2013 IEEE 54th Annual Symposium on Foundations of Computer Science

Efficient Accelerated Coordinate Descent Methods and Faster Algorithms for Solving Linear Systems

Yin Tat Lee
Mathematics Department
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
Email: yintat@mit.edu

Aaron Sidford
Electrical Engineering and Computer Science Department
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
Email: sidford@mit.edu

***Abstract*—In this paper we show how to accelerate randomized coordinate descent methods and achieve faster convergence rates without paying per-iteration costs in asymptotic running time. In particular, we show how to generalize and efficiently implement a method proposed by Nesterov, giving faster asymptotic running times for various algorithms that use standard coordinate descent as a black box. In addition to providing a proof of convergence for this new general method, we show that it is numerically stable, efficiently implementable, and in certain regimes, asymptotically optimal.**

To highlight the power of this algorithm, we show how it can used to create faster linear system solvers in several regimes:

- **We show how this method achieves a faster asymptotic runtime than conjugate gradient for solving a broad class of symmetric positive definite systems of equations.**
- **We improve the convergence guarantees for Kaczmarz methods, a popular technique for image reconstruction and solving overdetermined systems of equations, by accelerating an algorithm of Strohmer and Vershynin.**
- **We achieve the best known running time for solving Symmetric Diagonally Dominant (SDD) system of equations in the unit-cost RAM model, obtaining a running time of O(m log^{3/2}n (log log n)^{1/2} log((log n)/eps)) by accelerating a recent solver by Kelner *et al.***

Beyond the independent interest of these solvers, we believe they highlight the versatility of the approach of this paper and we hope that they will open the door for further algorithmic improvements in the future.

***Keywords*-convex optimization; coordinate descent; Kaczmarz method; symmetric diagonally dominant matrix**

I. Introduction

In recent years iterative methods for convex optimization that make progress in sublinear time using only partial information about the function and its gradient have become of increased importance to computer science. Beyond the hope for performance gains in practice, rapidly converging algorithms with sublinear time update steps create hope for new provable asymptotic running times for old problems and stronger guarantees for efficient algorithms in distributed and asynchronous settings.

The idea of using simple sublinear-time iterative steps to solve convex optimization problems is an old one [1], [2], [3]. It is an algorithmic design principle that has seen great practical success [4], [5], [3] but has been notoriously difficult to analyze. In the past few years great strides have been taken towards developing a theoretical understanding of randomized variants of these approaches. Of particular relevance to this paper, in 2006 Strohmer and Vershynin [6] showed that a particular sublinear update algorithm for solving overconstrained linear systems called *randomized Kaczmarz* converges exponentially, in 2010 Nesterov [7] analyzed randomized analog of gradient descent that updates only a single coordinate in each iteration, called *coordinate gradient descent method*, and provided a computationally inefficient but theoretically interesting accelerated variant, called *accelerated coordinate gradient descent method (ACDM)*, and in 2013 Kelner et al. [8] presented a simple combinatorial iterative algorithm with sublinear-time update steps that can be used to solve *symmetric diagonally dominant (SDD)* linear systems, a broad class of linear systems with numerous applications.

In this paper we provide a framework that both strengthens and unifies these results. We present a more general version of Nesterov's ACDM and show how to implement it so that each iteration has the same asymptotic runtime as its non-accelerated variants. We show that this method is numerically stable and optimal under certain assumptions. Then we show how to use this method to outperform conjugate gradient in solving a general class of symmetric positive definite systems of equations. Furthermore, we show how to cast both randomized Kaczmarz and the SDD solver of Kelner et al. in this framework and achieve faster running times through the use of ACDM.

Due to the success of the Kaczmarz method in practice [4], [5] and the wide array of theoretical problems for which the fastest running time is obtained through a SDD solver [9], [10], [11], we hope that the ideas in this paper can be used to make advancements on both fronts.

A. Previous Work

Given a convex differentiable function the *gradient descent method* is a simple greedy iterative method that computes the gradient at the current point and uses that

0272-5428/13 $26.00 © 2013 IEEE
DOI 10.1109/FOCS.2013.24

information to perform an update and make progress. This method is central to much of scientific computing and from a theoretical perspective the standard method is well understood [12]. There are multiple more sophisticated variants of this method [13], but many of them have only estimates of local convergence rates which makes them difficult to be applied to theoretical problems and be compared in general.

In 1983, Nesterov [14] proposed a way to accelerate the gradient descent method by iteratively developing an approximation to the function through what he calls an *estimate sequence*. This *accelerated gradient descent method* has the same worst case running time as conjugate gradient method and it is applicable to general convex functions. Recently, this method has been used to improve the fastest known running time of some fundamental problems in computer science, such as compressive sensing [15], [16], undirected maximum flow [17], linear programming [18], [19].

The accelerated gradient descent method is known to achieve an optimal convergence rate among all first order methods, that is algorithm that only have access to the function's value and gradient [12]. Therefore, to further improve accelerated gradient descent one must either assume more information about the function or find a way to reduce the cost of each iteration. Using the idea of fast but crude iteration steps, Nesterov proposed a randomized coordinate descent method [7], which minimizes convex functions by updating one randomly chosen coordinate in each iteration.

Coordinate descent methods, which use gradient information about a single coordinate to update a single coordinate in each iteration, have been around for a long time [2]. Various update schemes have been considered, such as cyclic coordinate update and the best coordinate update, however these schemes are either hard to estimate [20] or difficult to be implemented efficiently. Both the recent work of Strohmer and Vershynin [6] and Nesterov [7] overcame these obstacles by showing that by performing particular randomized updates one can produce methods with provable global convergence rate with small costs.

Applying the similar ideas of accelerated gradient descent, Nesterov also proposed an accelerated variant called the *accelerated coordinate descent method (ACDM)* that achieves a faster convergence rate. However, in both Nesterov's paper [7] and later work [21], this method was considered inefficient as the naive implementation of each iteration of ACDM requires $\Theta(n)$ time to update every coordinate of the input, at which point the accelerated gradient descent method would seem preferable.

B. Our Contributions

In this paper, we generalize Nesterov's ACDM and present a simple technique to implement update steps efficiently. Our contributions are as follows:

- **Generalization:** A generalization of ACDM to a broader class of sampling probabilities, overcoming technical challenges due to skewed sampling probabilities, so that any convergence rate achieved through Nesterov's coordinate descent method can be improved by ACDM. This generalization was essential for applications considered later in the paper.
- **Efficiency:** A proof that under mild assumptions about the oracle for querying function and gradient values, each iteration of ACDM can be implemented with the same asymptotic cost as a coordinate descent step.
- **Numerical Stability:** A proof that ACDM is numerically stable and can be implemented with finite precision arithmetic and no overhead in the standard unit cost RAM model.
- **Lower Bound:** A lower bound argument showing that ACDM achieves an optimal convergence rate among a certain class of coordinate descent algorithms.

In some sense, the principle difference between the asymptotic running time of ACDM and accelerated gradient descent (or conjugate gradient for linear case) is that as accelerated gradient descent depends on the maximum eigenvalue of the Hessian of the function being minimized, ACDM instead depends on the trace of the Hessian and has the possibility of each iteration costing a small fraction of the cost a single iteration of of accelerated gradient descent. As a result, any nontrivial bound on the trace of the Hessian and the computational complexity of performing a single coordinate update creates the opportunity for ACDM to yield improved running times. To emphasize this point, we focus on applications of our method to solving linear systems and we use three different cases to illustrate the flexibility and competitiveness of our method.

- **Symmetric Positive Definite Systems:** We show that under mild assumptions ACDM solves positive definite systems with a faster asymptotic running time than conjugate gradient (and even milder assumptions for Chebyshev method), and it is an optimal algorithm for solving general systems in certain regimes.
- **Overdetermined Systems:** For over-constrained systems of equations the randomized Kaczmarz method of Strohmer and Vershynin [6], which iteratively picks a random constraint and projects the current solution onto the plane corresponding to a random constraint, has been shown to have strong convergence guarantees and appealing practical performance. We show how to cast this method in the framework of coordinate descent and accelerate it using ACDM yielding improved asymptotic performance. Given the appeal of Kaczmarz methods for practical applications such as image reconstructions [5] , there is hope that this could yield improved performance in practice.
- **Symmetric Diagonally Dominant (SDD) Systems:** Since a breakthrough result in 2004 by Spielman and Teng [22] showed that such systems can be solved

in nearly-linear time, such systems have been used to create the fastest algorithm for a variety of problems ranging from max flow [23] to sampling random spanning trees [9] and much more. The fastest known solver for these systems in the standard unit-cost RAM model is due to Kelner *et. al.* [8] is $\tilde{O}(m \log^2 n \log \frac{1}{\epsilon})$ where m is the number of nonzero entries in the $n \times n$ matrix, and we use $\tilde{O}$ to hide $O(\text{poly}(\log \log n))$ terms. We show how direct application of ACDM to a simple algorithm in [8] yields a faster SDD solver with an asymptotic runtime of $\tilde{O}(m \log^{1.5} n \log \frac{1}{\epsilon})$ in the unit-cost RAM model, getting closer to the fastest known running time of $\tilde{O}(m \log n \log \frac{1}{\epsilon})$ by Koutis, Miller, and Peng [24] in a less restrictive computational model.

Just as the accelerated gradient descent method has improved the theoretical and empirical running time of various gradient descent algorithms [25], [15], we hope that ACDM will improve the running time of various algorithms for which coordinate descent based approaches have proven effective. Given the generality of our analysis and the previous difficulty in analyzing such methods, we hope that this is just the next step towards a new class of provably efficient algorithms with good empirical performance.

