regression with Other Datasets

We present numerical results for logistic regression problems for several other datasets in LIBSVM solved by ASCD, ARCD and AGCD in Figure 5. Here we see that AGCD always has superior performance as compared to ASCD and ARCD, and ASCD outperforms ARCD in most of the cases.

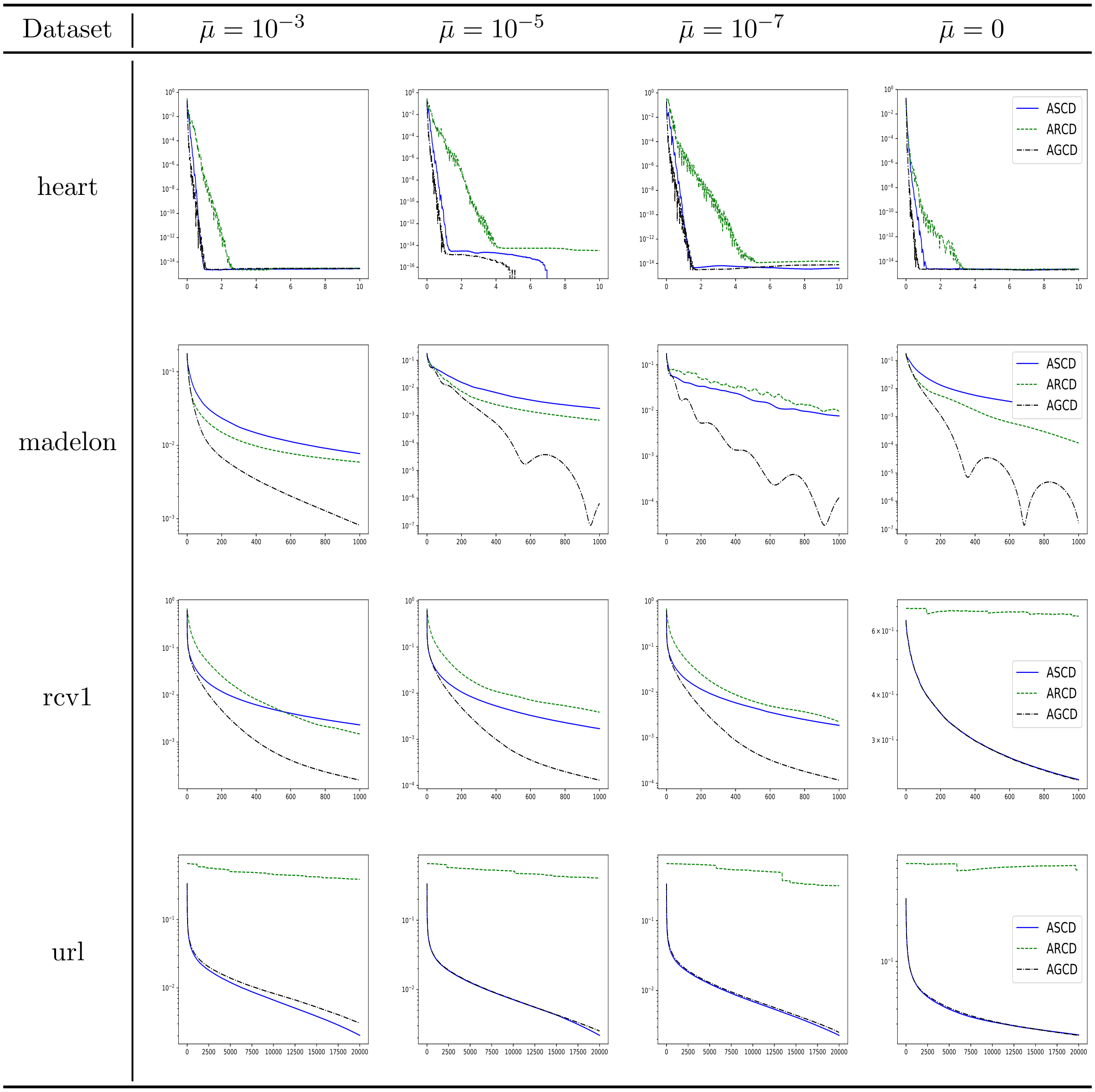

Figure 5: Plots showing the optimality gap versus run-time (in seconds) for some other logistic regression instances in LIBSVM, solved by ASCD, ARCD and AGCD.

References

[1] Zeyuan Allen-Zhu and Lorenzo Orecchia, *Linear coupling: An ultimate unification of gradient and mirror descent*, arXiv preprint arXiv:1407.1537 (2014).

[2] Zeyuan Allen-Zhu, Zheng Qu, Peter Richtarik, and Yang Yuan, *Even faster accelerated coordinate descent using non-uniform sampling*, International Conference on Machine Learning, 2016.

[3] Amir Beck and Luba Tetruashvili, *On the convergence of block coordinate descent type methods*, SIAM journal on Optimization **23** (2013), no. 4, 2037–2060.

[4] Dimitri Bertsekas and John Tsitsiklis, *Parallel and distributed computation: numerical methods*, vol. 23, Prentice hall Englewood Cliffs, NJ, 1989.

