Parallel and Hybrid Soft-Thresholding Algorithms with Line Search for Sparse Nonlinear Regression

Yang Yang[1], Marius Pesavento[2], Symeon Chatzinotas[1] 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.
Email: yang.yang@uni.lu, pesavento@nt.tu-darmstadt.de, symeon.chatzinotas@uni.lu, bjorn.ottersten@uni.lu

***Abstract*—In this paper, we propose a convergent iterative algorithm for nondifferentiable nonconvex nonlinear regression problems. The proposed parallel algorithm consists in optimizing a sequence of successively refined approximate functions. Compared with the popular iterative soft-thresholding algorithm commonly known as ISTA, which is the benchmark algorithm for such problems, it has two attractive features which lead to a notable reduction in the algorithm's complexity: the proposed approximate function does not have to be a global upper bound of the original function, and the stepsize can be efficiently computed by the line search scheme which is carried out over a properly constructed differentiable function. Furthermore, when the parallel algorithm cannot be fully parallelized due to memory/processor constraints, we propose a hybrid updating scheme that divides the whole set of variables into blocks which are updated sequentially. Since the stepsize is obtained by performing the line search along the coordinate of each block variable, the proposed hybrid algorithm converges faster than state-of-the-art hybrid algorithms based on constant stepsizes and/or decreasing stepsizes. Finally, the proposed algorithms are numerically tested.**

***Index Terms*—Big Data, Block Coordinate Descent, Line Search, Linear Regression, Nonlinear Regression, Successive Convex Approximation**

I. Introduction

In this paper, we consider the problem of estimating a sparse signal $\mathbf{x} \in \mathbb{R}^{I\times 1}$ from a noisy measurement $\mathbf{y} \in \mathbb{R}^{N\times 1}$ which is the output of a nonlinear system:

$$y_n = \sigma(\mathbf{a}_n^T\mathbf{x}) + v_n,\ n = 1, \ldots, N, \tag{1}$$

where v_n is the noise, $(\mathbf{a}_n)_{n=1}^N$ is the covariate and σ specifies the nonlinear regression model. Common choices of σ are cosine and sigmoid functions. If σ is the identity function, (2) reduces to the well known linear regression problem.

A natural measure for the data mismatch is the least square error augmented by regularization functions to promote the sparsity of $\mathbf{x}$:

$$\underset{\mathbf{x}}{\text{minimize}} \quad \frac{1}{2}\sum_{n=1}^N \left(y_n - \sigma(\mathbf{a}_n^T\mathbf{x})\right)^2 + \lambda \left\|\mathbf{x}\right\|_1. \tag{2}$$

The objective function can be written as the sum of a differentiable function $f(\mathbf{x})$ and a nondifferentiable but convex function $g(\mathbf{x})$:

$$f(\mathbf{x}) \triangleq \frac{1}{2}\sum_{n=1}^N \left(y_n - \sigma(\mathbf{a}_n^T\mathbf{x})\right)^2, \text{ and } g(\mathbf{x}) \triangleq \lambda \left\|\mathbf{x}\right\|_1. \tag{3}$$

As a special case of (2) when σ is the identity function, the linear regression problem has received extensive attention because it plays a fundamental role in many applications, for example, image processing, parameter estimation and subspace clustering. Nevertheless, it is typically a large scale optimization problem and cannot be solved by traditional convex optimization algorithms such as the interior point method that do not scale well. Many new algorithms have been proposed, for example, fast iterative soft-thresholding algorithm (FISTA), (greedy) block coordinate descent (BCD) method, and alternating direction method of multiplier (ADMM).

When σ is nonlinear, $f(\mathbf{x})$ is in general a nonconvex function. It was shown in [1] that under mild conditions, every stationary point of the nonconvex optimization problem (2) enjoys an optimal statistical rate of convergence. An iterative soft-thresholding algorithm (ISTA) was then proposed in [1] to find a stationary point of (2). However, its convergence speed is typically slow. Besides, the complexity per iteration is high because a successive line search scheme is employed to estimate the Lipschitz constant of ∇f and the soft-thresholding operator must be called several times inside a single iteration.

ISTA proposed in [1] is fully parallelizable. Fully parallelizable algorithms are generally desirable because all elements can be updated simultaneously and the convergence speed is typically faster than sequential update such as BCD algorithms [2]. However, fully parallelizable algorithms pose a demanding requirement on the memory and processing units due to the formidable sheer data volume. For example, to load the matrix $\mathbf{A} \triangleq [\mathbf{a}_1\, \mathbf{a}_2\, \ldots\, \mathbf{a}_N]$ with a dimension of 0.1 Million$\times$1 Million while each element is represented by a full double-precision floating point, the memory capacity needs to be as large as 745 GB. In this case, the requirement on the hardware is really demanding and difficult to satisfy all the time. It is possible to divide the fully parallelizable algorithms into blocks which are then executed sequentially. However, this is a rather naive approach because when a particular block is executed, the most recent updates of previous blocks will not be exploited.