II. Preliminaries

In this paper, we consider the unconstrained minimization problem $\min_{\vec{x} \in \mathbb{R}^n} f(\vec{x})$ where the *objective function* $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and convex. We let $f^* \stackrel{\text{def}}{=} \min_{\vec{x} \in \mathbb{R}^n} f(\vec{x})$ denote the minimum value of this optimization problem and we let $\vec{x}^* \stackrel{\text{def}}{=} \arg\min_{\vec{x} \in \mathbb{R}^n} f(\vec{x})$ denote an arbitrary point that achieves this value.

To minimize f, we restrict our attention to *first-order* iterative methods, that is algorithms that generate a sequence of points $\vec{x}_k$ such that $\lim f(\vec{x}_k) = f^*$, while only evaluating the objective function and its gradient at points. To compare such algorithms, we say *an iterative method has convergence rate* r if $f(\vec{x}_k) - f^* \leq O((1-r)^k)$ for this method.

Now, we say that f *has convexity parameter* σ *with respect to some norm* $\|\cdot\|$ if the following holds

$$\forall \vec{x}, \vec{y} \in \mathbb{R}^n \quad : \quad f(\vec{y}) \geq f(\vec{x}) + \langle \nabla f(\vec{x}), \vec{y} - \vec{x} \rangle + \frac{\sigma}{2} \|\vec{y} - \vec{x}\|^2 \quad (1)$$

and we say f *strongly convex* if $\sigma > 0$. We refer to the right hand side of (1) as the *lower envelope of* f *at* $\vec{x}$ and for notational convenience when the norm $\|\cdot\|$ is not specified explicitly we assume it to be the standard Euclidian norm $\|x\| \stackrel{\text{def}}{=} \sqrt{\sum_i \vec{x}_i^2}$.

Furthermore, we say f has L-Lipschitz gradient if

$$\forall \vec{x}, \vec{y} \in \mathbb{R}^n \quad : \quad \|\nabla f(\vec{y}) - \nabla f(\vec{x})\| \leq L \|\vec{y} - \vec{x}\|$$

The definition is related to an upper bound on f as follows:

Lemma 1. *[12, Thm 2.1.5] For continuously differentiable* $f : \mathbb{R}^n \to \mathbb{R}$*, it has* L*-Lipschitz gradient if and only if*

$$\forall \vec{x}, \vec{y} \in \mathbb{R}^n : f(\vec{y}) \leq f(\vec{x}) + \langle \nabla f(\vec{x}), \vec{y} - \vec{x} \rangle + \frac{L}{2} \|\vec{x} - \vec{y}\|^2 \quad . \quad (2)$$

We call the right hand side of (2) the *upper envelope of* f *at* $\vec{x}$. The convexity parameter μ and the Lipschitz constant of the gradient L provide lower and upper bounds on f. They serve as the essential characterization of f for first-order methods. For twice differentiable f, these values can also be computed by properties of the Hessian of f by the following well known lemma:

Lemma 2 ([7])**.** *Twice differentiable* $f : \mathbb{R}^n \to \mathbb{R}$ *has convexity parameter* μ *and* L*-Lipschitz gradient with respect to norm* $\|\cdot\|$ *if and only if* $\forall \vec{x} \in \mathbb{R}^n$ *the Hessian of* f *at* $\vec{x}$*,* $\nabla^2 f(\vec{x}) \in \mathbb{R}^{n \times n}$ *satisfies*

$$\forall \vec{y} \in \mathbb{R}^n \quad : \quad \mu \|\vec{y}\|^2 \leq \vec{y}^T \left(\nabla^2 f(\vec{x}) \right) \vec{y} \leq L \|\vec{y}\|^2 .$$

To analyze *coordinate-based iterative methods*, that is iterative methods that only consider one component of the current gradient in each iteration, we need to define several additional parameters characterizing f. Let $\vec{e}_i \in \mathbb{R}^n$ denote the standard basis vector for coordinate i, $f_i(\vec{x}) \in \mathbb{R}^n$ denote the partial derivative of f at $\vec{x}$ along $\vec{e}_i$, i.e. $f_i(\vec{x}) \stackrel{\text{def}}{=} \vec{e}_i^T \nabla f(\vec{x})$, and $\vec{f}_i(\vec{x})$ denote the corresponding vector, i.e $\vec{f}_i(\vec{x}) \stackrel{\text{def}}{=} f_i \cdot \vec{e}_i$. We say f *has component-wise Lipschitz continuous gradient with Lipschitz constants* $\{L_i\}$ if

$$\forall \vec{x} \in \mathbb{R}^n, \ \forall t \in \mathbb{R}, \ \forall i \in [n] : |f_i(\vec{x} + t \cdot \vec{e}_i) - f_i(\vec{x})| \leq L_i \cdot |t|.$$

Let $S_\alpha \stackrel{\text{def}}{=} \sum_{i=1}^n L_i^\alpha$ denote the total component-wise Lipschitz constant . Later we will see that S_α has a similar role for coordinate descent as L has for gradient descent.

We give two examples for convex functions induced by linear systems and calculate their parameters. Note that even though one example can be deduced from the other, we provide both as it allows us to introduce more notation.

Example 1. *Let* $f(\vec{x}) \stackrel{\text{def}}{=} \frac{1}{2} \langle \mathbf{A}\vec{x}, \vec{x} \rangle - \langle \vec{x}, \vec{b} \rangle$ *for symmetric positive definite matrix* $\mathbf{A} \in \mathbb{R}^{n \times n}$*. Since* $\mathbf{A} = \mathbf{A}^T$ *clearly* $\nabla f(\vec{x}) = \mathbf{A}\vec{x} - \vec{b}$ *and* $\nabla^2 f(\vec{x}) = \mathbf{A}$*. Therefore, by Lemma 2,* L *and* σ *satisfy* $\sigma \|\vec{x}\|^2 \leq \vec{x}^T \mathbf{A} \vec{x} \leq L \|\vec{x}\|^2$*. Consequently,* σ *is the smallest eigenvalue* $\lambda_{\min}$ *of* $\mathbf{A}$ *and* L *is the largest eigenvalue* $\lambda_{\max}$ *of* $\mathbf{A}$*. Furthermore,* $\forall i \in [n]$ *we see that* $f_i(\vec{x}) = \vec{e}_i^T \left(\mathbf{A}\vec{x} - \vec{b} \right)$ *and therefore* L_i *satisfies*

$$\forall t \in \mathbb{R} \quad : \quad |t| \cdot |\mathbf{A}_{ii}| = |\vec{e}_i^T \mathbf{A} (t e_i)| \leq L_i |t| \quad .$$

Since the positive definiteness of $\mathbf{A}$ *implies that* $\mathbf{A}$ *is positive on diagonal, we have* $L_i = \mathbf{A}_{ii}$*, and consequently* $S_1 = \text{tr}(\mathbf{A}) = \sum \mathbf{A}_{ii} = \sum \lambda_i$ *where* λ_i *are eigenvalues of* $\mathbf{A}$*.*

Example 2. *Let* $f(\vec{x}) = \frac{1}{2} \|\mathbf{A}\vec{x} - b\|^2$ *for any matrix* $\mathbf{A}$*. Then* $\nabla f(\vec{x}) = \mathbf{A}^T \left(\mathbf{A}\vec{x} - \vec{b} \right)$ *and* $\nabla^2 f(\vec{x}) = \mathbf{A}^T \mathbf{A}$*. Hence,* σ *and* L *satisfy* $\sigma \|\vec{x}\|^2 \leq \vec{x}^T \mathbf{A}^T \mathbf{A} \vec{x} \leq L \|\vec{x}\|^2$*. And we see that* σ *is the the smallest eigenvalue* $\lambda_{\min}$ *of* $\mathbf{A}^T \mathbf{A}$ *and* L *is the largest eigenvalue* $\lambda_{\max}$ *of* $\mathbf{A}^T \mathbf{A}$*. As in the previous example, we therefore have* $L_i = \|a_i\|^2$ *where* a_i *is the* i*-th column of* $\mathbf{A}$ *and* $S_1 = \sum \|a_i\|^2 = \|\mathbf{A}\|_F^2$*, the Frobenius norm of* $\mathbf{A}$*.*

III. Review of Previous Iterative Methods

In this section, we briefly review several iterative first-order methods for smooth convex minimization. This overview is by no means all-inclusive, our goal is simply to familiarize the reader with numerical techniques we will make use of later and motivate our presentation of the accelerated coordinate descent method. For a comprehensive review, there are multiple good references, e.g. [12], [26].

