A NOVEL LINE SEARCH METHOD FOR NONSMOOTH OPTIMIZATION PROBLEMS

Yang Yang and Marius Pesavento

Communication Systems Group, Darmstadt University of Technology, Darmstadt, Germany.
Emails: {yang, pesavento}@nt.tu-darmstadt.de

ABSTRACT

In this paper, we propose a novel exact/successive line search method for stepsize calculation in iterative algorithms for nonsmooth optimization problems. The proposed approach is to perform line search over a properly constructed differentiable function based on the original nonsmooth objective function, and it outperforms state-of-the-art techniques from the perspective of convergence speed, computational complexity and signaling burden. When applied to LASSO, the proposed exact line search is shown, either analytically or numerically, to exhibit several desirable advantages, namely: it is implementable in closed-form, converges fast and is robust with respect to the choice of problem parameters.

Index Terms— Descent Direction Method, Distributed and Parallel Algorithms, LASSO, Line Search, Nondifferentiable Optimization, Successive Convex Approximation

1. INTRODUCTION

In this paper, we consider iterative algorithms that solve the following optimization problem:

$$\begin{aligned} \underset{\mathbf{x}=(\mathbf{x}_k)_{k=1}^K}{\text{minimize}} \quad & U(\mathbf{x}) \triangleq f(\mathbf{x}_1,\ldots,\mathbf{x}_K) + \textstyle\sum_{k=1}^K g_k(\mathbf{x}_k), \\ \text{subject to} \quad & \mathbf{x}_k \in \mathcal{X}_k, k=1,\ldots,K, \end{aligned} \tag{1}$$

We make the following assumptions on problem (1):
(A1) The function $U(\mathbf{x})$ is coercive (i.e., $U(\mathbf{x}) \to +\infty$ if $\|\mathbf{x}\| \to +\infty$), $f(\mathbf{x})$ is differentiable, and $g_k(\mathbf{x}_k)$ is convex but not necessarily smooth.
(A2) The gradient $\nabla f(\mathbf{x})$ is Lipschitz continuous with a constant L (i.e., $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \le L\|\mathbf{x}-\mathbf{y}\|, \forall \mathbf{x},\mathbf{y} \in \mathcal{X}$).
(A3) The set $\mathcal{X}_k$ is nonempty, closed and convex.
Note that the coercivity of $U(\mathbf{x})$ implies that every lower level set of $U(\mathbf{x})$ is bounded.

We do not assume that $f(\mathbf{x})$ is convex, so problem (1) is in general nonconvex. Since the constraint set in (1) is uncoupled among different variables $\mathbf{x}_k$'s and changing one variable does not affect the feasibility of other variables, problem (1) is suitable for distributed computation [1, 2]. The block coordinate descent (BCD) method [1, Sec. 2.7] is such an example: in each iteration, only one variable is updated by the solution that minimizes $f(\mathbf{x})$ with respect to (w.r.t.) that variable while the remaining variables are fixed, and the variables are updated sequentially. This method has been applied in many practical problems, see [3] and the references therein.

On the other hand, BCD may suffer from slow convergence due to the sequential update, especially when the number of variables is large. Parallel update seems more desirable, but they converge under rather restrictive conditions, for example, under the diagonal dominance condition on the objective function $f(\mathbf{x})$ [2] or the condition that the stepsize is sufficiently small [1]. A recent progress in parallel algorithms has been made in [4] and it was shown that the stationary points of (1) can be found by solving a sequence of successively refined approximate problems. Note that this algorithm is essentially an iterative descent direction method and convergence is established if, among other conditions, the approximate function and stepsizes are properly selected.

Despite its novelty, the parallel algorithm proposed in [4] suffers from a limitation, namely, decreasing stepsizes must be used. On the one hand, a slowly decaying 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 that gives a good trade-off between convergence guarantee and convergence speed and current practices mainly rely on heuristics which are however sensitive to the problem parameters. It is possible to employ a constant stepsize and successive line search to determine the stepsize, but the former suffers from a slow convergence while the complexity to implement the latter is usually very high because the objective function in (1) is nonsmooth.

The contribution of this paper consists in the development of a novel exact/successive line search procedure for nonsmooth problems. In particular, the proposed line search is carried out over a properly constructed differentiable function. Thus it is much easier to implement than state-of-the-art line search methods that operate on the nonsmooth objective function in (1) directly. If $f(\mathbf{x})$ exhibits a specific structure, such as in convex quadratic functions, it is further possible to perform the exact/successive line search in closed-form with an affordable level of signaling exchange among different processors when implemented in a distributed manner. Besides, the proposed line search method typically yields faster convergence than decreasing stepsizes and it is robust to parameter changes. The advantages of the proposed algorithm will be demonstrated in the example of the popular LASSO problem.