To address this issue, several hybrid update schemes have been proposed in literature [3, 4, 5]: the whole set of variables $\mathbf{x}$ is divided into K block variables $\mathbf{x} = (\mathbf{x}_k)_{k=1}^K$, such that the iterative algorithm can be fully parallelized when being applied to solve the optimization problem (2) with respect to (w.r.t.) the block variable $\mathbf{x}_k$ (rather than the full variable $\mathbf{x}$). In the hybrid update scheme, all elements of $\mathbf{x}_k$ are updated simultaneously, and different block variables $(\mathbf{x}_k)_{k=1}^K$ are updated sequentially. Nevertheless, such schemes also have their limitations. The hybrid algorithm proposed in [3] is not applicable when the objective function is nonsmooth. The convergence of hybrid algorithms proposed in [4, 5] is only established under decreasing stepsizes, for example, $\gamma^t = 1/t$. On the one hand, a slow decay of the stepsize is preferable to make notable progress and to achieve satisfactory convergence speed; on the other hand, theoretical convergence is guaranteed only when the stepsize decays fast enough. In practice, it is a difficult task on its own to find a decay rate for the stepsize that provides a good trade-off between convergence speed and convergence guarantee, and current practices mainly rely on heuristics [5]. Although it is shown in [4, 5] that constant stepsizes can also be used, the choice of the constant stepsizes depends on unknown parameters that are not easy to obtain/estimate.

In this paper, we first propose a parallel and then a hybrid iterative Soft-ThrEsholding with Line search Algorithm (STELA) for problem (2). The proposed parallel STELA consists in optimizing a

The work of Yang, Chatzinotas and Ottersten is supported by the ERC project AGNOSTIC, and the work of Pesavento is supported by the EXPRESS Project within the DFG Priority Program CoSIP (DFG-SPP 1798).

sequence of successively refined approximate functions and it has several attractive features: i) the approximate function is a convex approximation of the original function but it does not need to be a global upper bound of the original function; ii) the (exact or successive) line search scheme to calculate the stepsize is carried out over a properly constructed differentiable function. The parallel STELA has a faster convergence than constant/decreasing stepsizes and a lower complexity than the traditional line search which is carried out over the original nonsmooth objective function. The proposed line search scheme is then extended to the hybrid STELA: when $\mathbf{x}_k$ is being updated, the stepsize is obtained by performing the line search along the coordinate of $\mathbf{x}_k$. Both the parallel and the hybrid STELA converge to a stationary point of (2).

II. The Proposed Parallel Successive Convex Approximation Algorithms

In this section, we propose an iterative algorithm to find a stationary point of problem (2). It consists of solving a sequence of successively refined approximate problems, which are presumably much easier to solve than the original problem.

To this end, given $\mathbf{x}^t$ at iteration t, we define the approximate function of $f(\mathbf{x})$ at $\mathbf{x}^t$ as $\tilde{f}(\mathbf{x};\mathbf{x}^t)$:

$$\tilde{f}(\mathbf{x};\mathbf{x}^t) = f(\mathbf{x}^t)+(\mathbf{x}-\mathbf{x}^t)^T\nabla f(\mathbf{x}^t)+\frac{1}{2}(\mathbf{x}-\mathbf{x}^t)^T\mathbf{H}^t(\mathbf{x}-\mathbf{x}^t), \quad (4)$$

where $\mathbf{H}^t \succ \mathbf{0}$ is a diagonal matrix so that $\tilde{f}(\mathbf{x};\mathbf{x}^t)$ is separable among the different scalar elements $(x_i)_{i=1}^I$. Note that $\tilde{f}(\mathbf{x};\mathbf{x}^t)$ is strongly convex and its function value is equal to $f(\mathbf{x})$ at $\mathbf{x}=\mathbf{x}^t$, but $\tilde{f}(\mathbf{x};\mathbf{x}^t)$ is not necessarily a global upper bound of $f(\mathbf{x})$.

The approximate problem is

$$\underset{\mathbf{x}\in\mathcal{X}}{\text{minimize}} \quad \tilde{f}(\mathbf{x};\mathbf{x}^t) + g(\mathbf{x}), \quad (5)$$

and its optimal point is denoted as

$$\mathbb{B}\mathbf{x}^t \triangleq \underset{\mathbf{x}\in\mathcal{X}}{\arg\min}\ \tilde{f}(\mathbf{x};\mathbf{x}^t) + g(\mathbf{x}) \quad (6a)$$

$$= \mathcal{S}_{(\mathbf{H}^t)^{-1}\boldsymbol{\lambda}}(\mathbf{x}^t - (\mathbf{H}^t)^{-1}\nabla f(\mathbf{x}^t)), \quad (6b)$$

where $\mathcal{S}_{\mathbf{a}}(\mathbf{b})$ is the soft-thresholding operator and $\mathcal{S}_{\mathbf{a}}(\mathbf{b}) \triangleq [\mathbf{b}-\mathbf{a}]^+ - [-\mathbf{b}-\mathbf{a}]^+$.