A. Gradient Descent

Given an initial point $\vec{x}_0$ and step sizes h_k, the *gradient descent method* applies the following iterative update rule:

$$\forall k \geq 0 \quad : \quad \vec{x}_{k+1} := \vec{x}_k - h_k \nabla f(\vec{x}_k).$$

For $h_k = \frac{1}{L}$, this method simply chooses the minimum point of the upper envelope of f at $\vec{x}_k$:

$$\vec{x}_{k+1} = \arg\min_{\vec{y}} f(\vec{x}_k) + \langle \nabla f(\vec{x}_k), \vec{y} - \vec{x}_k \rangle + \frac{L}{2} \|\vec{y} - \vec{x}_k\|^2 .$$

Thus, we see that the gradient descent method is a greedy method that chooses the minimum point based on the worst case estimate of the function based on the value of $f(x_k)$ and $\nabla f(x_k)$. It is well known that it provides the following guarantee [12, Cor 2.1.2, Thm 2.1.15]

$$f(\vec{x}_k) - f^* \leq \frac{L}{2} \left(1 - \frac{\sigma}{L}\right)^k \|\vec{x}_0 - \vec{x}^*\|^2 \quad . \tag{3}$$

B. Accelerated Gradient Descent

To speed up the greedy and memory-less gradient descent method, Nesterov [14], [12] suggested to use a quadratic function to estimate the function. Formally, we define an *estimate sequence* as follows: [1]

Definition 1 (Estimate Sequence)**.** *A triple of sequences* $\{\phi_k(x), \eta_k, \vec{x}_k\}_{k=0}^{\infty}$ *is called an* estimate sequence *of* f *if* $\lim_{k\to\infty} \eta_k = 0$ *and for any* $\vec{x} \in \mathbb{R}^n$ *and* $k \geq 0$ *we have*

$$\phi_k(\vec{x}) \leq (1 - \eta_k) f(\vec{x}) + \eta_k \phi_0(\vec{x}) \tag{4}$$

and

$$f(\vec{x}_k) \leq \min_{\vec{x} \in \mathbb{R}^n} \phi_k(\vec{x}). \tag{5}$$

An estimate sequence of f is an approximate lower bound of f which is slightly above f^*. This relaxed definition allows us to find a better approximation of f instead of relying on the worst case upper envelope at each step.

A good estimate sequence gives an efficient algorithm [12, Lem 2.2.1] by the following

$$\lim_{k\to\infty} f(\vec{x}_k) - f^* \leq \lim_{k\to\infty} \eta_k \left(\phi_0(\vec{x}^*) - f^*\right) = 0.$$

Since an estimate sequence is an approximate lower bound, a natural computable candidate is to use the convex combination of lower envelopes of f at some points.

[1] Note that our definition deviates slightly from Nesterov's [12, Def 2.2.1] in that we include condition 5.

Since it can be shown that any convex combinations of lower envelopes at evaluation points $\{y_k\}$ satisfies (4) under some mild condition, additional points $\{y_k\}$ other than $\{x_k\}$ can be used to tune the algorithm. Nesterov's *accelerated gradient descent method* can be obtained by tuning the the free parameters $\{y_k\}$ and $\{\eta_k\}$ to satisfy (5). Among all first order methods, this method is optimal up to constants in terms of number of queries made to f and ∇f. The performance of the accelerated gradient descent method can be characterized as follows: [12]

$$f(\vec{x}_k) - f^* \leq L \left(1 - \sqrt{\frac{\sigma}{L}}\right)^k \|\vec{x}_0 - \vec{x}^*\|^2 \quad . \tag{6}$$

C. Coordinate Descent

The *coordinate descent method* of Nesterov [7] is a variant of gradient descent in which only one coordinate of the current iterate is updated at a time. For a fixed $\alpha \in \mathbb{R}$, each iteration k of coordinate descent consists of picking a random a random coordinate $i_k \in [n]$ where

$$\Pr[i_k = j] = P_\alpha(j) \quad \text{where} \quad P_\alpha(j) \stackrel{\text{def}}{=} L_{i_k}^\alpha / S_\alpha$$

and then performing a descent step on that coordinate:

$$\vec{x}_{k+1} := \vec{x}_k - \frac{1}{L_{i_k}} \vec{f}_{i_k}(\vec{x}_k).$$

To analyze this algorithm's convergence rate, we define the norm $\|\vec{x}\|_{1-\alpha} \stackrel{\text{def}}{=} \sqrt{\sum L_i^{1-\alpha} \vec{x}_i^2}$, its dual $\|\vec{x}\|_{1-\alpha}^* \stackrel{\text{def}}{=} \|\vec{x}\|_{\alpha-1}$ and the inner product $\langle \vec{x}, \vec{y} \rangle_{1-\alpha} \stackrel{\text{def}}{=} \sum L_i^{1-\alpha} \vec{x}_i \vec{y}_i$ and we let $\sigma_{1-\alpha}$ denote the convexity parameter of f with respect to $\|\cdot\|_{1-\alpha}$. Using the definition of coordinate-wise Lipschitz constant, each step can be shown to have the following guarantee on expected improvement [7]

$$f(\vec{x}_k) - \mathbb{E}\left[f(\vec{x}_{k+1})\right] \geq \frac{1}{2S_\alpha} \left(\|\nabla f(\vec{x}_k)\|_{1-\alpha}^*\right)^2 .$$

and further analysis shows the following convergence guarantee coordinate descent [7]

$$\mathbb{E}\left[f(\vec{x}_k)\right] - f^* \leq \left(1 - \frac{\sigma_{1-\alpha}}{S_\alpha}\right)^k \left(f(\vec{x}_0) - f^*\right).$$

IV. General Accelerated Coordinate Descent

In this section, we present our general and iteration-efficient *accelerated coordinate descent method (ACDM)*. In particular, we show how to improve the asymptotic convergence rate of any coordinate descent based algorithm without paying asymptotic cost. We remark that the bulk of the credit for conceiving of such a method belongs to Nesterov [7] who provided a different proof of convergence for such a method for the $\alpha = 0$ case, however we note that changes to the algorithm were necessary to deal with the $\alpha = 1$ case used in all of our applications.

A. ACDM by Probabilistic Estimate Sequences

Following the spirit of the estimate sequence proof of accelerated gradient descent [12], here we present a proof of ACDM convergence through what we call a *(probabilistic) estimation sequence*.

Definition 2 ((Probabilistic) Estimate Sequence)**.** *A triple of sequences* $\{\phi_k(\vec{x}), \eta_k, \vec{x}_k\}_{k=0}^{\infty}$ *where* $\phi_k : \mathbb{R}^n \to \mathbb{R}$ *and* $\vec{x}_k \in \mathbb{R}^n$ *are chosen according to some probability distribution is called a* (probabilistic) estimate sequence of f *if* $\lim_{k\to 0} \eta_k = 0$ *and for all* $k \geq 0$ *we have*

$$\mathbb{E}\left[\phi_k(\vec{x})\right] \leq (1-\eta_k) f(\vec{x}) + \eta_k \mathbb{E}\left[\phi_0(\vec{x})\right], \tag{7}$$

$$\mathbb{E}\left[f(\vec{x}_k)\right] \leq \min_{\vec{x}\in\mathbb{R}^n} \mathbb{E}\left[\phi_k(\vec{x})\right] \tag{8}$$

A probabilistic estimation sequence gives a randomized minimization method due to the following

$$\lim_{k\to\infty} \mathbb{E}\left[f(\vec{x}_k)\right] - f^* \leq \lim_{k\to\infty} \eta_k \left(\mathbb{E}\phi_0(x^*) - f^*\right) = 0 \ .$$

Since a probabilistic estimation sequence can be constructed using random partial derivatives, rather than a full gradient computations, there is hope that probabilistic estimation sequences require less information for fast convergence and therefore outperform their deterministic counterparts.

Similar to the accelerated gradient descent method, in the following lemma we first show how to combine a sequence of lower envelopes to satisfy condition (7) and prove that it preserves a particular structure on the current lower bound.

Lemma 3 ((Probabilistic) Estimate Sequence Construction)**.** *Let* $\phi_0(\vec{x})$, $\{\vec{y}_k, \theta_k, i_k\}_{k=0}^{\infty}$ *be such that*

- $\phi_0 : \mathbb{R}^n \to \mathbb{R}$ *is an arbitrary function. Each* $\vec{y}_k \in \mathbb{R}^n$.
- *Each* $\theta_k \in (0,1)$ *and* $\sum_{k=0}^{\infty} \theta_k = \infty$.
- *Each* i_k *is chosen randomly so that* $\Pr[i_k = i] = \frac{L_i^\alpha}{S_\alpha}$.

Then the pair of sequences $\{\phi_k(\vec{x}), \eta_k\}_{k=0}^{\infty}$ *defined by*

- $\eta_0 = 1$ *and* $\eta_{k+1} = (1-\theta_k)\eta_k$
- $\phi_{k+1}(\vec{x}) = (1-\theta_k)\phi_k(\vec{x}) + \theta_k f(\vec{y}_k) + \theta_k \frac{S_\alpha}{L_{i_k}} \langle \vec{f}_{i_k}(\vec{y}_k), \vec{x} - \vec{y}_k \rangle_{1-\alpha} + \theta_k \frac{\sigma_{1-\alpha}}{2} \left\|\vec{x} - \vec{y}_k\right\|_{1-\alpha}^2$

satisfies condition (7). Furthermore, if $\phi_0(\vec{x}) = \phi_0^* + \frac{\zeta_0}{2}\left\|\vec{x} - \vec{v}_0\right\|_{1-\alpha}^2$, *then this process produces a sequence of quadratic functions of the form* $\phi_k(\vec{x}) = \phi_k^* + \frac{\zeta_k}{2}\left\|\vec{x} - \vec{v}_k\right\|_{1-\alpha}^2$ *where*

$$\zeta_{k+1} = (1-\theta_k)\zeta_k + \theta_k \sigma_{1-\alpha} \tag{9}$$

$$v_{k+1} = \frac{(1-\theta_k)\zeta_k \vec{v}_k}{\zeta_{k+1}} + \frac{\theta_k \sigma_{1-\alpha} \vec{y}_k}{\zeta_{k+1}} - \frac{S_\alpha \theta_k}{\zeta_{k+1} L_{i_k}} \vec{f}_{i_k}(\vec{y}_k) \tag{10}$$

$$\begin{aligned}\phi_{k+1}^* = {} & (1-\theta_k)\phi_k^* + \theta_k f(\vec{y}_k) - \frac{\theta_k^2 S_\alpha^2}{2\zeta_{k+1}} \frac{\left(f_{i_k}(\vec{y}_k)\right)^2}{L_{i_k}^{1+\alpha}} \\ & + \frac{\theta_k(1-\theta_k)\zeta_k}{\zeta_{k+1}} \frac{\sigma_{1-\alpha}}{2} \left\|\vec{y}_k - \vec{v}_k\right\|_{1-\alpha}^2 \\ & + \frac{\theta_k(1-\theta_k)\zeta_k}{\zeta_{k+1}} \frac{S_\alpha}{L_{i_k}} \langle \vec{f}_{i_k}(\vec{y}_k), \vec{v}_k - \vec{y}_k \rangle_{1-\alpha}.\end{aligned}$$

