Parallel Coordinate Descent Algorithms for Sparse Phase Retrieval

Yang Yang[1], Marius Pesavento[2], Yonina C. Eldar[3], and Björn Ottersten[1]
1. Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, L-1855 Luxembourg.
2. Communication Systems Group, Technische Universität Darmstadt, Darmstadt 64283, Germany.
3. Department of Electrical Engineering, Technion-Israel Institute of Technology, Haifa 32000, Israel.
Email: yang.yang@uni.lu, pesavento@nt.tu-darmstadt.de, yonina@ee.technion.ac.il, bjorn.ottersten@uni.lu

***Abstract*—In this paper, we study the sparse phase retrieval problem, that is, to estimate a sparse signal from a small number of noisy magnitude-only measurements. We propose an iterative soft-thresholding with exact line search algorithm (STELA). It is a parallel coordinate descent algorithm, which has several attractive features: i) fast convergence, as the approximate problem solved at each iteration exploits the original problem structure, ii) low complexity, as all variable updates have a closed-form expression, iii) easy implementation, as no hyperparameters are involved, and iv) guaranteed convergence to a stationary point for general measurements. These advantages are also demonstrated by numerical tests.**

***Index Terms*—DC Programming, Majorization Minimization, Phase Retrieval, Successive Convex Approximation**

I. Introduction

Phase retrieval refers to the problem of estimating a complex-valued signal $\mathbf{x}_0 \in \mathbb{C}^K$ from the magnitude (or squared magnitude) of its noisy measurements. It has received considerable attention due to its importance in various applications, for example, x-ray crystallography, diffraction imaging and astronomy (see [1, 2] or the magazine review [3, 4, 5] and the references therein).

Suppose the n-th measurement $y_n \in \mathbb{R}^+$ is the magnitude of the output of a linear system, where

$$y_n = \left|\mathbf{a}_n^H \mathbf{x}_0 + v_n\right|,\ n = 1, \ldots, N, \tag{1a}$$

or

$$y_n = \left|\mathbf{a}_n^H \mathbf{x}_0\right| + v_n,\ n = 1, \ldots, N, \tag{1b}$$

where $\mathbf{a}_n$ is a known linear operator, v_n is additive noise, and N is the number of measurements. Here we assume $\mathbf{x}_0$ is sparse to aid in recovery. A possible approach to recover the sparse signal $\mathbf{x}_0$ is to minimize the least-square criterion $f(\mathbf{x}) \triangleq \frac{1}{2}\sum_{n=1}^N \left(y_n - \left|\mathbf{a}_n^H \mathbf{x}\right|\right)^2$ regularized by a sparsity-promoting prior function $g(\mathbf{x}) \triangleq \mu \left\|\mathbf{x}\right\|_1$:

$$\underset{\mathbf{x}\in\mathbb{C}^K}{\text{minimize}} \quad h(\mathbf{x}) \triangleq \underbrace{\frac{1}{2}\sum_{n=1}^N \left(y_n - \left|\mathbf{a}_n^H \mathbf{x}\right|\right)^2}_{f(\mathbf{x})} + \underbrace{\mu \left\|\mathbf{x}\right\|_1}_{g(\mathbf{x})}, \tag{2}$$

and $\mu > 0$ is a given and fixed regularization gain. It is a very challenging optimization problem due to the fact that g is nonsmooth and, more notably, f is nonsmooth and nonconvex.

Several algorithms have been proposed to tackle different variants of the (sparse) phase retrieval problem. One prominent example is the gradient algorithm, which is shown in [1] to converge to $\mathbf{x}_0$ when N, the number of intensity (squared magnitude) measurements $(y_n)_{n=1}^N$, is sufficiently large. When $\mathbf{x}_0$ is sparse, it can be estimated by the GESPAR algorithm [6] based on the damped Gauss-Newton Method. Nevertheless, they cannot be directly applied for problem (2) because f is a function of the magnitude measurements and is not differentiable. A smoothing technique is proposed in [7] to approximate the nonsmooth function f by a smooth function, and this smooth function is then minimized by a standard gradient-based method, but the sparse prior is not considered in [7].

An alternative to tackle the nondifferentiability of f is a truncated amplitude flow algorithm based on the notion of generalized gradient proposed in [8]. It is further generalized in [9] to estimate a sparse signal from magnitude measurements as in the sparse phase retrieval problem (2). It consists of two steps: firstly, the support of the unknown sparse signal is estimated, and secondly, the coefficients of the nonzero elements in the support are estimated iteratively. It enjoys low complexity, but the support recovery is only accurate when the number of measurements is sufficiently large. In the case of an inaccurate support recovery in the first step, the error will be further propagated to the components estimate in the second step.

Alternating minimization schemes, also known as the Fienup methods, have a long history which form another popular class of algorithms for phase retrieval problems. A recent paper [10] establishes the convergence of the Fienup methods for Fourier measurements, but the convergence is still left open for general measurements.