This work is supported by the Seventh Framework Programme for Research of the European Commission under grant number ADEL 619647.

978-0-9928626-3-3/15/$31.00 ©2015 IEEE

2. STATE-OF-THE-ART SUCCESSIVE CONVEX APPROXIMATION FRAMEWORK

To begin with, we introduce the iterative algorithm based on the successive convex approximation framework proposed in [4]. In particular, problem (1) is solved as a sequence of successively refined approximate problems, each of which is presumably much easier to solve than the original problem (1). At iteration t, given the point $\mathbf{x} = \mathbf{x}^t$ generated in the previous iteration, the approximate problem is

$$\underset{(\mathbf{x}_k \in \mathcal{X}_k)_{k=1}^K}{\text{minimize}} \quad \sum_{k=1}^K (\tilde{f}_k(\mathbf{x}_k; \mathbf{x}^t) + g_k(\mathbf{x}_k)), \tag{2}$$

where $\sum_{k=1}^K \tilde{f}_k(\mathbf{x}_k; \mathbf{x}^t)$ is the approximate function of $f(\mathbf{x})$ around $\mathbf{x} = \mathbf{x}^t$ and it satisfies several technical conditions [4]:
(B1) The function $\tilde{f}_k(\mathbf{x}_k; \mathbf{y})$ is strongly convex in $\mathbf{x}_k \in \mathcal{X}_k$ with some constant $c > 0$ for any given $\mathbf{y} \in \mathcal{X}$;
(B2) The function $\tilde{f}_k(\mathbf{x}_k; \mathbf{y})$ is continuously differentiable in $\mathbf{x}_k \in \mathcal{X}_k$ for a fixed $\mathbf{y}$ and Lipschitz continuous in $\mathbf{y} \in \mathcal{X}$ for a fixed $\mathbf{x}_k$, and $\nabla_{\mathbf{x}_k} \tilde{f}_k(\mathbf{x}_k; \mathbf{x}) = \nabla_{\mathbf{x}_k} f(\mathbf{x})$;

If $f(\mathbf{x}_1, \ldots, \mathbf{x}_K)$ is convex in $\mathbf{x}_k$ for all k (but not necessarily jointly convex), an approximate function satisfying the above Assumptions (B1)-(B2) is given by:

$$\tilde{f}_k(\mathbf{x}_k; \mathbf{x}^t) = f(\mathbf{x}_k, \mathbf{x}^t_{-k}) + \tfrac{c}{2} \left\| \mathbf{x}_k - \mathbf{x}_k^t \right\|^2, \tag{3}$$

where $\mathbf{x}_{-k} \triangleq (\mathbf{x}_j)_{j \neq k}$ and c is a positive constant.

Since (2) is uncoupled among different variables $\mathbf{x}_k$ in both the objective function and the constraint set, it can be solved in parallel. Let us define the operator $\mathbb{B}_k \mathbf{x}$ as

$$\mathbb{B}_k \mathbf{x}^t \triangleq \underset{\mathbf{x}_k \in \mathcal{X}_k}{\arg\min} \left\{ \tilde{f}_k(\mathbf{x}_k; \mathbf{x}^t) + g_k(\mathbf{x}_k) \right\}. \tag{4}$$

It was shown in [4, Prop. 8(c)] that $\mathbb{B}\mathbf{x}^t - \mathbf{x}^t$ where $\mathbb{B}\mathbf{x} = (\mathbb{B}_k \mathbf{x})_{k=1}^K$ is a descent direction of (1), along which $U(\mathbf{x})$ can be decreased compared with $U(\mathbf{x}^t)$. The next point $\mathbf{x}^{t+1}$ is thus defined as:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \gamma^t (\mathbb{B}\mathbf{x}^t - \mathbf{x}^t), \tag{5}$$

where $\gamma^t \in (0, 1]$ is an appropriate stepsize that can be determined by the following standard rules.

Decreasing stepsize [1, 4, 5]: The sequence $\{\gamma^t\}$ decreases but it does not decrease too quickly in the following sense:

$$\gamma^t \to 0, \ \sum_{t=1}^\infty \gamma^t = \infty. \tag{6}$$

Examples satisfying (6) include

$$\gamma^{t+1} = \gamma^t (1 - d\gamma^t), \tag{7}$$

where $d \in (0, 1)$ is the so-called decreasing rate and it controls how fast the stepsize γ^t decreases.