Proof: The proof follows from direct calculations. ■

In the following theorem, we show how to choose $\vec{x}$, $\vec{y}$ and θ to satisfy the condition (8) and thereby derive a simple form of the general accelerated coordinate descent method. We also show that the number of iterations required for ACDM is $\tilde{O}\left(\sqrt{\frac{S_\alpha n}{\sigma_{1-\alpha}}}\right)$ which is strictly better than the number of iterations required for coordinate descent method, $\tilde{O}\left(\frac{S_\alpha}{\sigma_{1-\alpha}}\right)$. Later in Theorem 12, we show that this is optimal up to constant for the type of algorithm considered. Note that while several of the definitions specifications in the following theorem statement may at first glance seem unnatural our proof will show that they are nearly forced in order to achieve certain algorithm design goals.

Theorem 4 (Simple ACDM)**.** *For all* $i \in [n]$ *let* $\tilde{L}_i^\alpha \stackrel{\text{def}}{=} \max(L_i^\alpha, S_\alpha/n)$ *and let* $\tilde{S}_\alpha \stackrel{\text{def}}{=} \sum_{i=1}^n \tilde{L}_i^\alpha$. *Furthermore, for any* $\vec{x}_0 \in \mathbb{R}^n$ *and for all* $k \geq 0$, *let* $\vec{y}_0 = \vec{x}_0$,

$$\phi_0(\vec{x}) = f(\vec{x}_0) + \frac{\sigma_{1-\alpha}}{2}\left\|\vec{x} - \vec{x}_0\right\|_{1-\alpha}^2, \ \theta_k = \sqrt{\frac{\sigma_{1-\alpha}}{2\tilde{S}_\alpha n}}$$

Then applying Lemma 3 with these parameters, $\tilde{L}_i$ *as the coordinate-wise gradient Lipshitz constants, and choosing* $\vec{y}_k$ *and* $\vec{x}_k$ *such that*

$$\frac{\theta_k \zeta_k}{\zeta_{k+1}}(\vec{v}_k - \vec{y}_k) + \vec{x}_k - \vec{y}_k = 0, \ \vec{x}_k = \vec{y}_{k-1} - \frac{1}{\tilde{L}_{i_{k-1}}} \vec{f}_{i_k}(\vec{y}_{k-1})$$

yields a probabilistic estimate sequence. This accelerated coordinate descent method *satisfies*

$$\mathbb{E}\left[f(\vec{x}_k)\right] - f^* \leq \left(1 - \frac{1}{2}\sqrt{\frac{\sigma_{1-\alpha}}{S_\alpha n}}\right)^k \left(\phi_0(\vec{x}^*) - f^*\right). \tag{11}$$

Proof: By construction we know that condition (7) holds. It remains to show that $\vec{x}_k$ satisfies condition (8) and analyze the convergence rate. To prove (8), we proceed by induction to prove that $\mathbb{E}_k\left[f(\vec{x}_k)\right] \leq \mathbb{E}_k\left[\min_{\vec{x}\in\mathbb{R}^n} \phi_k(\vec{x})\right] = \mathbb{E}_k\left[\phi_k^*\right]$ where $\mathbb{E}_k$ indicates the expectation up to iteration k. The base case $f(\vec{x}_0) \leq \phi_0(\vec{x}_0)$ is trivial and we proceed by induction assuming that $\mathbb{E}_k\left[f(\vec{x}_k)\right] \leq \mathbb{E}_k\left[\phi_k^*\right]$. By Lemma (3) and the inductive hypothesis we get

$$\begin{aligned}\mathbb{E}_k\left[\phi_{k+1}^*\right] \geq \mathbb{E}_k \Bigg[& (1-\theta_k) f(\vec{x}_k) + \theta_k f(\vec{y}_k) - \frac{\theta_k^2 \tilde{S}_\alpha^2}{2\zeta_{k+1}} \frac{\left(f_{i_k}(\vec{y}_k)\right)^2}{\tilde{L}_{i_k}^{1+\alpha}} \\ & + \frac{\theta_k(1-\theta_k)\zeta_k}{\zeta_{k+1}} \left(\frac{\sigma_{1-\alpha}}{2}\left\|\vec{y}_k - \vec{v}_k\right\|_{1-\alpha}^2 + \frac{\tilde{S}_\alpha}{\tilde{L}_{i_k}} \langle \vec{f}_{i_k}(\vec{y}_k), \vec{v}_k - \vec{y}_k \rangle_{1-\alpha} \right) \Bigg]\end{aligned}$$

where for notational convenience we drop the expectation in each of the variables. By convexity $f(\vec{x}_k) \geq f(\vec{y}_k) + \langle \nabla f(\vec{y}_k), \vec{x}_k - \vec{y}_k \rangle$ so applying this and the definitions of $\left\|\cdot\right\|_{1-\alpha}$ and $\langle \cdot, \cdot \rangle_{1-\alpha}$ we get

$$\begin{aligned}\phi_{k+1}^* \geq {} & f(\vec{y}_k) - \frac{\theta_k^2 \tilde{S}_\alpha^2 f_{i_k}(\vec{y}_k)^2}{2\zeta_{k+1}\tilde{L}_{i_k}^{1+\alpha}} \\ & + (1-\theta_k)\left(\frac{\tilde{S}_\alpha}{\tilde{L}_{i_k}^\alpha} \left\langle \vec{f}_{i_k}(\vec{y}_k), \frac{\theta_k \zeta_k}{\zeta_{k+1}}(\vec{v}_k - \vec{y}_k) \right\rangle + \langle \nabla f(\vec{y}_k), \vec{x}_k - \vec{y}_k \rangle \right)\end{aligned}$$

Using that $\forall i \in [n]$ we have $\Pr[i_k = i] = \frac{\tilde{L}_i^\alpha}{\tilde{S}_\alpha}$ we get

$$\mathbb{E}_{k+1}\left[\phi_{k+1}^*\right] \geq \mathbb{E}_{k+1}\left[f(\vec{y}_k) - \frac{\theta_k^2 \tilde{S}_\alpha^2}{2\zeta_{k+1}} \frac{(f_{i_k}(\vec{y}_k))^2}{\tilde{L}_{i_k}^{1+\alpha}}\right] + (1-\theta_k)\left\langle \nabla f(\vec{y}_k), \frac{\theta_k \zeta_k}{\zeta_{k+1}}(\vec{v}_k - \vec{y}_k) + \vec{x}_k - \vec{y}_k \right\rangle .$$

From this formula we see that $\vec{y}_k$ was chosen specifically to cancel the second term so that

$$\mathbb{E}_{k+1}\left[\phi_{k+1}^*\right] \geq \sum_{i=1}^{n}\left[f(\vec{y}_k) - \frac{\theta_k^2 \tilde{S}_\alpha}{2\zeta_{k+1}} \frac{f_i(\vec{y}_k)^2}{\tilde{L}_i}\right]$$

and it simply remains to choose θ_k and $\vec{x}_{k+1}$ so that $\mathbb{E}_{k+1}[f(\vec{x}_{k+1})]$ is smaller than this quantity.

To meet condition (8), we simply need to choose $\{\theta_k\}_{k=0}^\infty$ so $\frac{\theta_k^2 \tilde{S}_\alpha}{\zeta_{k+1}} = \frac{1}{2n}$. Using $\tilde{L}_i^\alpha \geq \frac{S_\alpha}{n} \geq \frac{\tilde{S}_\alpha}{2n}$, we have

$$\mathbb{E}_{k+1}\left[\phi_{k+1}^*\right] \geq \sum_{i=1}^{n}\left[f(\vec{y}_k) - \frac{1}{2\tilde{S}_\alpha} \frac{f_i(\vec{y}_k)^2}{\tilde{L}_i^{1-\alpha}}\right]$$

To compute $\vec{x}_{k+1}$, we use the fact that applying Lemma 1 to the formula $f(\vec{y}_k - t\vec{f}_i(\vec{y}_k))$ yields

$$f\left(\vec{y}_k - \frac{1}{\tilde{L}_i}\vec{f}_i(\vec{y}_k)\right) \leq f(\vec{y}_k) - \frac{f_i(\vec{y}_k)^2}{2\tilde{L}_i}$$

and therefore for $\vec{x}_{k+1}$ as defined we have

$$\mathbb{E}_{k+1}[f(\vec{x}_{k+1})] \leq \sum_{i=1}^{n}\left[f(\vec{y}_k) - \frac{f_i(\vec{x})^2}{2\tilde{S}_\alpha \tilde{L}_i^{1-\alpha}}\right] \leq \mathbb{E}_{k+1}\left[\phi_{k+1}^*\right] .$$

Recalling that $\zeta_{k+1} = (1-\theta_k)\zeta_k + \theta_k \sigma_{1-\alpha}$ we see that choosing $\zeta_0 = \sigma_{1-\alpha}$ implies $\zeta_k = \sigma_{1-\alpha}$ for all k and therefore choosing $\theta_k = \sqrt{\frac{\sigma_{1-\alpha}}{2\tilde{S}_\alpha n}}$ completes the proof that the chosen parameters produce a probabilistic estimate sequence. Furthermore, we see that this choice implies that $\eta_k = \left(1 - \sqrt{\frac{\sigma_{1-\alpha}}{2\tilde{S}_\alpha n}}\right)^k$. Therefore, by the definition of a probabilistic estimate sequence and the fact that $\tilde{S}_\alpha \geq 2S_\alpha$, equation (11) follows. ■

B. Numerical Stability

In the previous section, we provided a simple proof how to achieve an ACDM with a convergence rate of $\tilde{O}(\sqrt{\frac{S_\alpha n}{\sigma_{1-\alpha}}})$. While sufficient for many purposes, the algorithm does not achieve the ideal dependence on initial error. For consistency with [7] we perform the change of variables

$$\alpha_k \stackrel{\text{def}}{=} \frac{\theta_k \zeta_k}{\zeta_k + \theta_k \sigma_{1-\alpha}}, \ \beta_k \stackrel{\text{def}}{=} \frac{(1-\theta_k)\zeta_k}{\zeta_{k+1}}, \ \gamma_k \stackrel{\text{def}}{=} \frac{\tilde{S}_\alpha \theta_k}{\zeta_{k+1}}$$

and by better tuning θ_k we derive the following algorithm.