[5] Sébastien Bubeck, Yin Tat Lee, and Mohit Singh, *A geometric alternative to nesterov's accelerated gradient descent*, arXiv preprint arXiv:1506.08187 (2015).

[6] Chih-Chung Chang and Chih-Jen Lin, *Libsvm: a library for support vector machines*, ACM transactions on intelligent systems and technology (TIST) **2** (2011), no. 3, 27.

[7] Olivier Fercoq and Peter Richtarik, *Accelerated, parallel, and proximal coordinate descent*, SIAM Journal on Optimization **25** (2015), no. 4, 1997–2023.

[8] Roy Frostig, Rong Ge, Sham Kakade, and Aaron Sidford, *Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization*, International Conference on Machine Learning, 2015.

[9] Mert Gurbuzbalaban, Asuman Ozdaglar, Pablo A Parrilo, and Nuri Vanli, *When cyclic coordinate descent outperforms randomized coordinate descent*, Advances in Neural Information Processing Systems, 2017, pp. 7002–7010.

[10] Bin Hu and Laurent Lessard, *Dissipativity theory for Nesterov's accelerated method*, arXiv preprint arXiv:1706.04381 (2017).

[11] Thorsten Joachims, *Advances in kernel methods*, MIT Press, Cambridge, MA, USA, 1999, pp. 169–184.

[12] Yin Tat Lee and Aaron Sidford, *Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems*, Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science (Washington, DC, USA), FOCS '13, IEEE Computer Society, 2013, pp. 147–156.

[13] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui, *A universal catalyst for first-order optimization*, Advances in Neural Information Processing Systems, 2015.

[14] Qihang Lin, Zhaosong Lu, and Lin Xiao, *An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization*, SIAM Journal on Optimization **25** (2015), no. 4, 2244–2273.

[15] Francesco Locatello, Anant Raj, Sai Praneeth Reddy, Gunnar Rätsch, Bernhard Schölkopf, Sebastian U Stich, and Martin Jaggi, *On matching pursuit and coordinate descent*, ICML 2018 - Proceedings of the 35th International Conference on Machine Learning (2018).

[16] Zhaosong Lu and Lin Xiao, *On the complexity analysis of randomized block-coordinate descent methods*, Mathematical Programming **152** (2015), no. 1-2, 615–642.

[17] Zhi-Quan Luo and Paul Tseng, *On the convergence of the coordinate descent method for convex differentiable minimization*, Journal of Optimization Theory and Applications **72** (1992), no. 1, 7–35.

[18] ______, *Error bounds and convergence analysis of feasible descent methods: a general approach*, Annals of Operations Research **46** (1993), no. 1, 157–178.

[19] Rahul Mazumder and Trevor Hastie, *The graphical lasso: new insights and alternatives*, Electronic Journal of Statistics **6** (2012), 2125.

[20] A. S. Nemirovsky and D. B. Yudin, *Problem complexity and method efficiency in optimization*, Wiley, New York, 1983.

[21] Yu Nesterov, *Efficiency of coordinate descent methods on huge-scale optimization problems*, SIAM Journal on Optimization **22** (2012), no. 2, 341–362.

[22] Yurii Nesterov, *A method of solving a convex programming problem with convergence rate* $O(1/k^2)$, Soviet Mathematics Doklady, vol. 27, 1983, pp. 372–376.

[23] Julie Nutini, Mark Schmidt, Issam Laradji, Michael Friedlander, and Hoyt Koepke, *Coordinate descent converges faster with the Gauss-Southwell rule than random selection*, International Conference on Machine Learning, 2015, pp. 1632–1641.

[24] John C. Platt, *Advances in kernel methods*, MIT Press, Cambridge, MA, USA, 1999, pp. 185–208.

[25] Peter Richtarik and Martin Takac, *Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function*, Mathematical Programming **144** (2014), no. 1-2, 1–38.

[26] Chaobing Song, Shaobo Cui, Yong Jiang, and Shu-Tao Xia, *Accelerated stochastic greedy coordinate descent by soft thresholding projection onto simplex*, Advances in Neural Information Processing Systems, 2017, pp. 4841–4850.

[27] Weijie Su, Stephen Boyd, and Emmanuel J Candes, *A differential equation for modeling Nesterovs accelerated gradient method: theory and insights*, Journal of Machine Learning Research **17** (2016), no. 153, 1–43.

[28] Ruoyu Sun and Yinyu Ye, *Worst-case complexity of cyclic coordinate descent:* $O(n^2)$ *gap with randomized version*, arXiv preprint arXiv:1604.07130 (2016).

[29] P. Tseng, *On accelerated proximal gradient methods for convex-concave optimization*, Tech. report, May 21, 2008.

[30] Ashia C Wilson, Benjamin Recht, and Michael I Jordan, *A Lyapunov analysis of momentum methods in optimization*, arXiv preprint arXiv:1611.02635 (2016).

[31] Yang You, Xiangru Lian, Ji Liu, Hsiang-Fu Yu, Inderjit S Dhillon, James Demmel, and Cho-Jui Hsieh, *Asynchronous parallel greedy coordinate descent*, Advances in Neural Information Processing Systems, 2016, pp. 4682–4690.