Note that $\mathbb{B}\mathbf{x}^t$ is always unique because $\tilde{f}(\mathbf{x};\mathbf{x}^t)$ is strongly convex in $\mathbf{x}$ for any given and fixed $\mathbf{x}^t$. Furthermore, gradient of $\tilde{f}(\mathbf{x};\mathbf{x}^t)$ and $f(\mathbf{x})$ are identical at $\mathbf{x}=\mathbf{x}^t$:

$$\nabla f(\mathbf{x};\mathbf{x}^t)\big|_{\mathbf{x}=\mathbf{x}^t} = \nabla f(\mathbf{x}^t) + \mathbf{H}^t(\mathbf{x}-\mathbf{x}^t)\big|_{\mathbf{x}=\mathbf{x}^t} = \nabla f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}^t}.$$

It follows from [2, Prop. 1] that $\mathbb{B}\mathbf{x}^t - \mathbf{x}^t$ is a descent direction and if we update the variable $\mathbf{x}$ according to the following rule:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t), \quad (7)$$

then there exists a stepsize $\gamma \in [0,1]$ such that $f(\mathbf{x}^{t+1}) < f(\mathbf{x}^t)$. A natural (and traditional) choice of the stepsize γ is given by the exact line search:

$$\min_{0\le\gamma\le 1} \left\{ f(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) + g(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) \right\}, \quad (8)$$

in which the stepsize that yields the largest decrease in objective function value along the direction $\mathbb{B}\mathbf{x}^t - \mathbf{x}^t$ is selected. Nevertheless, this choice leads to high computational complexity, because $f(\mathbf{x})$ is nonconvex and $g(\mathbf{x})$ is nondifferentiable, and the exact line search involves minimizing a nonconvex nondifferentiable function. To reduce the complexity, an alternative is to employ the so-called successive line search: given predefined constants $\alpha \in (0,1)$ and $\beta \in (0,1)$, the stepsize γ^t is set to be $\gamma^t = \beta^{m_t}$ where m_t is the smallest nonnegative integer that satisfies the inequality:

$$f(\mathbf{x}^t + \beta^m(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) + g(\mathbf{x}^t + \beta^m(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) \le f(\mathbf{x}^t) + g(\mathbf{x}^t) - \alpha\beta^m \left\|\mathbb{B}\mathbf{x}^t - \mathbf{x}^t\right\|^2. \quad (9)$$

The complexity of the successive line search lies in the repeated evaluation of the nondifferentiable function $g(\mathbf{x}^t+\beta^m(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))$ for $m = 0,\ldots,m_t$. This could be further saved by using constant and decreasing stepsizes. However, both of these stepsize rules usually lead to slow convergence and suffer from parameter tuning (cf. [2]), except for the case that $\mathbf{H}^t = c\mathbf{I}$ and $c > L_{\nabla f}$ ($L_{\nabla f}$ is the Lipschitz constant of ∇f): it is shown in [2] that a constant unit stepsize $\gamma^t = 1$ always yields a larger decrease than the successive line search scheme. As a matter of fact, the meticulous choice of stepsizes have become a major bottleneck for successive convex approximation algorithm [6].

To reduce the complexity of traditional exact line search scheme (8), it is shown in [2, Sec. III-A] that it suffices to perform the exact line search over the following differentiable function:

$$\min_{0\le\gamma\le 1} \left\{ f(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) + g(\mathbf{x}^t) + \gamma(g(\mathbb{B}\mathbf{x}^t) - g(\mathbf{x}^t)) \right\}, \quad (10)$$

which is an upper bound of the objective function in (8) after applying Jensen's inequality to the convex nondifferentiable function $g(\mathbf{S})$:

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

This exact line search scheme (10) has a much lower complexity than the traditional scheme (8) because the objective function in (10) is differentiable. Sometimes the optimal point of (10) may even have a closed-form solution, even though $f(\mathbf{x})$ is nonconvex; see [7] for such an example.

If the nonconvex differentiable function in (10) is still difficult to optimize, we could instead perform the successive version of the exact line search (10): given predefined constants $\alpha \in (0,1)$ and $\beta \in (0,1)$, the stepsize is set to $\gamma^t = \beta^{m_t}$, where m_t is the smallest nonnegative integer satisfying the inequality:

$$f(\mathbf{x}^t + \beta^m(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) + g(\mathbf{x}^t) + \beta^m(g(\mathbb{B}\mathbf{x}^t) - g(\mathbf{x}^t)) \le f(\mathbf{x}^t) + g(\mathbf{x}^t) + \alpha\beta^m\big(\nabla f(\mathbf{x}^t)^T(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t) + g(\mathbb{B}\mathbf{x}^t) - g(\mathbf{x}^t)\big). \quad (11)$$

This has a much lower complexity than the traditional successive line search scheme (9) because $g(\mathbb{B}\mathbf{x}^t)$ only needs to be calculated once.

The proposed variable update (6)-(7) with stepsizes given by (10) or (11) is summarized in Algorithm 1, and we name it as Soft-ThrEsholding with successive Line search Algorithm (STELA). In what follows, we draw comments on the important aspects of the proposed parallel STELA for problem (2).

Algorithm 1 The parallel STELA for the sparse nonlinear regression problem (2)

Data: $t=0$, $\mathbf{x}^0$ (arbitrary but fixed, e.g., $\mathbf{x}^0=\mathbf{0}$), stop criterion δ.
S1: Compute $\mathbb{B}\mathbf{x}^t$ according to (6a).
S2: Determine the stepsize γ^t by the exact line search (10) or the successive line search (11).
S3: Update $\mathbf{x}$ according to (7).
S4: If $|\nabla f(\mathbf{x}^t)^T(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)+g(\mathbb{B}\mathbf{x}^t)-g(\mathbf{x}^t)| < \delta$, STOP; otherwise $t \leftarrow t+1$ and go to **S1**.

On the convergence of STELA in Algorithm 1. It follows from [2, Theorem 1] that any limit point of the sequence $\{\mathbf{x}^t\}_t$ generated by STELA described in Algorithm 1 is a stationary point of problem (2). This statement is still valid when $\mathbf{H}$ is a positive definite matrix but not a diagonal matrix. In this case, however, the problem (5) cannot be decomposed among the elements of $\mathbf{x}$, and $\mathbb{B}\mathbf{x}^t$ must be found iteratively.