Exact line search [1, 5, 6]: The stepsize γ^t decreases the objective function $U(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t))$ to the largest extent:

$$\gamma^t \triangleq \underset{0 \leq \gamma \leq 1}{\arg\min} \left\{ \begin{matrix} f(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) \\ + \sum_{k=1}^K g_k(\mathbf{x}_k^t + \gamma(\mathbb{B}_k \mathbf{x}^t - \mathbf{x}_k^t)) \end{matrix} \right\}. \tag{8}$$

Successive line search [4, 5]: Given constants $\alpha, \beta \in (0, 1)$, the stepsize γ^t is set to be $\gamma^t = \beta^{m_t}$, where m_t is the smallest nonnegative integer m satisfying the following inequality:

$$\begin{aligned} &f(\mathbf{x}^t + \beta^m(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) + \textstyle\sum_{k=1}^K g_k(\mathbf{x}_k^t + \beta^m(\mathbb{B}_k \mathbf{x}^t - \mathbf{x}_k^t)) \\ &\quad \leq f(\mathbf{x}^t) + \textstyle\sum_{k=1}^K g_k(\mathbf{x}_k^t) - \alpha\beta^m c \left\| \mathbb{B}\mathbf{x}^t - \mathbf{x}^t \right\|^2. \end{aligned} \tag{9}$$

It was shown in [4, Th. 1] that any limit point of the sequence $\{\mathbf{x}^t\}$ generated by (5) with any of the stepsize rules (6)-(9) is a stationary point of (1). However, despite this attractive convergence guarantee, the stepsize rules (7)-(9) suffer from several practical limitations. Firstly, it is difficult to find a decreasing rate d in (7) that gives a good trade-off between convergence guarantee and convergence speed. Secondly, the exact line search (8) involves a nonsmooth optimization problem which is usually computationally expensive to solve. Thirdly, the successive line search (9) involves repeated evaluation of both the differentiable function $f(\mathbf{x})$ and the nonsmooth functions $g_k(\mathbf{x}_k)$ for $k = 1, \ldots, K$, which may consume considerable computational resources and incur a lot of signaling exchange among different processors when implemented in a distributed manner.

3. THE PROPOSED LINE SEARCH METHOD

In this section, we propose a novel exact/successive line search method that overcomes the shortcomings of state-of-the-art techniques. Firstly note that the main difficulty in (8) is the nonsmooth function $g_k(\mathbf{x}_k)$ which makes the objective function in (8) nonsmooth. As $g_k(\mathbf{x}_k)$ is convex, Jensen's inequality implies that for any $\gamma \in [0, 1]$:

$$\begin{aligned} g_k(\mathbf{x}_k^t + \gamma(\mathbb{B}_k \mathbf{x}^t - \mathbf{x}_k^t)) &\leq (1 - \gamma) g_k(\mathbf{x}_k^t) + \gamma g_k(\mathbb{B}_k \mathbf{x}^t) \\ &= \gamma(g_k(\mathbb{B}_k \mathbf{x}^t) - g_k(\mathbf{x}_k^t)) + g_k(\mathbf{x}_k^t). \end{aligned} \tag{10}$$

The function in (10) is a linear function of γ and it is an upper bound of $g_k(\mathbf{x}_k^t + \gamma(\mathbb{B}_k \mathbf{x}^t - \mathbf{x}_k^t))$ which is exact at either $\gamma = 0$ or $\gamma = 1$. Then it readily follows that for any $\gamma \in [0, 1]$:

$$\begin{aligned} &U(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) - U(\mathbf{x}^t) \\ &= f(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) - f(\mathbf{x}^t) \\ &\quad + \textstyle\sum_{k=1}^K (g_k(\mathbf{x}_k^t + \gamma(\mathbb{B}_k \mathbf{x}^t - \mathbf{x}_k^t)) - g_k(\mathbf{x}_k^t)) \\ &\leq f(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) - f(\mathbf{x}^t) \\ &\quad + \gamma(\textstyle\sum_{k=1}^K g_k(\mathbb{B}_k \mathbf{x}^t) - g_k(\mathbf{x}_k^t)) && \text{(11a)} \\ &\leq \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)^T \nabla f(\mathbf{x}^t) + \tfrac{L}{2}\gamma^2 \left\| \mathbb{B}\mathbf{x}^t - \mathbf{x}^t \right\|^2 \\ &\quad + \gamma \textstyle\sum_{k=1}^K (g_k(\mathbb{B}_k \mathbf{x}^t) - g_k(\mathbf{x}_k^t)) && \text{(11b)} \\ &\leq -\gamma(c - \tfrac{L}{2}\gamma) \left\| \mathbb{B}\mathbf{x}^t - \mathbf{x}^t \right\|^2, && \text{(11c)} \end{aligned}$$

where (11a), (11b), and (11c) comes from (10), the descent lemma [1, Prop. A.24] while L is given in Assumption (A2), and [4, Prop. 8(c)], respectively. For any γ that is sufficiently small, the term in (11c) is negative, which implies that the term in (11a) is negative and $U(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) < U(\mathbf{x}^t)$.