In [11], the majorization minimization (MM) algorithm is proposed to find a stationary point of problem (2). It iteratively minimizes an upper bound on f, which is smooth and convex. It has a guaranteed convergence to a stationary point, but it suffers from high complexity and slow convergence. This is because constructing the upper bound function consists of calculating the maximum eigenvalue of the matrix $[\mathbf{a}_1\, \mathbf{a}_2\, \ldots\, \mathbf{a}_N]$, which is computationally expensive when the number of measurements N and/or the signal dimension K is large. Furthermore, due to the restrictive upper bound constraint, the MM algorithm generally suffers from slow convergence [12].

In this paper, we propose an iterative soft-thresholding with exact line search algorithm (STELA) for the sparse phase retrieval problem (2). STELA is based on the recently developed successive convex approximation (SCA) framework for sparse signal estimation with difference-of-convex (DC) priors [13], which is a nontrivial combination of the standard MM algorithm [11] and standard SCA algorithms for smooth f and convex g [14]. In particular, at each iteration of STELA, we minimize a *convex approximation* of the global upper bound function, which is equivalent to minimizing this bound *approximately* by performing only one iteration of the parallel coordinate descent algorithm. The proposed algorithm STELA has several advantages, namely, i) fast convergence, as the approximate problem is the best-response type approximation and reserves the problem structure; ii) low complexity, as all variable updates have

The work of Y. Yang and B. Ottersten is supported by H2020-ERC AdG-AGNOSTIC (742648). The work of M. Pesavento is supported by the EXPRESS Project within the DFG Priority Program CoSIP (DFG-SPP 1798). The work of Y. C. Eldar is supported by the European Unions Horizon 2020 research and innovation program under grant No. 646804-ERC-COG-BNYQ and the Israel Science Foundation under grant No. 0100101

a closed-form expression; iii) easy implementation, as no hyperparameter tuning is required (assuming the regularization gain μ in (2) is given and fixed); and iv) guaranteed convergence to a stationary point for general measurements. Finally, we illustrate these attractive features and compare STELA with state-of-the-art algorithms by numerical tests.

II. The Proposed Parallel Coordinate Descent Algorithm

In this section, building on the SCA framework introduced in [13], we propose a parallel coordinate descent algorithm for (2). As f in the objective function is nonconvex and nonsmooth, the central idea in the proposed algorithm STELA is to first find a descent direction by constructing a smooth upper bound function and then minimizing a *convex approximation* of this bound augmented by the convex prior g. With a proper stepsize, the function value h could be decreased by updating the variable along the descent direction. This iterative process is repeated until a stationary point is found.

Finding the descent direction. Expanding the quadratic function in (2), we can rewrite the function f in (2) as

$$f(\mathbf{x}) = f^+(\mathbf{x}) - f^-(\mathbf{x}), \tag{3}$$

where

$$f^+(\mathbf{x}) \triangleq \frac{1}{2}\mathbf{x}^H\left(\sum_{n=1}^N \mathbf{a}_n\mathbf{a}_n^H\right)\mathbf{x} + \frac{1}{2}\sum_{n=1}^N |y_n|^2,$$
$$f^-(\mathbf{x}) \triangleq \sum_{n=1}^N y_n|\mathbf{a}_n^H\mathbf{x}|.$$

Although f is nonsmooth because f^- is nonsmooth and convex, it is the difference of two convex functions. This structure will be exploited when designing our iterative algorithm.

To start with, we note that $f^-(\mathbf{x})$ is convex in $\mathbf{x}$ and there always exists a subgradient $\boldsymbol{\xi}(\mathbf{x})$ such that for any $\mathbf{x}^1$ and $\mathbf{x}^2 \in \mathbb{C}^K$:

$$f^-(\mathbf{x}^1) - f^-(\mathbf{x}^2) \geq \Re(\boldsymbol{\xi}(\mathbf{x}^2)^H(\mathbf{x}^1 - \mathbf{x}^2)), \tag{4a}$$

where $\Re(\cdot)$ is the real operator. The expression of $\boldsymbol{\xi}(\mathbf{x})$ can be derived by following a similar line of analysis to [11]: for any $\mathbf{x}^1$ and $\mathbf{x}^2$,

$$\begin{aligned}\left|\mathbf{a}_n^H\mathbf{x}^1\right| - \left|\mathbf{a}_n^H\mathbf{x}^2\right| &\overset{(a)}{=} \left|\mathbf{a}_n^H\mathbf{x}^1 e^{-j\arg(\mathbf{a}_n^H\mathbf{x}^2)}\right| - \left|\mathbf{a}_n^H\mathbf{x}^2 e^{-j\arg(\mathbf{a}_n^H\mathbf{x}^2)}\right| \\ &\overset{(b)}{\geq} \Re\big(\mathbf{a}_n^H\mathbf{x}^1 e^{-j\arg(\mathbf{a}_n^H\mathbf{x}^2)}\big) - \\ &\quad \Re\big(\mathbf{a}_n^H\mathbf{x}^2 e^{-j\arg(\mathbf{a}_n^H\mathbf{x}^2)}\big) \\ &= \Re\big((e^{j\arg(\mathbf{a}_n^H\mathbf{x}^2)}\mathbf{a}_n)^H(\mathbf{x}^1 - \mathbf{x}^2)\big),\end{aligned} \tag{4b}$$

