A Novel Iterative Convex Approximation Method

Yang Yang and Marius Pesavento

***Abstract*—In this paper, we propose a novel iterative convex approximation algorithm to efficiently compute stationary points for a large class of possibly nonconvex optimization problems. The stationary points are obtained by solving a sequence of successively refined approximate problems, each of which is much easier to solve than the original problem. To achieve convergence, the approximate problem only needs to exhibit a weak form of convexity, namely, pseudo-convexity. We show that the proposed framework not only includes as special cases a number of existing methods, for example, the gradient method and the Jacobi algorithm, but also leads to new algorithms which enjoy easier implementation and faster convergence speed. We also propose a novel line search method for nondifferentiable optimization problems, which is carried out over a properly constructed differentiable function with the benefit of a simplified implementation as compared to state-of-the-art line search techniques that directly operate on the original nondifferentiable objective function. The advantages of the proposed algorithm are shown, both theoretically and numerically, by several example applications, namely, MIMO broadcast channel capacity computation and LASSO in sparse signal recovery.**

***Index Terms*—Exact Line Search, LASSO, MIMO Broadcast Channel, Nonconvex Optimization, Nondifferentiable Optimization, Successive Convex Approximation.**

I. Introduction

In this paper, we propose an iterative algorithm to solve the following general optimization problem:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X}, \end{aligned} \tag{1}$$

where $\mathcal{X} \subseteq \mathcal{R}^n$ is a closed and convex set, and $f(\mathbf{x}) : \mathcal{R}^n \to \mathcal{R}$ is a proper and differentiable function with a continuous gradient. We assume that problem (1) has a solution.

Problem (1) also includes some class of nondifferentiable optimization problems, if the nondifferentiable function $g(\mathbf{x})$ is convex:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) + g(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X}, \end{aligned} \tag{2}$$

because problem (2) can be rewritten into a problem with the form of (1) by the help of auxiliary variables:

$$\begin{aligned} \underset{\mathbf{x},y}{\text{minimize}} \quad & f(\mathbf{x}) + y \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X},\, g(\mathbf{x}) \leq y. \end{aligned} \tag{3}$$

We do not assume that $f(\mathbf{x})$ is convex, so (1) is in general a nonconvex optimization problem. The focus of this paper is on the development of efficient iterative algorithms for computing the stationary points of problem (1). The optimization problem (1) represents general class of optimization problems with a vast number of diverse applications. Consider for example the sum-rate maximization in the MIMO multiple access channel (MAC) [1], the broadcast channel (BC) [2, 3] and the interference channel (IC) [4, 5, 6, 7, 8, 9], where $f(\mathbf{x})$ is the sum-rate function of multiple users (to be maximized) while the set $\mathcal{X}$ characterizes the users' power constraints. In the context of the MIMO IC, (1) is a nonconvex problem and NP-hard [5]. As another example, consider portfolio optimization in which $f(\mathbf{x})$ represents the expected return of the portfolio (to be maximized) and the set $\mathcal{X}$ characterizes the trading constraints [10]. Furthermore, in sparse (l_1-regularized) linear regression, $f(\mathbf{x})$ denotes the least square function and $g(\mathbf{x})$ is the sparsity regularization function [11, 12].

Commonly used iterative algorithms belong to the class of descent direction methods such as the conditional gradient method and the gradient projection method [13], which often suffer from slow convergence. To speed up the convergence, the block coordinate descent (BCD) method that uses the notion of the nonlinear best-response has been widely studied [13, Sec. 2.7]. In particular, this method is applicable if the constraint set of (1) has a Cartesian product structure $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_K$ such that

$$\begin{aligned} \underset{\mathbf{x}=(\mathbf{x}_k)_{k=1}^K}{\text{minimize}} \quad & f(\mathbf{x}_1, \ldots, \mathbf{x}_K) \\ \text{subject to} \quad & \mathbf{x}_k \in \mathcal{X}_k,\, k = 1, \ldots, K. \end{aligned} \tag{4}$$

The BCD method is an iterative algorithm: in each iteration, only one variable is updated by its best-response $\mathbf{x}_k^{t+1} = \arg\min_{\mathbf{x}_k \in \mathcal{X}_k} f(\mathbf{x}_1^{t+1}, \ldots, \mathbf{x}_{k-1}^{t+1}, \mathbf{x}_k, \mathbf{x}_{k+1}^{t}, \ldots, \mathbf{x}_K^{t})$ (i.e., the point that minimizes $f(\mathbf{x})$ with respect to (w.r.t.) the variable $\mathbf{x}_k$ only while the remaining variables are fixed to their values of the preceding iteration) and the variables are updated sequentially. This method and its variants have been successfully adopted to many practical problems [1, 6, 7, 10, 14].