On the choice of H. One common choice of $\mathbf{H}^t$ is $\mathbf{H}^t = c\mathbf{I}$ where c is some given positive constant. Besides, if the matrix $\mathbf{H}^t$ contains second order information, the algorithm could be further accelerated. For example, if $f(\mathbf{x})$ is strictly convex in x_k (recall that $f(\mathbf{x})$ may not be convex in $\mathbf{x}$), then $H_{kk}^t = \nabla_{x_k}^2 f(\mathbf{x}^t) > 0$ for all k. If $f(\mathbf{x})$ is not convex in x_k, then $H_{kk}^t = \nabla_{x_k}^2 f(\mathbf{x}^t) + c^t$ and c must be large enough to guarantee that $\mathbf{H}^t \succ \mathbf{0}$, e.g., $c^t > |\min_k \nabla_{x_k}^2 f(\mathbf{x}^t)|$.

On the connection to ISTA. In ISTA proposed in [1], the variable is updated as follows:

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}} f(\mathbf{x}^t) + (\mathbf{x}-\mathbf{x}^t)\nabla f(\mathbf{x}^t) + \frac{c^t}{2}\left\|\mathbf{x}-\mathbf{x}^t\right\|^2 + \lambda\left\|\mathbf{x}\right\|_1. \tag{12}$$

This is a special case of Algorithm 1 when $\mathbf{H}^t = c^t\mathbf{I}$ in (4) and the stepsize $\gamma = 1$ in (7), and convergence is guaranteed if $c^t > L_{\nabla f}$, where $L_{\nabla f}$ is the Lipschitz constant of ∇f. To see this, it follows from the descent lemma [8] that $\tilde{f}(\mathbf{x};\mathbf{x}^t) \geq f(\mathbf{x})$ if $c^t > L_{\nabla f}$. According to [2], if the approximate function $\tilde{f}(\mathbf{x};\mathbf{x}^t)$ is a global upper bound of the original function $f(\mathbf{x})$, the convergence of Algorithm 1 is guaranteed under the constant unit stepsize, i.e., $\gamma^t = 1$ for all t.

On the complexity of STELA and ISTA. When the value of $L_{\nabla f}$ is not known, c^t should be estimated iteratively: for a constant $\beta > 1$, define $\mathbf{x}^\star(\beta^m)$ as

$$\mathbf{x}^\star(\beta^m) \triangleq \arg\min_{\mathbf{x}} \left\{ \begin{array}{l} f(\mathbf{x}^t) + (\mathbf{x}-\mathbf{x}^t)\nabla f(\mathbf{x}^t) \\ +\frac{\beta^m}{2}\left\|\mathbf{x}-\mathbf{x}^t\right\|^2 + \lambda\left\|\mathbf{x}\right\|_1 \end{array} \right\}. \tag{13}$$

Then $\mathbf{x}^{t+1} = \mathbf{x}^\star(\beta^{m_t})$ while m_t is the first nonnegative integer such that the following inequality is satisfied for some $\alpha \in (0,1)$:

$$\begin{aligned} &f(\mathbf{x}^\star(\beta^m)) + g(\mathbf{x}^\star(\beta^m)) \\ &< f(\mathbf{x}^t) + g(\mathbf{x}^t) - \alpha\beta^m\left\|\mathbf{x}^\star(\beta^m) - \mathbf{x}^t\right\|^2. \end{aligned}$$

In other words, $\mathbf{x}^\star(\beta^m)$ should be evaluated according to (13) for m_t times, namely, $m = 0, 1, \ldots, m_t$. This is however not necessary in the proposed algorithm STELA, because computing $\mathbb{B}\mathbf{x}^t$ according to (6) does not depend on any unknown parameters and the soft-thresholding operator only needs to be called once.

III. The Proposed Hybrid Successive Convex Approximation Algorithms

The Algorithm 1 proposed for (2) in the previous section is fully parallelizable. When the problem dimension is extremely large and there is only a limited number of processors/clusters, Algorithm 1 may not be fully parallelized due to hardware constraints. We could naively divide it into blocks which are then executed sequentially. However, when a particular block variable $\mathbf{x}_k$ is updated, the most recent updates of the previous block variables $(\mathbf{x}_j)_{j=1}^{k-1}$ will not be exploited. To address this issue, we design in this section a hybrid successive convex approximation algorithm.

We divide the variable $\mathbf{x}$ into K block variables $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K)$, such that Algorithm 1 could be fully parallelized when being applied to the following optimization problem w.r.t. the block variable $\mathbf{x}_k$ (rather than the full variable $\mathbf{x}$):

$$\min_{\mathbf{x}_k} \quad f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K) + \textstyle\sum_{k=1}^K g(\mathbf{x}_k). \tag{14}$$

If all block variables are updated sequentially (in a cyclic order) based on (14), and when one block variable $\mathbf{x}_k$ is being updated, the other block variables $\mathbf{x}_{-k} \triangleq (\mathbf{x}_j)_{j\neq k}$ are fixed, the resulting block coordinate descent algorithm converges to a stationary point of problem (2) under certain conditions [9]. However, the optimization problem (14) may not be easy to solve. One solution approach would be to apply Algorithm 1 to solve (14) iteratively, and the resulting algorithm will be of two layers, while the parallel STELA is in the inner layer and the block variables are sequentially alternated in the outer layer.