where (a) and (b) follow from the fact that, for any complex number $z \in \mathbb{C}$, $|z|$ is invariant with respect to phase and $|z| \geq \Re(z)$ (equality is achieved when z is real and nonnegative), respectively. Therefore, a subgradient of $f^-(\mathbf{x})$ is (recall that the measurements $(y_n)_{n=1}^N$ are real and nonnegative)

$$\boldsymbol{\xi}(\mathbf{x}) = \sum_{n=1}^N y_n e^{j\arg(\mathbf{a}_n^H\mathbf{x})}\mathbf{a}_n = \mathbf{A}^H(\mathbf{y} \odot e^{j\arg(\mathbf{A}\mathbf{x})}). \tag{5}$$

where $\mathbf{A} \triangleq [\mathbf{a}_1 \dots \mathbf{a}_N]^H \in \mathbb{C}^{N\times K}$, $\odot$ stands for the element-wise product, and $e^{(\cdot)}$ and $\arg(\cdot)$ are meant to be applied element-wise when the argument is a vector.

Given a point $\mathbf{x}^t$ at iteration t, we readily infer from (4) that f^- is lower bounded by a linear function:

$$f^-(\mathbf{x}) \geq f^-(\mathbf{x}^t) + \Re\big(\boldsymbol{\xi}(\mathbf{x}^t)^H(\mathbf{x} - \mathbf{x}^t)\big).$$

As a result, we design a smooth upper bound of $f(\mathbf{x}) = f^+(\mathbf{x}) - f^-(\mathbf{x})$, which is tight at $\mathbf{x} = \mathbf{x}^t$:

$$f(\mathbf{x}) \leq \underbrace{f^+(\mathbf{x}) - f^-(\mathbf{x}^t) - \Re\big(\boldsymbol{\xi}(\mathbf{x}^t)^H(\mathbf{x} - \mathbf{x}^t)\big)}_{\overline{f}(\mathbf{x};\mathbf{x}^t)}, \tag{6a}$$

$$= \frac{1}{2}\mathbf{x}^H\mathbf{A}^H\mathbf{A}\mathbf{x} - \Re\big(\boldsymbol{\xi}(\mathbf{x}^t)^H\mathbf{x}\big) + \text{const}, \tag{6b}$$

where const $= \sum_{n=1}^N |y_n|^2 - f^-(\mathbf{x}^t) + \Re\big(\boldsymbol{\xi}(\mathbf{x}^t)^H\mathbf{x}^t\big)$ and it is independent of the variable $\mathbf{x}$.

In the next step, we minimize the upper bound function $\overline{f}$ *approximately* in the sense that we optimize a convex approximation of $\overline{f}$. Note that if we minimize $\overline{f}$ exactly, we would obtain the standard MM algorithm [11]. We depart from the traditional MM algorithm because $\overline{f}$ is computationally expensive to minimize exactly.

At point $\mathbf{x}^t$, denote $\widetilde{f}(\mathbf{x};\mathbf{x}^t)$ as an approximation of $\overline{f}(\mathbf{x};\mathbf{x}^t)$. We adopt the *best-response type* approximation, where the approximate function $\widetilde{f}(\mathbf{x};\mathbf{x}^t)$ consists of the sum of K component functions, one for each coordinate (scalar element) x_k, and the component function for x_k is obtained by fixing other scalar elements $(x_j)_{j=1,j\neq k}^K$ in $\overline{f}(\mathbf{x};\mathbf{x}^t)$ and removing without loss of generality the constant that is independent of the variable $\mathbf{x}$:

$$\begin{aligned}\widetilde{f}(\mathbf{x};\mathbf{x}^t) &= \sum_{k=1}^K \overline{f}(x_k, (x_j^t)_{j=1,j\neq k}^K; \mathbf{x}^t) \\ &= \sum_{k=1}^K \Big(\frac{1}{2}\left\|\mathbf{a}_k(x_k - x_k^t) + \mathbf{A}\mathbf{x}^t\right\|_2^2 - \Re(x_k^H\xi_k(\mathbf{x}^t))\Big).\end{aligned} \tag{7}$$