Accelerated Coordinate Descent Method
1. Define $\tilde{L}_i = \max(L_i, (S_\alpha/n)^{1/\alpha})$ and $\tilde{S}_\alpha = \sum \tilde{L}_i^\alpha$
2. Define $\vec{v}_0 = \vec{x}_0, a_0 = \frac{1}{2n}, b_0 = 2$
3. For $k \geq 0$ iterate:
3a. Find $\alpha_k, \beta_k, \gamma_k \geq \frac{1}{2n}$ such that $\gamma_k^2 - \frac{\gamma_k}{2n} = \left(1 - \frac{\gamma_k \sigma_{1-\alpha}}{\tilde{S}_\alpha}\right)\frac{a_k^2}{b_k^2} = \beta_k \frac{a_k^2}{b_k^2} = \frac{\beta_k \gamma_k}{2n}\frac{1-\alpha_k}{\alpha_k}$.
3b. $\vec{y}_k = \alpha_k \vec{v}_k + (1-\alpha_k)\vec{x}_k$.
3c. Choose i_k according to $P_\alpha(i) = \tilde{L}_i^\alpha / \tilde{S}_\alpha$.
3d. $\vec{x}_{k+1} = \vec{y}_k - \frac{1}{\tilde{L}_{i_k}}\vec{f}_{i_k}(\vec{y}_k)$. $v_{k+1} = \beta_k \vec{v}_k + (1-\beta_k)\vec{y}_k - \frac{\gamma_k}{\tilde{L}_{i_k}}\vec{f}_{i_k}(\vec{y}_k)$.
3e. $b_{k+1} = \frac{b_k}{\sqrt{\beta_k}}$ and $a_{k+1} = \gamma_k b_{k+1}$.

Theorem 5 (Numerical Stability of ACDM)**.** *Suppose that in each iteration of ACDM step 3d has additive error ϵ, i.e. there exists $\vec{\epsilon}_{1,k}, \vec{\epsilon}_{2,k} \in \mathbb{R}^n$ with $\|\vec{\epsilon}_{1,k}\|_{1-\alpha} \leq \epsilon$ and $\|\vec{\epsilon}_{2,k}\|_{1-\alpha} \leq \epsilon$ such that step 3 is*

$$\vec{x}_{k+1} = \vec{y}_k - \frac{1}{\tilde{L}_{i_k}}\vec{f}_{i_k}(\vec{y}_k) + \vec{\epsilon}_{1,k}$$
$$\vec{v}_{k+1} = \beta_k \vec{v}_k + (1-\beta_k)\vec{y}_k - \frac{\gamma_k}{\tilde{L}_{i_k}}\vec{f}_{i_k}(\vec{y}_k) + \vec{\epsilon}_{2,k}.$$

If $\epsilon < \frac{\sigma_{1-\alpha}^2}{8\tilde{S}_\alpha^2 n}$ and $k \geq \sqrt{\frac{2\tilde{S}_\alpha n}{\sigma_{1-\alpha}}}$, then we have the convergence guarantee

$$\sigma_{1-\alpha}\mathbb{E}\left[\|\vec{v}_{k+1} - \vec{x}^*\|_{1-\alpha}^2\right] + (\mathbb{E}[f(\vec{x}_{k+1})] - f^*) \leq \delta_k \quad (12)$$

where $\delta_k \stackrel{\text{def}}{=} 32\sigma_{1-\alpha}\left(1 - \frac{1}{5}\sqrt{\frac{\sigma_{1-\alpha}}{S_\alpha n}}\right)^k ||x_0 - x^*||_{1-\alpha}^2 + 32\sigma_{1-\alpha}\left(1 - \frac{1}{5}\sqrt{\frac{\sigma_{1-\alpha}}{S_\alpha n}}\right)^k \frac{1}{S_\alpha^2}(f(x_0) - f^*) + 24kS_\alpha\varepsilon^2$ *and the additional convergence guarantee that* $\frac{1}{k}\sum_{j=k}^{2k-1}\|\nabla f(\vec{y}_k)\|_{1-\alpha}^2 \leq 2000\frac{\tilde{S}_\alpha}{n}\delta_k$.

Proof: In the full paper, we give a proof of convergence of this in the unit-cost RAM model by studying the following potential function [7] for suitable constants $a_k, b_k \in \mathbb{R}$,

$$a_k(\mathbb{E}[f(\vec{x}_k)] - f^*) + b_k\mathbb{E}\left[\|\vec{v}_k - x_*\|_{1-\alpha}^2\right] .$$

■

This theorem provides useful estimates for the error $f(\vec{x}_{k+1}) - f^*$, the residual $\|\vec{v}_{k+1} - \vec{x}^*\|_{1-\alpha}^2$, and the norm of gradient $\|\nabla f(\vec{y}_k)\|_{1-\alpha}^*$. Note how the estimate depends on the initial error $f(\vec{x}_0) - f^*$ mildly as compared to Theorem 4. We use this fact in creating an efficient SDD solver in Section V-C.

C. Efficient Iteration

In both Nesterov's paper [7] and later work [21] the original ACDM proposed by Nesterov was not recommend since a naive implementation takes $O(n)$ time to update the vector $\vec{v}_k$, and therefore is likely slower than accelerated gradient descent. However, if we make the mild assumption that we can compute $\nabla f(t\vec{x} + s\vec{y})$ for $s, t \in \mathbb{R}$ and $\vec{y}, \vec{y} \in \mathbb{R}$ in the same asymptotic runtime as it takes to compute $\nabla f(\vec{x})$

(i.e. we do not need to compute the sum explicitly), then we can implement ACDM without additional asymptotic computational costs per iteration as compared to the cost of the coordinate descent method performing an update on the given coordinate.

Lemma 6 (Efficient ACDM Iteration)**.** *For $S_\alpha = O(\mathsf{poly}(n))$ and $\sigma_{1-\alpha} = \Omega(\mathsf{poly}(\frac{1}{n}))$ each iteration of ACDM can be implemented in $O(1)$ time plus the time to make one oracle call of the form $\vec{f}_{i_k}(t\vec{x} + s\vec{y})$ for $s, t \in \mathbb{R}$ and $\vec{x}, \vec{y} \in \mathbb{R}^n$, using at most an additional $O(\log n)$ bits of precision so long as the number of iterations of ACDM is $O\left(\sqrt{\frac{S_\alpha n}{\sigma_{1-\alpha}}}\log(n)\right)$.*

Proof: In the full paper, we give a lazy update scheme to achieve same iteration and prove its stability. ■

V. Faster Linear System Solvers

In this section, we show how ACDM can be used to achieve asymptotic runtimes that outperform various state-of-the-art methods for solving linear systems in a variety of settings.

A. Comparison to Conjugate Gradient Method

Here we compare the performance of ACDM to conjugate gradient (CG) and show that under mild assumptions about the linear system being solved ACDM achieves a better asymptotic running time.