Proposed simplied exact line search: Instead of directly minimizing $U(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) - U(\mathbf{x}^t)$ (or equivalently $U(\mathbf{x}^t+\gamma(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))$) over γ as in (8), we propose to minimize its (tight) upper bound function in (11a):

$$\gamma^t = \arg\min_{0\le\gamma\le 1} \left\{ \begin{array}{l} f(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) \\ \quad +\gamma\sum_{k=1}^K (g_k(\mathbb{B}_k\mathbf{x}^t) - g_k(\mathbf{x}_k^t)) \end{array} \right\}, \quad (12)$$

where the constant term $f(\mathbf{x}^t)$ is discarded without loss of generality. Problem (12) has a differentiable objective function with a scalar variable and a bound constraint, so the vast variety of numerical algorithms for differentiable problems in literature can readily be applied [1].

To simplify the discussion, we define the objective function in (12) as $h^t(\gamma)$:

$$h^t(\gamma) \triangleq f(\mathbf{x}^t+\gamma(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))+\gamma\textstyle\sum_{k=1}^K (g_k(\mathbb{B}_k\mathbf{x}^t)-g_k(\mathbf{x}_k^t)). \quad (13)$$

If $f(\mathbf{x})$ is *convex* in $\mathbf{x}$ and $\gamma^\star$ nulls the gradient of $h^t(\gamma)$: $\nabla_\gamma h^t(\gamma^\star) = 0$, then γ^t in (12) is simply the projection of $\gamma^\star$ onto the interval $[0, 1]$:

$$\gamma^t = \begin{cases} 1, & \text{if } \nabla_\gamma h^t(\gamma)|_{\gamma=1} \le 0, \\ 0, & \text{if } \nabla_\gamma h^t(\gamma)|_{\gamma=0} \ge 0, \\ \gamma^\star, & \text{otherwise.} \end{cases}$$

It is sometimes possible to compute $\gamma^\star$ analytically, e.g., $f(\mathbf{x})$ is convex quadratic in $\mathbf{x}$. If not, $\gamma^\star$ can be found efficiently by bisection method: since $h^t(\gamma)$ is convex in γ, it follows that $\nabla_\gamma h^t(\gamma) < 0$ if $\gamma < \gamma^\star$ and $\nabla_\gamma h^t(\gamma) > 0$ if $\gamma > \gamma^\star$. Then given an interval $[\gamma_{\text{low}}, \gamma_{\text{up}}]$ containing $\gamma^\star$ (the initial value of γ_{low} and γ_{up} is 0 and 1, respectively), we set $\gamma_{\text{mid}} = (\gamma_{\text{low}} + \gamma_{\text{up}})/2$ and refine γ_{low} and γ_{up} as follows: $\gamma_{\text{low}} = \gamma_{\text{mid}}$ if $\nabla_\gamma h^t(\gamma_{\text{mid}}) > 0$ or $\gamma_{\text{up}} = \gamma_{\text{mid}}$ otherwise. This procedure is repeated for a finite number of times until the gap $\gamma_{\text{up}} - \gamma_{\text{low}}$ is smaller than a prescribed precision.

Proposed simplified successive line search: If no structure in $f(\mathbf{x})$ (e.g., convexity) can be exploited to efficiently compute γ^t according to the exact line search (12), the successive line search can instead be employed. In the proposed simplified successive line search, instead of directly search over $U(\mathbf{x}^t+\gamma(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))-U(\mathbf{x})$ as in (9), we set $\gamma^t = \beta^{m_t}$, where m_t is the smallest nonnegative integer m for which the following inequality is satisfied:

$$\begin{aligned} f(\mathbf{x}^t + \beta^m(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) + \beta^m(\textstyle\sum_{k=1}^K g_k(\mathbb{B}_k\mathbf{x}^t) - g_k(\mathbf{x}_k^t)) \\ \le f(\mathbf{x}^t) - \alpha\beta^m c \left\|\mathbb{B}\mathbf{x}^t - \mathbf{x}^t\right\|^2. \end{aligned} \quad (14)$$

In other words, the proposed successive line search is carried out over $h^t(\lambda)$, i.e., the tight upper bound of $U(\mathbf{x}^t+\gamma(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))-U(\mathbf{x}^t)$. Such a constant m_t always exists and we derive its upper bound: (11c) indicates that (14) is satisfied if

$$\begin{aligned} &-\beta^m(c - \tfrac{L}{2}\beta^m)\left\|\mathbb{B}\mathbf{x}^t - \mathbf{x}^t\right\|^2 \le -\alpha\beta^m c\left\|\mathbb{B}\mathbf{x}^t - \mathbf{x}^t\right\|^2 \\ &\iff 0 \le \beta^m \le \tfrac{2(1-\alpha)c}{L} \iff m \ge \log_\beta(2(1-\alpha)c/L), \end{aligned}$$

so $m_t \le \lceil \log_\beta(2(1-\alpha)c/L) \rceil$, where $\lceil a \rceil$ is the smallest integer that is larger than a.