As a matter of fact, $\widetilde{f}(\mathbf{x};\mathbf{x}^t) + g(\mathbf{x})$ serves as an approximation of the global upper bound function $\overline{f}(\mathbf{x};\mathbf{x}^t) + g(\mathbf{x})$ and thus also the original objective function $h(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x})$ in (2). We denote by $\mathbb{B}\mathbf{x}^t$ the optimal point that minimizes $\widetilde{f}(\mathbf{x};\mathbf{x}^t) + g(\mathbf{x})$:

$$\mathbb{B}\mathbf{x}^t \triangleq \arg\min_{\mathbf{x}} \widetilde{f}(\mathbf{x};\mathbf{x}^t) + g(\mathbf{x}) \tag{8a}$$

$$= \Big[\left|\mathbf{c}^t\right| - \mu\cdot\mathbf{d}(\mathbf{A}^H\mathbf{A})^{-1}\Big]^+ \odot e^{j\arg(\mathbf{c}^t)}, \tag{8b}$$

where $[\mathbf{x}]^+ \triangleq \max(\mathbf{x}, \mathbf{0})$, $\mathbf{d}(\mathbf{X})$ denotes the diagonal vector of $\mathbf{X}$, $\left|\mathbf{x}\right|$ and $\mathbf{x}^{-1}$ are applied element-wise, and

$$\mathbf{c}^t \triangleq \mathbf{x}^t - \mathbf{d}(\mathbf{A}^H\mathbf{A})^{-1} \odot (\mathbf{A}^H\mathbf{A}\mathbf{x}^t - \boldsymbol{\xi}(\mathbf{x}^t)).$$

Since $\mathbb{B}\mathbf{x}^t$ has a closed-form expression in the form of soft-thresholding in (8b), the "approximate problem" in (8a) can be more efficiently minimized than the global upper bound function $\overline{f} + g$.

It can be verified that $\widetilde{f}(\mathbf{x};\mathbf{x}^t)$ is a strongly convex function (hence the name "convex approximation") and $\mathbb{B}\mathbf{x}^t$ is unique. Furthermore, it satisfies the so-called gradient consistency condition, that is, its gradient is identical to that of $\overline{f}(\mathbf{x};\mathbf{x}^t)$ at (and only at) $\mathbf{x} = \mathbf{x}^t$:

$$\nabla\widetilde{f}(\mathbf{x}^t;\mathbf{x}^t) = \mathbf{A}^H\mathbf{A}\mathbf{x}^t - \boldsymbol{\xi}(\mathbf{x}^t) = \nabla\overline{f}(\mathbf{x}^t;\mathbf{x}^t). \tag{9}$$

It follows from [14, Prop. 1] that (i) $\mathbb{B}\mathbf{x}^t - \mathbf{x}^t$ is a *descent direction* of $\overline{f}(\mathbf{x};\mathbf{x}^t) + g(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}^t$ in the sense that

$$\Re((\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)^H\nabla\overline{f}(\mathbf{x}^t;\mathbf{x}^t)) + g(\mathbb{B}\mathbf{x}^t) - g(\mathbf{x}^t) < 0, \tag{10}$$

and (ii) there exists a stepsize $\gamma > 0$ such that

$$\overline{f}(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t);\mathbf{x}^t) + g(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) < \overline{f}(\mathbf{x}^t;\mathbf{x}^t) + g(\mathbf{x}^t).$$

Since $\overline{f}(\mathbf{x};\mathbf{x}^t)$ is a global upper bound of $f(\mathbf{x})$ which is tight at $\mathbf{x} = \mathbf{x}^t$ (cf. (6)), the above inequality implies that

$$f(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) + g(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) < f(\mathbf{x}^t) + g(\mathbf{x}^t).$$

This motivates us to update $\mathbf{x}^{t+1}$ as follows

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \gamma^t(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t). \tag{11}$$

Stepsize calculation. In practice, the stepsize γ^t could be obtained by performing exact line search, which aims at finding the stepsize that yields the largest decrease of $f(\mathbf{x}) + g(\mathbf{x})$ along the descent direction $\mathbb{B}\mathbf{x}^t - \mathbf{x}^t$:

$$\underset{\gamma}{\text{minimize}}\ f(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) + g(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)). \quad (12)$$

However, since the optimization problem in (12) is nonconvex, an alternative is to replace f in (12) by its convex upper bound $\overline{f}$:

$$\underset{\gamma}{\text{minimize}}\ \overline{f}(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t); \mathbf{x}^t) + g(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)). \quad (13)$$

Although the optimization problem in (13) is convex and presumably has a much lower complexity than (12), it is still a nonsmooth problem due to g.