To reduce the complexity, we propose an iterative algorithm in which the block variable $\mathbf{x}_k$ is being updated, all elements of $\mathbf{x}_k$ are updated in parallel by solving an approximate problem that is much easier to optimize than the original problem (14). To start with, we reformulate problem (2) into the following equivalent one:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{y}}{\text{minimize}} \quad & f(\mathbf{x}_1, \ldots, \mathbf{x}_K) + \mathbf{1}^T\mathbf{y} \\ \text{subject to} \quad & g(\mathbf{x}_k) \leq y_k, \forall k. \end{aligned} \tag{15}$$

Suppose block variable $(\mathbf{x}_k, y_k)$ will be updated at iteration t. Define

$$(\mathbb{B}_k\mathbf{x}^t, y_k^\star(\mathbf{x}^t)) \triangleq \arg\min_{g(\mathbf{x}_k)\leq y_k} \tilde{f}(\mathbf{x}_k;\mathbf{x}^t) + y_k, \tag{16}$$

with $y_k^\star(\mathbf{x}^t) = g(\mathbb{B}_k\mathbf{x}^t)$. Note that $\tilde{f}(\mathbf{x}_k;\mathbf{x}^t)$ defined in (4) is an approximate function of $f(\mathbf{x}_k, \mathbf{x}_{-k}^t)$ at $\mathbf{x} = \mathbf{x}^t$ and it is not necessarily a global upper bound of $f(\mathbf{x})$. Moreover, $\mathbb{B}_k\mathbf{x}^t$ can be computed in closed-form by the soft-thresholding operator, cf. (6).

Since the objective function in (16) is convex, $(\mathbb{B}_k\mathbf{x}^t, y^\star(\mathbf{x}^t)) - (\mathbf{x}_k^t, y_k^t)$ is a descent direction of $f(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}^t$ along the coordinate of $\mathbf{x}_k$:

$$(\mathbb{B}_k\mathbf{x}^t - \mathbf{x}^t)\nabla\tilde{f}_k(\mathbf{x}_k;\mathbf{x}^t) + y^\star(\mathbf{x}^t) - y_k^t < 0.$$

Then $\mathbf{x}$ is updated according to the following expression: $\mathbf{x}^{t+1} = (\mathbf{x}_j^{t+1})_{j=1}^K$ and

$$\mathbf{x}_j^{t+1} = \begin{cases} \mathbf{x}_k^t + \gamma_k^t(\mathbb{B}_k\mathbf{x}^t - \mathbf{x}_k^t), & \text{if } j = k, \\ \mathbf{x}_j^t, & \text{otherwise.} \end{cases} \tag{17}$$

In other words, only the block variable $\mathbf{x}_k$ is updated according to (17) while other block variables $(\mathbf{x}_j)_{j\neq k}$ are equal to their value in the previous iteration. The stepsize γ_k^t in (17) could be determined efficiently by the line search introduced in the previous section, namely, either the exact line search

$$\min_{0\leq\gamma\leq 1} \left\{ f(\mathbf{x}_{-k}^t, \mathbf{x}_k^t + \gamma(\mathbb{B}_k\mathbf{x}^t - \mathbf{x}_k^t)) + \gamma(g(\mathbb{B}_k\mathbf{x}^t) - g(\mathbf{x}_k^t)) \right\}, \tag{18}$$

where $\mathbf{x}_{-k} \triangleq (\mathbf{x}_j)_{j\neq k}$, or the successive line search

$$\begin{aligned} &f(\mathbf{x}_k^t + \beta^m(\mathbb{B}_k\mathbf{x}^t - \mathbf{x}_k^t), \mathbf{x}_{-k}^t) + y_k^t + \beta^m(y^\star(\mathbf{x}^t) - y_k^t) + \sum_{j\neq k} y_j^t \\ \leq & f(\mathbf{x}^t) + \sum_{j=1}^K y_j^t + \alpha\beta^m(\nabla_k f(\mathbf{x}^t)^T(\mathbb{B}_k\mathbf{x}^t - \mathbf{x}_k^t) + y_k^\star(\mathbf{x}^t) - y_k^t) \end{aligned} \tag{19}$$

It may be tempting to update y_k^{t+1} as $y_k^{t+1} = y_k^t + \gamma^t(y^\star(\mathbf{x}^t) - y^t)$, but we propose the following update rule: $\mathbf{y}^{t+1} = (y_j^{t+1})_{j=1}^K$ and

$$y_j^{t+1} = \begin{cases} g(\mathbf{x}_k^t + \gamma^t(\mathbb{B}_k\mathbf{x}^t - \mathbf{x}_k^t)), & \text{if } j = k, \\ y_j^t, & \text{otherwise,} \end{cases} \tag{20}$$

because it yields a lower value:

$$\begin{aligned} y_k^{t+1} &\leq (1-\gamma^t)g(\mathbf{x}_k^t) + \gamma^t g(\mathbb{B}_k\mathbf{x}^t) \\ &\leq (1-\gamma^t)y_k^t + \gamma^t y^\star(\mathbf{x}^t) = y_k^t + \gamma^t(y^\star(\mathbf{x}^t) - y^t), \end{aligned}$$

Algorithm 2 The Hybrid STELA for the sparse nonlinear regression problem (2)