The new successive line search (14) involves the evaluation of the differentiable function $f(\mathbf{x})$ only and it outperforms, from the perspective of both computational complexity and signaling exchange, state-of-the-art techniques (9) in which the objective function $f(\mathbf{x}) + g(\mathbf{x})$ must be repeatedly evaluated (for different values of m) and compared with a certain benchmark until m_t is found.

Theorem 1. *Any limit point of the sequence $\{\mathbf{x}^t\}$ generated by (5) with the simplified exact line search (12) or the simplifed successive line search (14) is a stationary point of (1).*

Sketch of proof: The idea of the proof is to show that the update (5) with either (12) or (14) yields a larger decrease in the objective function $U(\mathbf{x})$ in each iteration than the decreasing stepsize (6) does. Then the convergence readily follows from the convergence of (5) under the decreasing stepsize (6) which was proved in [4, Th. 1]. Due to space limitations we omit the detailed steps and refer to [7]. ■

Theorem 1 establishes that there is no loss of convergence if the line search is carried out over $h^t(\gamma)$, a differentiable function constructed based on the fundamental property of convex functions. Thus the proposed line search methods (12) and (14) generally yields faster convergence than state-of-the-art decreasing stepsizes (6) and are easier to implement than state-of-the-art line search techniques (8) and (9), as will be illustrated in the next section for the example of the LASSO problem in sparse signal estimation.

4. EXAMPLE APPLICATION: LASSO

In this section, we specialize the proposed algorithm to solve the LASSO problem, an important and widely studied problem in sparse signal estimation [4, 8–12]:

$$\text{minimize} \quad U(\mathbf{x}) \triangleq \tfrac{1}{2}\left\|\mathbf{A}\mathbf{x} - \mathbf{b}\right\|_2^2 + \mu\left\|\mathbf{x}\right\|_1, \quad (15)$$

where $\mathbf{A} \in \mathbb{R}^{N\times K}$ (with $N \ll K$), $\mathbf{b} \in \mathbb{R}^{K\times 1}$ and $\mu > 0$ are given parameters. Problem (15) is convex, but the objective function is nonsmooth and cannot be minimized in closed-form. We thus apply the proposed iterative algorithm.

To begin with, scalar decomposition of $\mathbf{x}$ is adopted, i.e., $\mathbf{x} = (x_k)_{k=1}^K$. Define $f(\mathbf{x}) \triangleq \frac{1}{2}\left\|\mathbf{A}\mathbf{x} - \mathbf{b}\right\|_2^2$ and $g_k(x_k) \triangleq \mu|x_k|$. The approximate problem for x_k is (cf. (3))

$$\begin{aligned} \mathbb{B}_k\mathbf{x}^t &= \arg\min_{x_k} \left\{f(x_k, \mathbf{x}_{-k}^t) + \tfrac{c}{2}(x_k - x_k^t)^2 + g_k(x_k)\right\} \\ &= (d_k + c)^{-1}\mathcal{S}_\mu(r_k(\mathbf{x}^t) + c\, x_k^t), k = 1, \ldots, K, \end{aligned} \quad (16)$$

where $\mathbf{x}_{-k}^t \triangleq (x_j^t)_{j\ne k}$, $\mathcal{S}_a(b) \triangleq (b-a)^+ - (-b-a)^+$ with $(a)^+ \triangleq \max(a, 0)$ is the well-known soft-thresholding operator [6, 9], $\mathbf{d} \triangleq \text{diag}(\mathbf{A}^T\mathbf{A})$ and

$$\mathbf{r}(\mathbf{x}) \triangleq \mathbf{d} \circ \mathbf{x} - \mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) \quad (17)$$

with $\mathbf{a} \circ \mathbf{b}$ denoting the Hadamard product between $\mathbf{a}$ and $\mathbf{b}$. Then employing our proposed exact line search (12) yields

$$\gamma^t = \arg\min_{0\le\gamma\le 1} \left\{ \begin{array}{l} \frac{1}{2}\left\|\mathbf{A}(\mathbf{x}^t+\gamma(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))-\mathbf{b}\right\|_2^2 \\ +\gamma\mu\left(\left\|\mathbb{B}\mathbf{x}^t\right\|_1-\left\|\mathbf{x}^t\right\|_1\right) \end{array} \right\} \quad (18)$$

$$= \left[-\frac{(\mathbf{A}\mathbf{x}^t-\mathbf{b})^T(\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))+\mu(\left\|\mathbb{B}\mathbf{x}^t\right\|_1-\left\|\mathbf{x}^t\right\|_1)}{(\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))^T(\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))}\right]_0^1. \quad (19)$$