To further reduce the complexity of the line search in (13), we apply the definition of convex functions to g: for any $\gamma \in [0, 1]$,

$$\begin{aligned} g(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) &\le (1-\gamma)g(\mathbf{x}^t) + \gamma g(\mathbb{B}\mathbf{x}^t) \\ &= g(\mathbf{x}^t) + \gamma(g(\mathbb{B}\mathbf{x}^t) - g(\mathbf{x}^t)). \end{aligned} \quad (14)$$

In other words, the nonsmooth function $g(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t))$ is always upper bounded by the differentiable and linear function $g(\mathbf{x}^t) + \gamma(g(\mathbb{B}\mathbf{x}^t) - g(\mathbf{x}^t))$. We thus propose to perform the line search over the following function which is obtained by replacing the nonsmooth function g in (13) by its upper bound (14):

$$\gamma^t \triangleq \underset{0\le\gamma\le 1}{\arg\min} \left\{ \begin{array}{l} \overline{f}(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t); \mathbf{x}^t) \\ +g(\mathbf{x}^t) + \gamma(g(\mathbb{B}\mathbf{x}^t) - g(\mathbf{x}^t)) \end{array} \right\} \quad (15a)$$

$$= \left[-\frac{\Re((\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)^H \nabla\overline{f}(\mathbf{x}^t; \mathbf{x}^t)) + g(\mathbb{B}\mathbf{x}^t) - g(\mathbf{x}^t)}{\|\mathbf{A}(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)\|_F^2} \right]_0^1, \quad (15b)$$

where $[x]_0^1$ denotes the projection of x onto $[0, 1]$: $[x]_0^1 = \max(0, \min(x, 1))$, and $\nabla\overline{f}(\mathbf{x}^t; \mathbf{x}^t)$ is given in (9). The proposed line search admits a simple closed-form expression in (15b). Note that $\gamma^t > 0$ as the term inside the square bracket is strictly positive in view of (10); a formal proof is given in [13, Prop. 2]. Therefore, we can show that $\{h(\mathbf{x}^t)\}$ is monotonically decreasing:

$$\begin{aligned} & h(\mathbf{x}^t) = f(\mathbf{x}^t) + g(\mathbf{x}^t) \\ &= \overline{f}(\mathbf{x}^t + \gamma^t(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t); \mathbf{x}^t) + g(\mathbf{x}^t) + \gamma^t(g(\mathbb{B}\mathbf{x}^t) - g(\mathbf{x}^t))\big|_{\gamma=0} \\ &\overset{(a)}{>} \overline{f}(\mathbf{x}^t + \gamma^t(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t); \mathbf{x}^t) + g(\mathbf{x}^t) + \gamma^t(g(\mathbb{B}\mathbf{x}^t) - g(\mathbf{x}^t)) \\ &\overset{(b)}{\ge} \overline{f}(\mathbf{x}^t + \gamma^t(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t); \mathbf{x}^t) + g(\mathbf{x}^t + \gamma^t(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) \\ &\overset{(c)}{\ge} f(\mathbf{x}^t + \gamma^t(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) + g(\mathbf{x}^t + \gamma^t(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) \\ &\overset{(d)}{=} f(\mathbf{x}^{t+1}) + g(\mathbf{x}^{t+1}) = h(\mathbf{x}^{t+1}), \end{aligned} \quad (16)$$

where (a) comes from the optimality of γ^t in (15a), (b) from the definition of convex functions in (14), (c) from the fact that $\overline{f}(\mathbf{x}; \mathbf{x}^t)$ is a global upper bound of $f(\mathbf{x})$, and (d) from the update rule (11).

We term the proposed update rules (8) and (15) as Soft-Thresholding with Exact Line search Algorithm (STELA). This iterative procedure is formally summarized in Algorithm 1. In what follows, we draw a few comments on its properties.

On the convergence speed. The best-response type approximation is commonly used in Gauss-Seidel type Block Coordinate Descent (BCD) algorithms [15, Sec. 2.7]. STELA inherits the best-response type approximation, but unlike the sequential update in BCD algorithms, all elements (x_k) are updated in parallel. Besides, as $\nabla^2 f^+(\mathbf{x}) = \mathbf{A}^H\mathbf{A}$, partial second-order information in terms of the diagonal elements of $\mathbf{A}^H\mathbf{A}$ is exploited in (8), and the convergence is thus typically faster than in first-order algorithms. In addition, the stepsize is calculated based on the line search, which guarantees notable decrease in the objective function value at each iteration.

Algorithm 1 STELA for Sparse Phase Retrieval (2)

Input: linear operator $\mathbf{A}$, measurements $\mathbf{y}$, sparsity regularization gain μ, stop criterion δ.

Initialization: $t = 0$, $\mathbf{x}^0$ (arbitrary but fixed).