For symmetric positive definite (SPD) matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ and vector $\vec{b} \in \mathbb{R}^n$, we solve the linear system of equations $\mathbf{A}\vec{x} = \vec{b}$ via the following equivalence unconstrained quadratic minimization problem:

$$\min_{\vec{x}\in\mathbb{R}^n} f(\vec{x}) \stackrel{\text{def}}{=} \frac{1}{2}\langle \mathbf{A}\vec{x}, \vec{x}\rangle - \langle \vec{b}, \vec{x}\rangle \ . \tag{13}$$

Let m denote the number of nonzero entries in $\mathbf{A}$, let nnz_i denote the number of nonzero entries in the ith row of $\mathbf{A}$. To make the analysis simpler, we assume that the nonzero entries is somewhat uniform, namely $\mathrm{nnz}_i = O(\frac{m}{n})$. This assumption is trivially met for dense matrices, finite difference matrices, etc. Letting $0 \le \lambda_1 \le \ldots \le \lambda_n$ denote the eigenvalues of $\mathbf{A}$, we get the following:

Theorem 7 (ACDM on SPD Systems)**.** *Assume $\mathbf{A}$ is a SPD matrix with $\mathrm{nnz}_i = O(\frac{m}{n})$. Let the numerical rank $r(\mathbf{A}) = \sum_{i=1}^n \lambda_i/\lambda_n$. ACDM applied to* (13) *with $\alpha = 1$ produces an approximate solution in $\tilde{O}\left(m\sqrt{\frac{r(\mathbf{A})}{n}}\sqrt{\frac{\lambda_n}{\lambda_1}}\log\frac{1}{\epsilon}\right)$ time with ϵ error in $\mathbf{A}$ norm in expectation.*

Proof: The running time of ACDM with $\alpha = 1$ depends on σ_0 and S_1. From Example 1, the total component-wise Lipschitz constant S_1 is the trace of A, which is $\sum \lambda_i$ and the convexity parameter σ_0 is λ_1, therefore by Theorem 5 the convergence rate of ACDM is $\sqrt{\frac{\lambda_1}{n\sum_{i=1}^n \lambda_i}}$ as desired. Furthermore, the running time of each step depends on the running time of the oracle, i.e. computing $f_i(x) = (Ax)_i$, which by our assumption on nnz_i takes time $O\left(\frac{m}{n}\right)$. ■

To compare with conjugate gradient, we know that one crude bound for the rate of convergence of conjugate gradient is $O\left(\sqrt{\frac{\lambda_1}{\lambda_n}}\right)$. Hence the total running time of CG to produce an epsilon approximate solution is $\tilde{O}\left(m\sqrt{\frac{\lambda_n}{\lambda_1}}\log\frac{1}{\epsilon}\right)$. Therefore, with this bound ACDM is always faster or matches the running time since the numerical rank of $\mathbf{A}$ is always less than or equals to n. Thus, we see that when the numerical rank of $\mathbf{A}$ is $o(n)$, ACDM will likely [2] have a faster asymptotic running time.

To be more fair in our comparison to conjugate gradient, we note that in [27] tighter bound on the performance of CG was derived and they showed that in fact CG has a running time of $\tilde{O}\left(\sum \mathrm{nnz}_i\left(\frac{\sum_{i=1}^n \lambda_i}{\lambda_1}\right)^{1/3}\right)$ implying that ACDM is faster than CG when $\sum_{i=1}^n \lambda_i \le n^3\lambda_1$ and it is usually satisfied. In the extreme cases that the condition is false, CG will need to run for $O(n)$ iterations at which point an exact answer could be computed.

B. Accelerating Randomized Kaczmarz

The Kaczmarz method [1] is an iterative algorithm to solve $\mathbf{A}\vec{x} = b$ for any full row rank matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$. Letting $\vec{a}_i \in \mathbb{R}^n$ denote the i-th row of the matrix $\mathbf{A}$, we know that the solution of $\mathbf{A}\vec{x} = \vec{b}$ is the intersection of the hyperplanes $H_i \stackrel{\text{def}}{=} \{x : \langle \vec{a}_i, \vec{x}_k\rangle = \vec{b}_i\}$. The Kaczmarz method simply iteratively picks one of these hyperplanes and projects onto it by the following formula:

$$x_{k+1} = \mathrm{proj}_{H_{i_k}}(\vec{x}_k) \stackrel{\text{def}}{=} \frac{\vec{b}_{i_k} - \langle a_{i_k}, \vec{x}_k\rangle}{\|a_{i_k}\|^2} a_{i_k} \ .$$

There are many schemes that can be chosen to pick the hyperplane i_k, many of which are difficult to analyze and compare, but in a breakthrough result, Strohmer and Vershynin in 2008 analyzed the randomized schemes which sample the hyperplane with probability proportional to $\|a_i\|_2^2$. They proved the following

Theorem 8 (Strohmer and Vershynin [6])**.** *The Kaczmarz method samples row i with probability proportionally to $\|a_i\|_2^2$ at each iteration and yields the following*

$$\forall k \ge 0 \ : \ \mathbb{E}\left[\|\vec{x}_k - \vec{x}^*\|_2^2\right] \le (1 - \kappa(\mathbf{A})^{-2})^k \|\vec{x}_0 - \vec{x}^*\|^2$$

where $\vec{x}^*$ is such that $\mathbf{A}\vec{x}^* = \vec{b}$, $\kappa(\mathbf{A}) \stackrel{\text{def}}{=} \left\|\mathbf{A}^{-1}\right\|_2 \cdot \|\mathbf{A}\|_F$ is the relative condition number *of $\mathbf{A}$, $\mathbf{A}^{-1}$ is the left inverse of $\mathbf{A}$, $\left\|\mathbf{A}^{-1}\right\|_2$ is the smallest non-zero spectral value of $\mathbf{A}$ and $\|\mathbf{A}\|_F^2 \stackrel{\text{def}}{=} \sum a_{ij}^2$ is the Frobenius norm of $\mathbf{A}$.*

Here we show show to cast this algorithm as an instance of coordinate descent and obtain an improved convergence

[2] The running time of CG may be asymptotic faster when the eigenvalues form clusters.

rate by applying ACDM. We remark that accelerated Kaczmarz will sample rows with a slightly different probability distribution. As long as this does not increase the expected computational cost of an iteration, it will yield an algorithm with a faster asymptotic running time.

Theorem 9 (Accelerated Kaczmarz)**.** *The ACDM method samples row i with probability proportionally to* $\max\left\{\|a_i\|_2^2, \frac{\|\mathbf{A}\|_F^2}{m}\right\}$ *and performs extra $O(1)$ work at each iteration. It yields the following*

$$\forall k \geq 0 \;\; : \;\; \mathbb{E}\left\|\vec{x}_k - \vec{x}^*\right\|_2^2 \leq 3\left(1 - \frac{\kappa(\mathbf{A})^{-1}}{2\sqrt{m}}\right)^k \left\|\vec{x}_0 - \vec{x}^*\right\|^2.$$

Proof: To cast Strohmer and Vershynin's randomized Kaczmarz algorithm in the framework of coordinate descent, we consider minimizing the objective function of theorem theorem directly, i.e. $\min_{\vec{x}\in\mathbb{R}^n} \frac{1}{2}\left\|\vec{x} - \vec{x}^*\right\|_2^2$. Since $\mathbf{A}$ has full row rank, we write $\vec{x} = \mathbf{A}^T\vec{y}$ and consider the equivalent problem $\min_{\vec{y}\in\mathbb{R}^m} \frac{1}{2}\left\|\mathbf{A}^T\vec{y} - \vec{x}^*\right\|_2^2$. Expanding the objective function and using $\mathbf{A}\vec{x}^* = \vec{b}$, we get

$$\left\|\mathbf{A}^T\vec{y} - \vec{x}^*\right\|_2^2 = \left\|\mathbf{A}^T\vec{y}\right\|_2^2 - 2\langle\vec{b},\vec{y}\rangle + \left\|\vec{x}^*\right\|_2^2 \;\; .$$

Therefore, we attempt solve the following equivalent problem using accelerated coordinate descent.

$$\min_{\vec{y}\in\mathbb{R}^m} f(\vec{y}) \quad \text{where} \quad f(\vec{y}) \stackrel{\text{def}}{=} \frac{1}{2}\left\|\mathbf{A}^T\vec{y}\right\|_2^2 - \langle\vec{b},\vec{y}\rangle \;\; .$$

From Example 2, we know that the i-th direction component-wise Lipschitz constant is $L_i = \|a_i\|^2$ where a_i is the i-th row of $\mathbf{A}$ and we know that $\nabla f(\vec{y}) = \mathbf{A}\mathbf{A}^T\vec{y} - b$. Therefore, each step of ACDM consists of the following [3]

$$\vec{y}_{k+1} = \vec{y}_k - \frac{1}{L_{i_k}}\vec{f}_{i_k}(\vec{y}_k) = \vec{y}_k - \frac{1}{\|\vec{a}_{i_k}\|^2}\left(\mathbf{A}\mathbf{A}^T\vec{y}_k - \vec{b}\right)_{i_k} \;\; .$$

Recalling that we had performed the transformation $\vec{x} = \mathbf{A}^T\vec{y}$ we see that the corresponding step in $\vec{x}$ is

$$\vec{x}_{k+1} = \vec{x}_k - \frac{1}{\|a_i\|^2}\left(\mathbf{A}\vec{x}_k - \vec{b}\right)_{i_k} = \vec{x}_k + \frac{\vec{b}_{i_k} - \langle\vec{a}_{i_k},\vec{x}_k\rangle}{\|a_{i_k}\|_2^2}\vec{a}_{i_k}.$$

Therefore, ACDM applied this way yields precisely the randomized Kaczmarz method of Strohmer and Vershynin.

However, to apply the ACDM method and provide complete theoretical guarantees we need to address the problem that f as we have constructed it is not strongly convex. This is clear by the fact that the null space of $\mathbf{A}^T$ may be non trivial.

To remedy this problem we let $Z \subseteq \mathbb{R}^m$ denote the null space of $\mathbf{A}^T$, i.e. $Z \stackrel{\text{def}}{=} \{\vec{x} \in \mathbb{R}^m \mid \mathbf{A}^T\vec{x} = 0\}$, and we define the semi-norm $\|\cdot\|_{Z^\perp}$ on $\mathbb{R}^m$ by $\|\vec{y}\|_{Z^\perp} \stackrel{\text{def}}{=} \inf_{\vec{z}\in Z}\|y + z\|$. Now it is not hard to see that f is strongly convex under this seminorm with convexity parameter $\sigma_{Z^\perp} = \left\|\mathbf{A}^{-1}\right\|_2^{-2}$. Furthermore, one can prove similarly to the proof of Lemma 3 and Theorem 4 that the algorithm in this theorem achieves the desired convergence rate. ∎

[3]We ignore the thresholding here for illustration purpose.