Data: $t = 0$, $\mathbf{x}^0$ (arbitrary but fixed, e.g., $\mathbf{x}^0 = \mathbf{0}$).
Repeat the following steps until convergence:
S1: Set $k = \mod(t, K) + 1$. Compute $(\mathbb{B}_k\mathbf{x}^t, y_k^\star(\mathbf{x}^t))$ according to (16).
S2: Determine the stepsize γ^t by the exact line search (18) or the successive line search (19).
S3: Update $\mathbf{x}$ and $\mathbf{y}$ according to (17) and (20), respectively.
S4: $t \leftarrow t + 1$ and go to **S1**.

where the first inequality is based on the convexity of $g(\mathbf{x})$ and the second inequality comes from the fact that $y^\star(\mathbf{x}^t) = g(\mathbb{B}_k\mathbf{x}^t)$ and $y_k^t \geq g(\mathbf{x}^t)$.

The proposed hybrid STELA is summarized in Algorithm 2, and its convergence properties are given in the following theorem.

Theorem 1. *Every limit point of the sequence $\{\mathbf{x}^t\}_t$ generated by the hybrid STELA in Algorithm 2 is a stationary point of (2).*

Proof: We need to show that every limit point of the sequence $\{\mathbf{x}^t, \mathbf{y}^t\}_t$ generated by Algorithm 2 is a stationary point of (15). We remark that the approximate function $\tilde{f}_k(\mathbf{x}_k; \mathbf{x}^t) + y_k$ satisfies the assumptions made in [3, Theorem 4], except that $\tilde{f}_k(\mathbf{x}_k; \mathbf{x}^t) + y$ is a convex function. However, the strict convexity assumed in [3, Theorem 4] is just an intermediate result and we show that the conclusion (cf. [3, Eq. (6.9)]) drawn from the strict convexity is still satisfied, namely, for any limit point $(\mathbf{x}, \mathbf{y})$ of the sequence generated by Algorithm 2, if $(\mathbb{B}_k\mathbf{x}, y_k^\star(\mathbf{x})) \neq (\mathbf{x}_k, y_k)$, then

$$\begin{aligned}&\tilde{f}_k(\mathbf{x}_k + \gamma(\mathbb{B}_k\mathbf{x} - \mathbf{x}_k); \mathbf{x}) + y_k + \gamma(y_k^\star(\mathbf{x}) - y_k)\\ >&\tilde{f}_k(\mathbf{x}_k; \mathbf{x}^t) + y_k + \gamma(\mathbb{B}_k\mathbf{x} - \mathbf{x}_k)^T\nabla\tilde{f}_k(\mathbf{x}_k; \mathbf{x}^t) + \gamma(y_k^\star(\mathbf{x}) - y_k),\end{aligned}$$

which, after removing the common terms on both sides of the inequality, is equivalent to

$$\tilde{f}_k(\mathbf{x}_k+\gamma(\mathbb{B}_k\mathbf{x}-\mathbf{x}_k); \mathbf{x}) > \tilde{f}_k(\mathbf{x}_k; \mathbf{x}^t)+\gamma(\mathbb{B}_k\mathbf{x}-\mathbf{x}_k)^T\nabla\tilde{f}_k(\mathbf{x}_k; \mathbf{x}^t). \quad (21)$$

To see this, we first discuss the three possible cases implied if $(\mathbb{B}_k\mathbf{x}, y_k^\star(\mathbf{x})) \neq (\mathbf{x}_k, y_k)$:

$$(\mathbb{B}_k\mathbf{x}, y_k^\star(\mathbf{x})) \neq (\mathbf{x}_k, y_k) \Leftrightarrow \begin{cases} \text{Case 1:} & \mathbb{B}_k\mathbf{x} \neq \mathbf{x}_k, y_k^\star(\mathbf{x}) \neq y_k, \\ \text{or Case 2:} & \mathbb{B}_k\mathbf{x} \neq \mathbf{x}_k, y_k^\star(\mathbf{x}) = y_k, \\ \text{or Case 3:} & \mathbb{B}_k\mathbf{x} = \mathbf{x}_k, y_k^\star(\mathbf{x}) \neq y_k. \end{cases}$$

Case 3 can be excluded because $y_k^\star = g(\mathbb{B}_k\mathbf{x})$ and it follows from the update rule of $\mathbf{y}$ in (20) and the Maximum Theorem in [10, VI. 3] that $\mathbf{y} = g(\mathbf{x})$. In view of Case 1 and Case 2, $(\mathbb{B}_k\mathbf{x}, y_k^\star(\mathbf{x})) \neq (\mathbf{x}_k, y_k)$ is equivalent to $\mathbb{B}_k\mathbf{x} \neq \mathbf{x}_k$. Then (21) follows directly from the strong (and thus strict) convexity of $\tilde{f}_k(\mathbf{x}_k; \mathbf{x}^t)$ w.r.t. $\mathbf{x}_k$:

$$\tilde{f}_k(\mathbf{x}_k+\gamma(\mathbb{B}_k\mathbf{x}-\mathbf{x}_k); \mathbf{x}) > \tilde{f}_k(\mathbf{x}_k; \mathbf{x})+\gamma(\mathbb{B}_k\mathbf{x}-\mathbf{x}_k)^T\nabla\tilde{f}_k(\mathbf{x}_k; \mathbf{x}^t).$$

Therefore, (21) is satisfied and the proof of Theorem 1 follows the same line of argument of [3, Theorem 4]. ■