S1: Find the descent direction $\mathbb{B}\mathbf{x}^t - \mathbf{x}^t$:

$$\begin{aligned} \nabla\overline{f}(\mathbf{x}^t; \mathbf{x}^t) &= \mathbf{A}^H(\mathbf{A}\mathbf{x}^t - \mathbf{y} \odot e^{j\arg(\mathbf{A}\mathbf{x}^t)}), \\ \mathbf{c}^t &= \mathbf{x}^t - \mathbf{d}(\mathbf{A}^H\mathbf{A})^{-1} \odot \nabla\overline{f}(\mathbf{x}^t; \mathbf{x}^t), \\ \mathbb{B}\mathbf{x}^t &= \left[|\mathbf{c}^t| - \mu \cdot \mathbf{d}(\mathbf{A}^H\mathbf{A})^{-1}\right]^+ \odot e^{j\arg(\mathbf{c}^t)}. \end{aligned}$$

S2: Compute the stepsize γ^t:

$$\gamma^t = \left[-\frac{\Re((\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)^H \nabla\overline{f}(\mathbf{x}^t; \mathbf{x}^t)) + \mu(\|\mathbb{B}\mathbf{x}^t\|_1 - \|\mathbf{x}^t\|_1)}{(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)^H\mathbf{A}^H\mathbf{A}(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)} \right]_0^1.$$

S3: Update $\mathbf{x}$:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \gamma^t(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t).$$

S4: Check the stop criterion: STOP if

$$\left|\Re((\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)^H \nabla\overline{f}(\mathbf{x}^t; \mathbf{x}^t)) + \mu(\|\mathbb{B}\mathbf{x}^t\|_1 - \|\mathbf{x}^t\|_1)\right| \le \delta.$$

Otherwise $t \leftarrow t + 1$ and go to **S1**.

On the complexity. STELA enjoys low complexity because both the approximate problem and the stepsize have a closed-form expression, which involves basic linear algebraic operations. The most complex operation is to obtain $\mathbf{d}(\mathbf{A}^H\mathbf{A})$, the diagonal elements of $\mathbf{A}^H\mathbf{A}$. It can be obtained by calculating the squared Frobenius norm of each row of $\mathbf{A}$ (matrix-matrix multiplication between $\mathbf{A}^H$ and $\mathbf{A}$ is not needed), which is fully parallelizable. STELA is also easy to use, because it does not involve any hyperparameters.

On the convergence. Since STELA is a descent algorithm, the sequence $\{h(\mathbf{x}^t)\}$ is monotonically decreasing, cf. (16). From the mathematical perspective, problem (2) is equivalent to minimizing a convex and smooth function $f^+(\mathbf{x})$ regularized by a DC prior $g(\mathbf{x}) - f^-(\mathbf{x})$. Therefore it readily follows from [13, Thm. 1] that every limit point of the sequence $\{\mathbf{x}^t\}$ generated by STELA in Algorithm 1 is a stationary point of (2).

On the comparison with the MM algorithm. Minimizing the upper bound function $\overline{f}(\mathbf{x}; \mathbf{x}^t) + g(\mathbf{x})$ exactly leads to the standard MM algorithm:

$$\underset{\mathbf{x}}{\text{minimize}}\ \overline{f}(\mathbf{x}; \mathbf{x}^t) + g(\mathbf{x}).$$

As this problem does not have a closed-form solution and must be solved iteratively, $\mathbf{x}^H\mathbf{A}^H\mathbf{A}\mathbf{x}$ in $\overline{f}$ (cf. (6)) is further upper bounded by $\lambda_{\max}(\mathbf{A}^H\mathbf{A})\mathbf{x}^H\mathbf{x}$ in the MM algorithm proposed in [11]:

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}} \left\{ \begin{array}{l} \frac{1}{2}\lambda_{\max}(\mathbf{A}^H\mathbf{A})\mathbf{x}^H\mathbf{x} \\ -\Re\left((\mathbf{x} - \mathbf{x}^t)^H\boldsymbol{\xi}(\mathbf{x}^t)\right) + \mu\|\mathbf{x}\|_1 \end{array} \right\}. \quad (17)$$

Although $\mathbf{x}^{t+1}$ in (17) has a closed-form expression, calculating the maximum eigenvalue of $\mathbf{A}^H\mathbf{A}$ is computationally expensive given the large dimension of $\mathbf{A}$. Furthermore, the MM algorithm (17) may converge slowly because $\mathbf{A}$ is typically ill-conditioned with many zero eigenvalues and the global upper bound function adopted in (17) tends to be conservative. In contrast, $\lambda_{\max}(\mathbf{A}^H\mathbf{A})$ is not needed in STELA, as the approximate function $\widetilde{f}(\mathbf{x}; \mathbf{x}^t)$ in (7) does not have to be a global upper bound of the original function $f(\mathbf{x})$. This renders more flexibility in designing new algorithms that converge faster and have lower complexity.

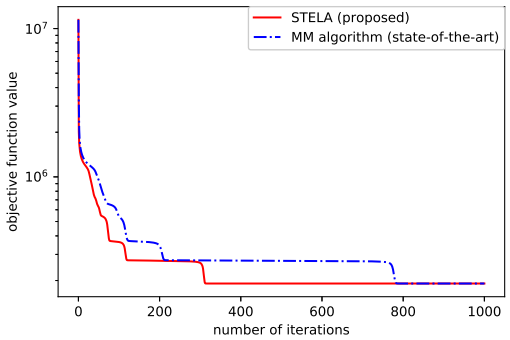

Figure 1. Objective function value $h(\mathbf{x}^t)$ versus the number of iterations t.