The optimization problem in (18) is convex quadratic with a closed-form solution (19), where $[x]_a^b \triangleq \max(\min(x,b),a)$. Therefore, with (16) and (19), all elements of $\mathbf{x}$ are updated *in parallel* and in *closed-form*. In contrast with the decreasing stepsize scheme used in [4], the stepsize based on the exact line search (16) and (19) yields notable progress in all iterations and the convergence speed is thus greatly enhanced. We name the proposed update (16) and (19) as Soft-Thresholding with simplified Exact Line search Algorithm (STELA).

Similarly, to illustrate the advantage of the proposed successive line search (14), we remark that m_t can also be calculated in closed-form. Therefore, if the line search has to be performed distributedly, the signaling exchange only needs to be carried out once among different processors, which is a much less overhead than in state-of-the-art techniques (9). We omit the details due to space limitations and refer to [7].

Computational complexity: The computational overhead associated with the proposed exact line search (19) can significantly be reduced if it is carefully implemented as outlined in the following. The most complex operation in (19) is the matrix-vector multiplication, namely, $\mathbf{A}\mathbf{x}^t-\mathbf{b}$ in the numerator and $\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)$ in the denominator. On one hand, $\mathbf{A}\mathbf{x}^t-\mathbf{b}$ is already available from the computation of $\mathbf{r}(\mathbf{x}^t)$ in (16)-(17). On the other hand, the product $\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)$ is also required to compute $\mathbf{A}\mathbf{x}^{t+1}-\mathbf{b}$ in the following iteration of the update (17) as it can alternatively be computed as:

$$\mathbf{A}\mathbf{x}^{t+1}-\mathbf{b} = (\mathbf{A}\mathbf{x}^t-\mathbf{b})+\gamma^t\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t). \quad (20)$$

As a result, (19) does not incur additional matrix-vector multiplications, but only affordable vector-vector multiplications.

Signaling exchange: When $\mathbf{A}$ is too large to be stored and processed by a centralized architecture, a parallel hardware architecture can be employed. Assume there are P processors and partition $\mathbf{A}$ and $\mathbf{x}$ as $\mathbf{A}=[\mathbf{A}_1\ \mathbf{A}_2\ \ldots\ \mathbf{A}_P]$ and $\mathbf{x}=(\mathbf{x}_p)_{p=1}^P$, where $\mathbf{A}_p \in \mathbb{R}^{N\times K_p}$ ($\sum_{p=1}^P K_p = K$) and $\mathbf{x}_p \in \mathbb{R}^{K_p\times 1}$ are stored and processed locally in processor p.

We remark that the level of signaling exchange to calculate (16) and (19) is as same as [4], where the update direction is found by (16) and the predetermined decreasing stepsize (7) is used. In other words, the line search (19) does not incur any additional signaling: in (19), since $\mathbf{A}\mathbf{x}^t-\mathbf{b}$ is already available when calculating $\mathbf{r}(\mathbf{x}^t)$, different processors only need to exchange $\mathbf{A}_p(\mathbb{B}_p\mathbf{x}^t-\mathbf{x}_p^t)$ (to form $\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)$) and $\left\|\mathbb{B}_p\mathbf{x}^t\right\|_1-\left\|\mathbf{x}_p^t\right\|_1$. On the one hand, $\mathbf{A}_p(\mathbb{B}_p\mathbf{x}^t-\mathbf{x}_p^t)$ has to be exchanged anyway to compute $\mathbf{A}\mathbf{x}^{t+1}-\mathbf{b}$ in (20) so that $\mathbb{B}\mathbf{x}^{t+1}$ can be computed according to (16) in the next iteration. On the other hand, $\left\|\mathbb{B}_p\mathbf{x}^t\right\|_1-\left\|\mathbf{x}_p^t\right\|$ is a scalar that has to be exchanged among different processors, but this is also required (in terms of $\left\|\mathbf{x}_p^t+\gamma^t(\mathbb{B}\mathbf{x}^t-\mathbf{x}_p^t)\right\|_1$) in [4] to calculate $U(\mathbf{x}^t)$ in each iteration, see [4, Sec. VI-A].