The hybrid STELA in Algorithm 2 is complementary to the parallel STELA in Algorithm 1. On the one hand, the update of the elements of a particular block variable in the hybrid STELA is based on the same principle as the parallel STELA, namely, the construction of an approximate function and the line search scheme to calculate the stepsize. As a result, the hybrid STELA in Algorithm 2 has the same features as the parallel STELA in Algorithm 1. On the other hand, the approximate problem solved in each iteration of the hybrid STELA has a much smaller dimension than that of the parallel STELA. Therefore, a problem that is for example handled by a single memory/processing unit could be of a much larger size. Furthermore, due to the use of the line search schemes (18)-(19), the convergence speed of the proposed hybrid STELA is much faster than state-of-the-art hybrid algorithms whose convergence is only proved under constant and decreasing stepsizes [4, 5].

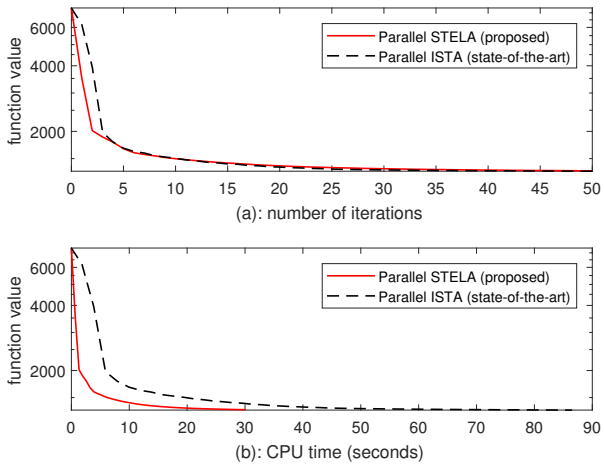

Figure 1. Nonlinear regression problem: Achieved objective function value versus the number of iterations and CPU time.

IV. Numerical Results

In this section, we perform numerical tests to compare the proposed parallel STELA with ISTA proposed in [1] and to illustrate the advantage of the hybrid STELA. All algorithms are tested under identical conditions in Matlab R2017a on a PC equipped with an operating system of Windows 10 64-bit, an Intel i7-7600U 2.80GHz CPU, and a 16GB RAM. All of the Matlab codes are available online at http://orbilu.uni.lu/handle/10993/35047.

The dimension of $\mathbf{A}$ and $\mathbf{x}$ is 50000×10000 and 50000×1, respectively. The elements of $\mathbf{A}$ are first generated according to the normal distribution, and each column is then normalized to unity. The density (the proportion of nonzero elements) of the sparse vector $\mathbf{x}_{\text{true}}$ is 0.1. The vector $\mathbf{y}$ is generated as $\mathbf{y} = \mathbf{A}^T\mathbf{x}_{\text{true}} + \mathbf{e}$ where $\mathbf{e}$ is drawn from an i.i.d. Gaussian distribution with variance 10^{-4}. The regularization gain λ is set to $\lambda = 0.1\|\mathbf{A}\mathbf{y}\|_\infty$, which allows $\mathbf{x}_{\text{true}}$ to be recovered to a high accuracy [2]. The simulation results are averaged over 10 repetitions.

We first test the performance of the proposed parallel STELA for the nonlinear regression problem (2) with $\sigma(x) = 2x + \cos(x)$ [1], while the benchmark algorithm is the parallel ISTA. The numerical result is shown in Figure 1, where the achieved objective function value versus the number of iterations and the CPU time (in seconds) is plotted in Figure 1 (a) and Figure 1 (b), respectively. Firstly, we see from Figure 1 (a) that the proposed parallel STELA converges faster in the first few iterations than the parallel ISTA. This is due to the fact that the proposed approximate function is not necessarily a global upper bound of the original function, and we have more freedom constructing the approximate function than in the parallel ISTA. Then we see from Figure 1 (b) that the improvement in terms of the required CPU time is notable: the proposed parallel STELA converges in 20 seconds and the parallel ISTA converges in 60 seconds. The resulting acceleration factor is 3, and this factor is consistent with the total CPU time for 50 iterations of the proposed

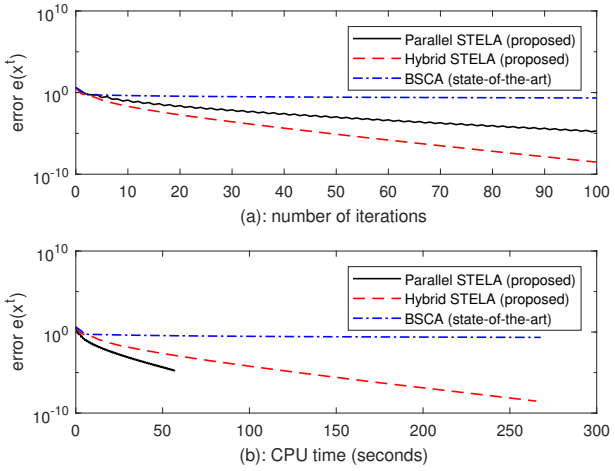

Figure 2. Linear regression problem (LASSO): Achieved error versus the number of iterations and CPU time.

parallel STELA and the parallel ISTA, namely from 30 seconds and 90 seconds. The acceleration in CPU time comes from the reduction in the complexity per iteration. In particular, the soft-thresholding operator is only called once in the proposed parallel STELA, while it must be called several times in the parallel ISTA because of the successive line search scheme to estimate $L_{\nabla f}$, as described in (13).