C. Faster SDD Solvers in the Unit-cost RAM Model

A matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is called *Symmetric Diagonally Dominant (SDD)* if $\mathbf{A}^T = \mathbf{A}$ and $\forall i \neq j \in [n]$ we have $\mathbf{A}_{ii} \geq \sum_{i\neq j}|A_{ij}|$. Solving such systems has had numerous applications in both theoretical and applied computer science. For an overview of such systems of equations, their applications, and their solvers we refer the reader to [11].

The fastest running times in general for solving such systems are $\tilde{O}(m\log n\log\frac{1}{\varepsilon})$[4] due to Koutis, Miller, and Peng (cite KMP). However, the numerical stability of this algorithm is difficult to bound and when analyzed in the standard unit-cost RAM model, the best known running time is $\tilde{O}(m\log^2 n\log\frac{1}{\varepsilon})$ due to Kelner *et al.* [8].

Here we show how to cast the simplest solver presented [8] as an instance of coordinate descent, and by applying ACDM, we obtain a faster running time of $\tilde{O}(m\log^{3/2} n\log\frac{1}{\varepsilon})$ in the unit-cost RAM model. Our presentation will make heavy use of several insights from [8] and we refer the reader to that paper for further background.

Following the reasoning in [8], solving SDD systems can be reduced to solving the Laplacian system $\mathcal{L}\vec{x} = \vec{\chi}$ corresponding to a weighted connected graph $G = (V, E, \omega)$ where we call ω_e the weight of edge $e \in E$ and $r_e \stackrel{\text{def}}{=} \frac{1}{\omega_e}$ the resistance. For notational convenience, we arbitrarily orient each edge in E and using this convention define a graph's incidence matrix $\mathbf{B} \in \mathbb{R}^{E\times V}$, resistance matrix $\mathbf{R} \in \mathbb{R}^{E\times E}$, and Laplacian matrix, $\mathcal{L} \in \mathbb{R}^{V\times V}$ as follows:

$$\mathbf{B}_{(a,b),c} \stackrel{\text{def}}{=} \mathbb{1}_{a=c} - \mathbb{1}_{b=c},\; \mathbf{R}_{e_1,e_2} \stackrel{\text{def}}{=} r_{e_1}\mathbb{1}_{e_1=e_2},\; \mathcal{L} \stackrel{\text{def}}{=} \mathbf{B}^T\mathbf{R}^{-1}\mathbf{B}.$$

Letting $\vec{x}^*$ be the solution of $\mathcal{L}\vec{x} = \vec{\chi}$, we will prove the following:

Theorem 10. *By applying ACDM to a simple Laplacian system solver in [8], we can produce an $\vec{x} \in \mathbb{R}^n$ such that $\|\vec{x} - \vec{x}^*\|_{\mathcal{L}} \leq \epsilon\|\vec{x}^*\|_{\mathcal{L}}$ in time $O(m\log^{3/2} n\sqrt{\log\log n}\log(\frac{\log n}{\epsilon}))$.*

Proof: By the method of Lagrange multipliers, the minimizer $\vec{z}^*$ of the problem

$$\min_{\mathbf{B}^T\vec{z}=\vec{\chi}} \frac{1}{2}\|\vec{z}\|_{\mathbf{R}}^2 \quad \text{where} \quad \|\vec{z}\|_{\mathbf{R}} \stackrel{\text{def}}{=} \sqrt{\sum_{e\in E} r_e\vec{z}_e^2}$$

is $\mathbf{R}^{-1}\mathbf{B}\vec{x}^*$. Therefore, we can find $\vec{x}^*$ by minimizing $f(\vec{z}) \stackrel{\text{def}}{=} \frac{1}{2}\|\vec{z}\|_{\mathbf{R}}^2$ and use a BFS to get $\vec{x}$ from $\vec{z}$.

Now, let $\vec{z}_0$ be any vector such that $\mathbf{B}^T\vec{z}_0 = \vec{\chi}$. With this, the problem can be simplified to $\min_{\mathbf{B}^T\vec{y}=0}\frac{1}{2}\|\vec{z}_0 + \vec{y}\|_{\mathbf{R}}^2$. Now, the $\{\vec{c} \in \mathbb{R}^E | \mathbf{B}^T\vec{c} = 0\}$ is simply the set of circulations of the graph, called *cycle space*, and therefore to turn this constrained problem into an unconstrained problem we simply require a good basis for cycle space.

Such a basis can easily be found by a spanning tree. Given a spanning tree T of G, for any $(a,b) \in E\backslash T$ let $\vec{c}_{(a,b)} \in \mathbb{R}^E$ be the circulation that corresponds to sending 1 unit on (a,b) and 1 unit on the unique path from b to a in T. Now, the

[4]Where we use $\tilde{O}$ to hide $O(\log\log n)$ terms.

set $\{\vec{c}_e \mid e \in E \setminus T\}$ forms a basis for cycle space and with this insight we can simplify the problem further to

$$\min_{\vec{y}\in\mathbb{R}^{E\setminus T}} \frac{1}{2}\left\|\vec{z}_0 + \mathbf{C}\vec{y}\right\|_{\mathbf{R}}^2 \text{ where } \mathbf{C} \stackrel{\text{def}}{=} [c_{e_1} c_{e_2} \cdots] \in \mathbb{R}^{E\times E\setminus T} .$$

Now, for every *off-tree edge*, i.e. $e \in E \setminus T$, there is only one cycle c_e that passes through it. So, if we let $\mathbf{R}_{E\setminus T}$ be the diagonal matrix for the resistances of the off tree edges we have that for any $\vec{y} \in E \setminus T$

$$\vec{y}^T\mathbf{C}^T\mathbf{R}\mathbf{C}\vec{y} \geq \sum_{e\in E\setminus T} r_e\vec{y}(e)^2 \geq \left(\min_{e\in E\setminus T} r_e\right)\|\vec{y}\|_2^2 .$$

Therefore, the convexity parameter of $\|\vec{z}_0 + \mathbf{C}\vec{y}\|_{\mathbf{R}}^2$ is at least $\min_{e\in E\setminus T} r_e$. This could be wasteful if the resistances vary, so we compensate by rescaling the space, $\tilde{y} = \mathbf{R}^{1/2}\vec{y}$, to get the following problem:

$$\min_{\tilde{y}\in\mathbb{R}^{E\setminus T}} f(\tilde{y}) \quad \text{where} \quad f(\tilde{y}) = \frac{1}{2}\left\|\vec{z}_0 + \mathbf{C}\mathbf{R}^{-1/2}\tilde{y}\right\|_{\mathbf{R}}^2 .$$

By the same reasoning as above, the convexity parameter of f with respect to the Euclidian norm is 1 and we bound the e-th direction component-wise Lipschitz constant L_e as follows

$$L_e = \mathbb{1}_e^T\mathbf{R}^{-1/2}\mathbf{C}^T\mathbf{R}\mathbf{C}\mathbf{R}^{-1/2}\mathbb{1}_e = \frac{\vec{c}_e^T\mathbf{R}\vec{c}_e}{r_e} = \text{st}\,(e) + 1.$$

where $\text{st}\,(e)$ denotes the *stretch of e by the tree T*, i.e. the sum of the resistances of the edges on the path in T connecting the endpoints of e divided by the resistance of the edge e. Therefore, the total component-wise Lipschitz constant is given by $S_1 = \sum_{e\in E\setminus T}(\text{st}\,(e) + 1) \leq m + \sum_{e\in E}\text{st}\,(e)$. Now, as in [8], we can using the following result of Abraham and Neiman to ensure $S_1 = O(m \log n \log\log n)$

Theorem 11. *[28] In $O(m \log n \log\log n)$ time, we can compute a spanning tree with $\sum \text{st}(e) = O(m \log n \log\log n)$.*

Now, applying ACDM to f and letting e_k denote the off-tree edge i.e., coordinate, picked in iteration k, we get [5]

$$\begin{aligned}\tilde{y}_{k+1} &:= \tilde{y}_k - \frac{1}{L_{e_k}}\left(\mathbf{R}^{-1/2}\mathbf{C}^T\mathbf{R}(\vec{z}_0 + \mathbf{C}\mathbf{R}^{-1/2}\tilde{y})\right)_{e_k}\cdot\mathbb{1}_{e_k}\\ &= \tilde{y}_k - \frac{1}{L_{e_k}r_{e_k}^{1/2}}\sum_e \vec{c}_{e_k}(e)r_{e_k}(\vec{z}_0 + \mathbf{C}\mathbf{R}^{-1/2}\tilde{y})(e)\cdot\mathbb{1}_{e_k}\\ &= \tilde{y}_k - \frac{1}{L_{e_k}r_{e_k}^{1/2}}\sum_{e\in\vec{c}_{e_k}} r_e(\vec{z}_0 + \mathbf{C}\mathbf{R}^{-1/2}\tilde{y})(e)\cdot\mathbb{1}_{e_k}\end{aligned}$$

Recalling $\vec{y} = \mathbf{R}^{-1/2}\tilde{y}$ and the derivation of L_e, we have

$$\vec{y}_{k+1} := \vec{y}_k - \left[\frac{1}{r_e(\text{st}(e_k)+1)}\sum_{e\in\vec{c}_{e_k}} r_e(\vec{z}_0 + \mathbf{C}\vec{y})(e)\right]\cdot\mathbb{1}_{e_k}.$$

[5]We ignore the thresholding here again for illustration purpose.