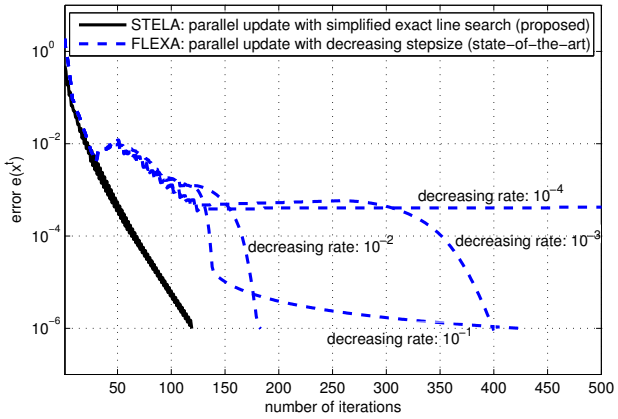

Fig. 1. Convergence of STELA (proposed) and FLEXA (state-of-the-art) for LASSO.

Simulations: We first compare in Fig. 1 the proposed algorithm STELA with FLEXA [4] in terms of error defined as $e(\mathbf{x}^t) = \left\|\nabla f(\mathbf{x}^t)-[\nabla f(\mathbf{x}^t)-\mathbf{x}^t]_{-\mu\mathbf{1}}^{\mu\mathbf{1}}\right\|_1$. Note that $\mathbf{x}^\star$ is a solution of (15) if and only if $e(\mathbf{x}^\star)=0$. In STELA, we set $c=0$ in (16); this does not violate the convergence result because $d_k=\sum_{n=1}^N A_{nk}^2>0$ for all k. For FLEXA, it is implemented as same as [4] except that all elements are updated in parallel and the selective update scheme is not employed, because it cannot overcome the bottleneck of the decreasing stepsize. We also remark that the stepsize rule for FLEXA is $\gamma^{t+1}=\gamma^t(1-\min(1,10^{-4}/e(\mathbf{x}^t))d\gamma^t)$ with $\gamma^0=0.9$ while d is the decreasing rate. The code and data generating Fig. 1 (and later Fig. 2) is available online [13].

Note that the error $e(\mathbf{x}^t)$ plotted in Fig. 1 needs not monotonically decrease (but $U(\mathbf{x}^t)$ does) because both STELA and FLEXA are descent direction methods. For FLEXA, when the decreasing rate is low ($d=10^{-4}$), no improvement is observed after 100 iterations. Similar behavior is also observed for $d=10^{-3}$, until the stepsize becomes small enough. When the stepsize is quickly decreasing ($d=10^{-1}$), although improvement is made in all iterations, the asymptotic convergence speed is slow because the stepsize is too small to make notable improvement. For this example, $d=10^{-2}$ works well, but the value of a good decreasing rate is parameters dependent (e.g., $\mathbf{A}$, $\mathbf{b}$ and μ) and no general rule works well for all choices of parameters. By comparison, STELA is fast to converge and robust w.r.t. the choice of parameters.

We also compare in Fig. 2 the proposed algorithm STELA with FISTA [9], ADMM [10], GreedyBCD [12] and SpaRSA [11]. We simulated GreedyBCD out of [12] because it is the one that has guaranteed convergence. The dimension of $\mathbf{A}$ is 2000×4000 and 5000×10000 (the left and right column of Fig. 2, respectively). The density of $\mathbf{x}_{\text{true}}$ is 0.1, 0.2 and 0.4 (the upper, middle and lower row of Fig. 2, respectively). The