We then compare the parallel STELA, hybrid STELA, and the block successive convex approximation (BSCA) algorithm proposed in [4] for the linear regression problem (LASSO), i.e., $\sigma(x) = x$. In this case, the function $f(\mathbf{x}) = \sum_{n=1}^{N} \left(y_n - \mathbf{a}_n^T \mathbf{x}\right)^2$ is a quadratic function and (2) is convex. For the parallel STELA, we set $\mathbf{H}^t$ in (4) as the diagonal of $\mathbf{A}\mathbf{A}^T$ while $\mathbf{A} = [\mathbf{a}_1\, \mathbf{a}_2\, \ldots\, \mathbf{a}_N]$, and the stepsize is calculated by the exact line search as it has a closed-form expression, see [2, Sec. IV-III]. The approximate functions for the hybrid STELA and the BSCA algorithm are defined in the same manner, except that the approximation is w.r.t. a block variable, say $\mathbf{x}_k$ (rather than the whole set of variables $\mathbf{x}$ as in the parallel STELA).

Note that the hybrid STELA and the BSCA algorithm have the same approximate function, and their only difference lies in the stepsize: the stepsize of the hybrid STELA is obtained by the exact line search along the coordinate of the block variable being updated (cf. (18)), which has a closed-form expression, see [2, Sec. IV-III], and the stepsize of the BSCA algorithm is the decreasing stepsize $\gamma^t = 1/t$. We set the number of block variables $K = 5$, and the dimension of each block variable $\mathbf{x}_k$ is $50000/5 \times 1 = 10000 \times 1$. In one iteration of the hybrid STELA and the BSCA algorithm, all block variables are updated once in a sequential order.

As expected, it is shown in Figure 2 (b) that the hybrid STELA needs a longer CPU time to converge than the parallel STELA, because the block variables are updated sequentially. However, we see from Figure 2 (a) that the hybrid STELA needs much less number of iterations to converge. For example, the solution obtained by the parallel STELA after 100 iterations is already obtained by the hybrid STELA after 40 iterations. This is because when a particular block variable is updated, the latest information that becomes available from the updates of previous block variables is exploited. In applications where the sheer data volume is too big for a single memory/processing unit, employing the hybrid STELA yields a faster convergence than naively dividing the parallel STELA into blocks and executing them sequentially.

Comparing the proposed hybrid STELA and the BSCA algorithm, the notable acceleration in convergence speed brought by the line search is consolidated, as their only difference is the choice of the stepsize. We remark that the performance of the BSCA algorithm may be improved by a fine tuning of the stepsize's decreasing rate. However, this is a difficult task on its own and there is no universal choice that works well for all problem setups.

V. Concluding Remarks

In this paper, we have proposed a parallel algorithm for nondifferentiable nonconvex nonlinear regression problems. The proposed parallel algorithm consists in optimizing a sequence of successively refined approximate functions, and it has two attractive features which lead to a notable reduction in the algorithm's complexity: the proposed approximate function does not have to be a global upper bound of the original function, and the stepsize can be efficiently computed by the line search scheme which is carried out over a properly constructed differentiable function. Furthermore, when the parallel algorithm cannot be fully parallelized due to memory/processor constraints, we have proposed a hybrid updating scheme that divides the whole set of variables into blocks which are updated sequentially. Since the stepsize is obtained by performing the line search along the coordinate of each block variable, the proposed hybrid algorithm converges faster than state-of-the-art hybrid algorithms based on constant stepsizes and/or decreasing stepsizes. The advantages of the proposed algorithms are finally illustrated by numerical simulations.

References

[1] Z. Yang, Z. Wang, H. Liu, Y. C. Eldar, and T. Zhang, "Sparse Nonlinear Regression: Parameter Estimation and Asymptotic Inference," 2016, in Proc. International Conference on Machine Learning (ICML). [Online]. Available: http://proceedings.mlr.press/v48/yangc16.pdf

[2] 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.

[3] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A Unified Convergence Analysis of Block Successive Minimization Methods for Nonsmooth Optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1126–1153, Jan. 2013.

[4] M. Razaviyayn, M. Hong, Z.-Q. Luo, and J.-S. Pang, "Parallel Successive Convex Approximation for Nonsmooth Nonconvex Optimization," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, 2014, pp. 1440–1448.

[5] F. Facchinei, G. Scutari, and S. Sagratella, "Parallel Selective Algorithms for Nonconvex Big Data Optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 7, pp. 1874–1889, Nov. 2015.

[6] K. Slavakis, G. B. Giannakis, and G. Mateos, "Modeling and Optimization for Big Data Analytics: (Statistical) learning tools for our era of data deluge," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 18–31, Sep. 2014.

[7] Y. Yang and M. Pesavento, "A parallel best-response algorithm with exact line search for nonconvex sparsity-regularized rank minimization," Apr. 2018, to appear in Proc. ICASSP. [Online]. Available: http://orbilu.uni.lu/handle/10993/33772

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

[9] P. Tseng, "Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization," *Journal of Optimization Theory and Applications*, vol. 109, no. 3, pp. 475–494, Jun. 2001.

[10] C. Berge, *Topological Spaces: Including a Treatment of Multi-Valued Functions, Vector Spaces and Convexity*. Dover Publications, 1997.