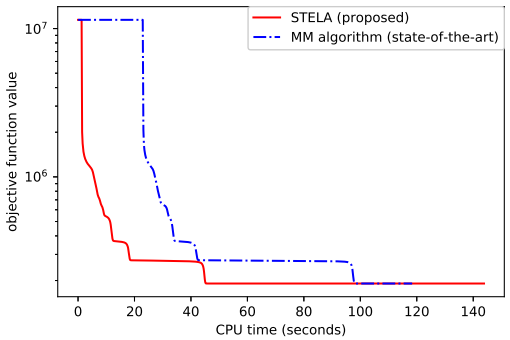

Figure 2. Objective function value $h(\mathbf{x}^t)$ versus the CPU time (in seconds).

III. Simulations

In this section, we perform numerical tests to compare the proposed algorithm STELA with the MM algorithm [11] (cf. (17)) and the sparse truncated amplitude flow algorithm (SPARTA) [9].

The simulation parameters are set as follows. All elements of $\mathbf{A}$ are randomly generated by the complex normal distribution $1/\sqrt{2}(\mathcal{N}(\mathbf{0},\mathbf{I}) + j\mathcal{N}(\mathbf{0},\mathbf{I}))$. The density of the sparse signal $\mathbf{x}_0$ is 0.1, so the number of nonzero elements is $K_0 \triangleq 0.1K$, and the nonzero elements are randomly generated by complex normal distribution $\mathcal{N}(\mathbf{0},\mathbf{I}) + j\mathcal{N}(\mathbf{0},\mathbf{I})$. The noise vector $\mathbf{v}$ is real and follows the Gaussian distribution with mean 0 and variance 10^{-4}, namely, $\mathcal{N}(\mathbf{0}, 10^{-4})$. The magnitude-only measurement vector $\mathbf{y}$ is generated according to (1a). The sparsity regularization parameter μ is set to be $\mu = 0.05\left\|\mathbf{y}^T\mathbf{A}\right\|_\infty$, which is observed empirically to have a high successful estimate rate. If not otherwise stated, the simulation results are averaged over 20 repetitions.

We set $K = 2000$ and $N = 100K_0 = 20000$, and compare STELA with the MM algorithm, both with the same random initialization. The number of iterations and the achieved objective value is plotted in Fig. 1, from which we see that STELA needs fewer iterations to converge. In particular, STELA converges after 300 iterations, while the MM algorithm converges after 800 iterations. This consolidates our argument that the restrictive global upper bound constraint in the MM algorithm tends to slow down convergence. Removing this constraint makes it possible to design new approximate functions that lead to faster convergence. We remark that an acceleration scheme is proposed in [11], but it involves a successive line search (backtracking) over the original nonsmooth nonconvex objective function and has high complexity (see the discussion in the context of exact line search in (12)).

Interestingly, it appears that both algorithms consist of two phases before convergence, namely, first a fast improvement phase and then a slow refinement phase. The slow refinement phase might be related to the issue of escaping from the saddle points. We see from Fig. 1 that STELA enters the slow refinement phase earlier and exits the slow refinement phase faster than the MM algorithm. A thorough and solid theoretical analysis is interesting and left for future work.

Fig. 2 shows the CPU time and the achieved objective value. The running time consists of both the initialization stage required for preprocessing (represented by a flat curve) and the formal stage in which the iterations are carried out. For example, in STELA, $\mathbf{d}(\mathbf{A}^T\mathbf{A})$ is computed in the initialization stage since it is required in the iterative variable update in the formal stage, cf. (8); in MM algorithm, the maximum eigenvalue of $\mathbf{A}^H\mathbf{A}$ is computed.

We readily see from Fig. 2 that computing the maximum eigenvalue of $\mathbf{A}^H\mathbf{A}$ in the initialization stage is a major overhead of the MM algorithm. By comparison, computing the diagonal elements of $\mathbf{A}^H\mathbf{A}$ is computationally much more affordable. Besides, STELA has a slightly higher computational complexity per iteration due to the line search, but the overall CPU time before convergence (50 seconds) is still much shorter than that of the MM algorithm (100 seconds).