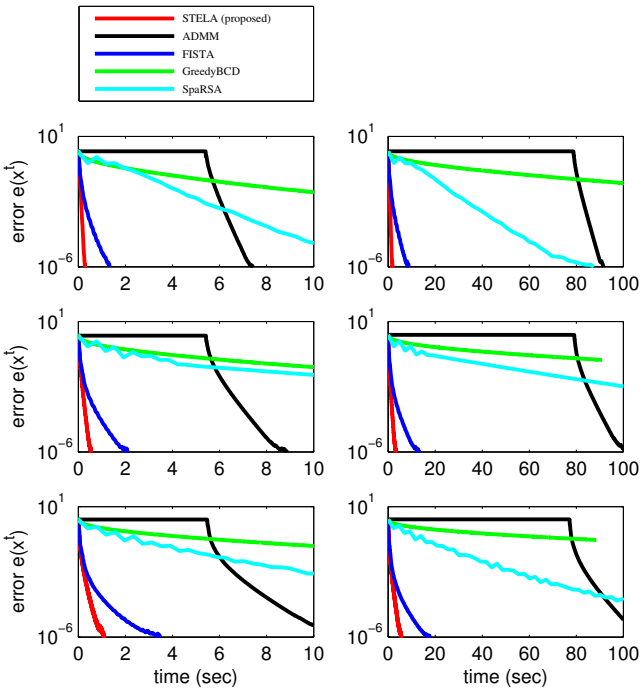

Fig. 2. Time versus error of different algorithms for LASSO. In the left and right column, the dimension of $\mathbf{A}$ is 2000×4000 and 5000×10000, respectively. In the higher, middle and lower column, the density of $\mathbf{x}_{\text{true}}$ is 0.1, 0.2 and 0.4.

hardware/software environment is specified in [7].

The comparison is in terms of CPU time either a given error bound $e(\mathbf{x}^t) \leq 10^{-6}$ is reached or the maximum number of iterations, namely, 2000, is reached. The running time consists of both the initialization stage (represented by a flat curve) and the formal stage. For example, in `STELA`, $\mathbf{d}(\mathbf{A}^T\mathbf{A})$ must be computed in the initialization stage.

It is easy to see from Fig. 2 that the proposed algorithm `STELA` converges faster than other algorithms. Some more comments follow. i) The proposed algorithm `STELA` is not sensitive to the density of the true signal $\mathbf{x}_{\text{true}}$. When the density is increased from 0.1 (left column) to 0.2 (middle column) and then to 0.4 (right column), the CPU time is increased negligibly. ii) The proposed algorithm `STELA` is relatively scalable w.r.t. the problem dimension. When the dimension of $\mathbf{A}$ is increased, the CPU time is only marginally increased. iii) The initilization stage of `ADMM` is time consuming because of some expensive matrix operations, e.g., $\mathbf{A}\mathbf{A}^T$, $\left(\mathbf{I} + \frac{1}{c}\mathbf{A}\mathbf{A}^T\right)^{-1}$ and $\mathbf{A}^T\left(\mathbf{I} + \frac{1}{c}\mathbf{A}\mathbf{A}^T\right)^{-1}\mathbf{A}$ (c is a given positive constant). More details can be found in [10, Sec. 6.4]. Furthermore, the CPU time of initialization stage of `ADMM` is increased dramatically when the dimension of $\mathbf{A}$ is increased from 2000×4000 to 5000×10000. iv) `SpaRSA` works better when the density of $\mathbf{x}_{\text{true}}$ is smaller, e.g., 0.1, than when it is large, e.g., 0.2 and 0.4. v) The asymptotic convergence speed of `GreedyBCD` is slow, because only one variable is updated in each iteration.

5. CONCLUDING REMARKS

In this paper, we have proposed a novel exact/successive line search method for nonsmooth optimization problems. The proposed approach is to perform line search over a differentiable upper bound function of the original nonsmooth function, and it outperforms state-of-the-art from the perspective of both computational complexity and signaling burden. When applied to the LASSO problem, the proposed simplified line search is easily implementable due to the existence of closed-form expression, converges faster than state-of-the-art algorithms and robust w.r.t. the choice of parameters.

REFERENCES

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

[2] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: Numerical methods*. Prentice Hall, 1989.

[3] S.-J. Kim and G. B. Giannakis, "Optimal resource allocation for MIMO Ad Hoc cognitive radio networks," *IEEE Transactions on Information Theory*, vol. 57, no. 5, pp. 3117–3131, May 2011.

[4] 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.

[5] M. Patriksson, "Cost approximation: A unified framework of descent algorithms for nonlinear programs," *SIAM Journal on Optimization*, vol. 8, no. 2, pp. 561–582, May 1998.

[6] M. Elad, "Why simple shrinkage is still relevant for redundant representations?" *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5559–5569, 2006.

[7] Y. Yang and M. Pesavento, "A novel iterative convex approximation method," Jun. 2015, submitted to *IEEE Transactions on Signal Processing*. [Online]. Available: http://arxiv.org/abs/1506.04972

[8] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 58, no. 1, pp. 267–288, Jun. 1996.

[9] 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.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends in Machine Learning*, 2010.

[11] S. Wright, R. Nowak, and M. Figueiredo, "Sparse Reconstruction by Separable Approximation," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2479–2493, Jul. 2009.

[12] Z. Peng, M. Yan, and W. Yin, "Parallel and distributed sparse optimization," *2013 Asilomar Conference on Signals, Systems and Computers*, Nov. 2013.

[13] Y. Yang, http://www.nts.tu-darmstadt.de/home_nts/staff_nts/mitarbeiterdetails_32448.en.jsp.