Noting that the $\vec{z}$ with $\mathbf{B}^T\vec{z} = \vec{\chi}$ corresponding to $\vec{y}$ is $\vec{z} = \vec{z}_0 + \mathbf{C}\vec{y}$, we write the update equivalently as

$$\vec{z}_{k+1} := \vec{z}_k - \left[\frac{1}{r_e(\text{st}(e_k)+1)}\sum_{e\in\vec{c}_{e_k}} r_e\cdot\vec{z}_k(e)\right]\cdot\vec{c}_{e_k}$$

which is precisely the algorithm of the simple solver in [8]. In [8], they also prove that calls to $\vec{f}_{e_k}$ and updates to $\vec{y}_{e_k}$ can be implemented in $O(\log n)$. Therefore, by applying ACDM we can obtain a faster algorithm. Note that apply ACDM efficiently computations of $\vec{f}_{e_k}$ need to be performed on the sum of two vector without explicit summing them. However, since in this case ∇f is linear, we can just call the oracle on the two vectors separately and also use the data structure for updating coordinates in each vector separately.

In order to actually solve the Laplacian system, we need to compute $\vec{x}$ such that $\left\|\vec{x} - \mathcal{L}^\dagger\vec{\chi}\right\|_{\mathcal{L}}^2 \leq \varepsilon\left\|\mathcal{L}^\dagger\vec{\chi}\right\|_{\mathcal{L}}^2$. However, by Lemma 6.2 of [8], it suffices to show that $\|\nabla f(\tilde{y})\|_2^2 \leq \epsilon f^*$. Note that

$$\|\nabla f(\tilde{y})\|_2^2 = \sum_{e\in E\setminus T}\frac{1}{r_e}\left(\sum_{e'\in\vec{c}_e}\vec{z}(e')r_{e'}\right)^2$$

Now, suppose we choose $\tilde{y}_0 = \vec{0}$. Then, we see that $\|\tilde{y}_0 - \tilde{y}^*\|_2^2 = \|\tilde{y}^*\|_2^2 = \left\|\mathbf{R}_{E\setminus T}^{1/2}\vec{y}^*\right\|_2^2 \leq 2f^*$, and using Lemma 6.1 of [8], we have that $\frac{1}{S_\alpha^2}(f(\tilde{y}_0) - f^*) \leq \frac{f^*}{S_\alpha} \leq f^*$. Furthermore, since by our choice of spanning tree $\frac{S_1}{|E\setminus T|} = O(\log n \log\log n)$, after $k = O(m\sqrt{\log n \log\log n}\log\frac{\log n}{\epsilon})$ iterations of ACDM, by Theorem 5, we have that $\frac{1}{k}\sum_{j=k}^{2k-1}\|\nabla f(\vec{y}_k)\|_2^2 \leq \epsilon f^*$. Therefore, if we stop ACDM at random iteration between k and $2k-1$, we have that $\mathbb{E}\|\nabla f(\vec{y}_k)\|_2^2 \leq \epsilon f^*$ as desired. ■

Beyond obtaining a faster running time, we remark that this algorithm did not require recursive techniques that were necessary for Kelner *et al.* to achieve there fastest running time. The simple solver that was accelerated actually had a running time of $O(m\log^2 n \log\log n \log(n/\varepsilon))$ time due to the potentially large initial error of $f(\tilde{y}_0)$. To achieve their fastest running time, they ran their solver several times to ultimately remove the n term in $\log(n/\varepsilon)$. In our case, since ACDM measures initial error both in terms of $f(\tilde{y}_0) - f^*$ and $\|\tilde{y} - \tilde{y}^*\|^2$ and since $\|\tilde{y} - \tilde{y}^*\|^2$ is relatively small, ACDM avoids this issue entirely.

VI. Towards the Optimality of Accelerated Coordinate Descent

In this section, we prove that the accelerated coordinate descent is optimal under the assumption that the iterative method generates a sequence of vectors $\{x_k\}$ such that

$$\vec{x}_{k+1} \in \vec{x}_0 + \text{span}\left\{\vec{f}_{i_0}(\vec{x}_0), \vec{f}_{i_1}(\vec{x}_1), \cdots, \vec{f}_{i_k}(\vec{x}_k)\right\}. \quad (14)$$

This assumption forbids the iterative method from starting at any other point other than the initial point and forbids the algorithm from randomly jumping to completely new points.

Theorem 12 (ACDM Lower Bound)**.** *Assume the iterative method satisfies the assumption (14) and further assume the method randomly picks each i_k uniformly at random. Then, for any $S_1 > 4\sigma n > 0$, $x_0 \in \mathbb{R}^n$ and $k \leq \frac{n^2}{2}$, there exists a convex function f with strong convex parameter σ and total component-wise Lipschitz constant S_1 such that*

$$\mathbb{E}\left[f(\vec{x}_k)\right] - f^* \geq \frac{\sigma}{2}\left(1 - 2\sqrt{\frac{2\sigma}{nS_1}}\right)^k \|\vec{x}^* - \vec{x}_0\|_2^2 - \sqrt{\frac{\sigma S_1}{n}}\left(1 - \frac{1}{2}\sqrt{\frac{n\sigma}{S_1}}\right)^{2n}.$$

Proof: See full version. ■

ACKNOWLEDGMENT

We thank Yan Kit Chim, Andreea Gane, Jonathan Kelner, Lap Chi Lau, and Lorenzo Orecchia for helpful discussions. This work was partially supported by Hong Kong RGC grant 2150701., NSF awards 0843915 and 1111109, and a NSF Graduate Research Fellowship (grant no. 1122374).

.

REFERENCES

[1] S. Kaczmarz, "Angenäherte auflösung von systemen linearer gleichungen," *Bull. Internat. Acad. Polon.Sci. Lettres A*, p. 335?57, 1937.

[2] W. Zangwill, *Nonlinear Programming: A Unified Approach*. Prentice-Hall, 1969.

[3] H. H. Bauschke and J. M. Borwein, "On projection algorithms for solving convex feasibility problems," *SIAM Rev.*, vol. 38, no. 3, pp. 367–426, Sep. 1996.

[4] F. Natterer, *The mathematics of computerized tomography*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001.

[5] G. T. Herman, *Fundamentals of computerized tomography: image reconstruction from projections*. Springer, 2009.

[6] T. Strohmer and R. Vershynin, "A Randomized Kaczmarz Algorithm with Exponential Convergence," *Journal of Fourier Analysis and Applications*, vol. 15, no. 2, pp. 262–278, Apr. 2008.

[7] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[8] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu, "A Simple, Combinatorial Algorithm for Solving SDD Systems in Nearly-Linear Time," Jan. 2013.

[9] J. A. Kelner and A. Madry, "Faster generation of random spanning trees," in *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 13–21.

[10] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng, "Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs," in *Proceedings of the 43rd annual ACM symposium on Theory of computing*. ACM, 2011, pp. 273–282.

[11] D. A. Spielman, "Algorithms, graph theory, and the solution of laplacian linear equations," in *Proceedings of the 39th international colloquium conference on Automata, Languages, and Programming - Volume Part II*, ser. ICALP'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 24–26.

[12] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, 2003, vol. I.

[13] C. T. Kelley, *Iterative methods for optimization*. Society for Industrial and Applied Mathematics, 1987, vol. 18.

[14] Y. Nesterov, "A method for solving a convex programming problem with convergence rate 1/k^2," *Doklady AN SSSR*, vol. 269, pp. 543–547, 1983.

[15] A. Beck and M. Teboulle, "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, Jan. 2009.

[16] S. Becker, J. Bobin, and E. Candès, "NESTA: a fast and accurate first-order method for sparse recovery," *SIAM Journal on Imaging Sciences*, vol. 4, no. 1, pp. 1–39, 2011.

[17] Y. T. Lee, S. Rao, and N. Srivastava, "A New Approach to Computing Maximum Flows using Electrical Flows," *Proceedings of the 45th symposium on Theory of Computing - STOC '13*, 2013.

[18] Y. Nesterov, "Rounding of convex sets and efficient gradient methods for linear programming problems," *Optimization Methods Software*, vol. 23, no. 1, pp. 109–128, Feb. 2008.

[19] D. Bienstock and G. Iyengar, "Solving fractional packing problems in oast(1/esp) iterations," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, ser. STOC '04. New York, NY, USA: ACM, 2004, pp. 146–155.

[20] Z. Q. Luo and P. Tseng, "On the convergence of the coordinate descent method for convex differentiable minimization," *Journal of Optimization Theory and Applications*, vol. 72, no. 1, pp. 7–35, Jan. 1992.

[21] P. Richtárik and M. Takáč, "Iteration Complexity of Randomized Block-Coordinate Descent Methods for Minimizing a Composite Function," p. 33, Jul. 2011.

[22] D. A. Spielman and S.-H. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. ACM, 2004, pp. 81–90.

[23] J. A. Kelner, L. Orecchia, Y. T. Lee, and A. Sidford, "An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations," *arXiv preprint arXiv:1304.2338*, 2013.

[24] I. Koutis, G. L. Miller, and R. Peng, "A nearly-m log n time solver for sdd linear systems," in *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, oct. 2011, pp. 590 –598.

[25] T. Goldstein, B. ODonoghue, and S. Setzer, "Fast alternating direction optimization methods," *CAM report*, pp. 12–35, 2012.

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

[27] D. A. Spielman and J. Woo, "A Note on Preconditioning by Low-Stretch Spanning Trees," Mar. 2009.

[28] I. Abraham and O. Neiman, "Using petal-decompositions to build a low stretch spanning tree," in *Proceedings of the 44th symposium on Theory of Computing - STOC '12*. New York, New York, USA: ACM Press, May 2012, p. 395.