In what follows, we compare STELA with SPARTA in terms of the estimation error defined as $\min_{\phi\in[0,2\pi]}\left\|\mathbf{x}e^{j\phi} - \mathbf{x}_0\right\|_F / \left\|\mathbf{x}_0\right\|_F$. STELA is run 10 times with different random initializations (RI), and the minimum estimation error is reported, while SPARTA starts with the spectral initialization (SI). This experiment is repeated 100 times, and the average estimation error with respect to the signal dimension K, density and number of measurements N is given in Table I.

density	K	K_0	$N(=16K_0)$	STELA (RI)	SPARTA (SI)
0.1	100	10	160	0.1184	0.5523
	200	20	320	0.4369	0.7184
	300	30	480	0.8392	0.8588
0.05	100	5	80	0.2614	0.7239
	200	10	160	0.6413	0.8048
	300	15	240	0.8975	0.9499

Table I
THE AVERAGE ESTIMATION ERROR OF STELA AND SPARTA

We see from Table I that STELA has a smaller estimation error than SPARTA in the tested scenarios. The measurement/dimension (N/K) ratio is fixed for a given density, but the performance of both algorithms deteriorate as K increases, which implies that more measurements are needed to maintain the same estimation error. Note that in SPARTA, the support is estimated first and then the elements in the support are estimated iteratively. However, when the number of noisy measurements is relatively small, the support estimate is not accurate, and the final estimate is erroneous. Furthermore, the performance guarantee of SPARTA is only available for SI, which may incur a high computational complexity. By comparison, STELA with simple RI is more robust to noise and lack of measurements.

IV. Concluding Remarks

In this paper, we studied the problem of estimating a sparse signal from a small number of magnitude-only measurements from a linear system. To solve this nonsmooth nonconvex optimization problem, we proposed a parallel coordinate descent algorithm called STELA. The proposed algorithm STELA is based on a nontrivial combination of MM algorithms and SCA algorithms: at each iteration, a global upper bound function of the original nonsmooth nonconvex objective function is minimized approximately by performing the parallel coordinate descent update once. Then the stepsize is calculated by performing line search over a properly designed differentiable function and admits a closed-form expression. STELA has several advantages, such as fast convergence, low complexity, and guaranteed convergence to a stationary point for general measurements.

REFERENCES

[1] E. J. Candes, X. Li, and M. Soltanolkotabi, "Phase Retrieval via Wirtinger Flow: Theory and Algorithms," *IEEE Transactions on Information Theory*, vol. 61, no. 4, pp. 1985–2007, Apr. 2015.

[2] J. Sun, Q. Qu, and J. Wright, "A Geometric Analysis of Phase Retrieval," *Foundations of Computational Mathematics*, vol. 18, no. 5, pp. 1131–1198, Oct. 2018.

[3] Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, and M. Segev, "Phase Retrieval with Application to Optical Imaging: A contemporary overview," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 87–109, May 2015.

[4] Y. C. Eldar, N. Hammen, and D. G. Mixon, "Recent Advances in Phase Retrieval [Lecture Notes]," *IEEE Signal Processing Magazine*, vol. 33, no. 5, pp. 158–162, Sep. 2016.

[5] D. R. Luke, "Phase retrieval, what's new?" *SIAG / OPT Views and News*, vol. 25, no. 1, pp. 1–6, 2017.

[6] Y. Shechtman, A. Beck, and Y. C. Eldar, "GESPAR: Efficient Phase Retrieval of Sparse Signals," *IEEE Transactions on Signal Processing*, vol. 62, no. 4, pp. 928–938, Feb. 2014.

[7] S. Pinilla, J. Bacca, and H. Arguello, "Phase Retrieval Algorithm via Nonconvex Minimization Using a Smoothing Function," *IEEE Transactions on Signal Processing*, vol. 66, no. 17, pp. 4574–4584, Sep. 2018.

[8] G. Wang, G. B. Giannakis, and Y. C. Eldar, "Solving Systems of Random Quadratic Equations via Truncated Amplitude Flow," *IEEE Transactions on Information Theory*, vol. 64, no. 2, pp. 773–794, Feb. 2018.

[9] G. Wang, L. Zhang, G. B. Giannakis, M. Akcakaya, and J. Chen, "Sparse Phase Retrieval via Truncated Amplitude Flow," *IEEE Transactions on Signal Processing*, vol. 66, no. 2, pp. 479–491, Jan. 2018.

[10] E. J. R. Pauwels, A. Beck, Y. C. Eldar, and S. Sabach, "On Fienup Methods for Sparse Phase Retrieval," *IEEE Transactions on Signal Processing*, vol. 66, no. 4, pp. 982–991, Feb. 2018.

[11] T. Qiu and D. P. Palomar, "Undersampled Sparse Phase Retrieval via Majorization-Minimization," *IEEE Transactions on Signal Processing*, vol. 65, no. 22, pp. 5957–5969, Nov. 2017.

[12] Y. Sun, P. Babu, and D. P. Palomar, "Majorization-Minimization Algorithms in Signal Processing, Communications, and Machine Learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 794–816, Feb. 2017.

[13] Y. Yang, M. Pesavento, S. Chatzinotas, and B. Ottersten, "Successive Convex Approximation Algorithms for Sparse Signal Estimation with Nonconvex Regularizations," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 6, pp. 1–19, Dec. 2018.

[14] Y. Yang and M. Pesavento, "A Unified Successive Pseudoconvex Approximation Framework," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3313–3328, Jul. 2017.

[15] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.