When the number of variables is large, the convergence speed of the BCD method may be slow due to the sequential nature of the update. A parallel variable update based on the best-response seems attractive as a mean to speed up the updating procedure, however, the convergence of a parallel best-response algorithm is only guaranteed under rather restrictive conditions, c.f. the diagonal dominance condition on the objective function $f(\mathbf{x}_1, \ldots, \mathbf{x}_K)$ [15], which is not only difficult to satisfy but also hard to verify. If $f(\mathbf{x}_1, \ldots, \mathbf{x}_K)$ is convex, the parallel algorithms converge if the stepsize is inversely proportional to the number of block variables K. This choice of stepsize, however, tends to be overly conservative in systems with a large number of block variables and inevitably slows down the convergence [2, 10, 16].

A recent progress in parallel algorithms has been made in

The authors are with Communication Systems Group, Darmstadt University of Technology, Germany (Email: {yang, pesavento}@nt.tu-darmstadt.de). Their work is supported by the Seventh Framework Programme for Research of the European Commission under grant number ADEL-619647.

[8, 9, 17, 18], in which it was shown that the stationary point of (1) can be found by solving a sequence of successively refined *approximate problems* of the original problem (1), and convergence to a stationary point is established if, among other conditions, the approximate function (the objective function of the approximate problem) and stepsizes are properly selected. The parallel algorithms proposed in [8, 9, 17, 18] are essentially descent direction methods. A description on how to construct the approximate problem such that the convexity of the original problem is preserved as much as possible is also contained in [8, 9, 17, 18] to achieve faster convergence than standard descent directions methods such as classical conditional gradient method and gradient projection method.

Despite its novelty, the parallel algorithms proposed in [8, 9, 17, 18] suffer from two limitations. Firstly, the approximate function must be strongly convex, and this is usually guaranteed by artificially adding a quadratic regularization term to the original objective function $f(\mathbf{x})$, which however may destroy the desirable characteristic structure of the original problem that could otherwise be exploited, e.g., to obtain computationally efficient closed-form solutions of the approximate problems [6]. Secondly, the algorithms require the use of a decreasing stepsize. 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 for the stepsize that provides a good trade-off between convergence speed and convergence guarantee, and current practices mainly rely on heuristics [17].

The contribution of this paper consists of the development of a novel iterative convex approximation method to solve problem (1). In particular, the advantages of the proposed iterative algorithm are the following:

1) The approximate function of the original problem (1) in each iteration only needs to exhibit a weak form of convexity, namely, pseudo-convexity. The proposed iterative method not only includes as special cases many existing methods, for example, [4, 6, 8, 9, 14, 17], but also opens new possibilities for constructing approximate problems that are easier to solve. For example, in the MIMO BC sum-rate maximization problems (Sec. IV-A), the new approximate problems can be solved in closed-form. We also show by a counterexample that the assumption on pseudo-convexity is tight in the sense that if it is not satisfied, the algorithm may not converge.

2) The stepsizes can be determined based on the problem structure, typically resulting in faster convergence than in cases where constant stepsizes [2, 10, 16] and decreasing stepsizes [8, 17] are used. For example, a constant stepsize can be used when $f(\mathbf{x})$ is given as the difference of two convex functions as in DC programming [19]. When the objective function is nondifferentiable, we propose a new exact/successive line search method that is carried out over a properly constructed differentiable function. Thus it is much easier to implement than state-of-the-art techniques that operate on the original nondifferentiable objective function directly.

In the proposed algorithm, the exact/successive line search is used to determine the stepsize and it can be implemented in a centralized controller, whose existence presence is justified for particular applications, e.g., the base station in the MIMO BC, and the portfolio manager in multi-portfolio optimization [10]. We remark that also in applications in which centralized controller are not admitted, however, the line search procedure does not necessarily imply an increased signaling burden when it is implemented in a distributed manner among different distributed processors. For example, in the LASSO problem studied in Sec. IV-B and Sec. IV-C, the stepsize based on the exact line search can be computed in closed-form and it does not incur any additional signaling as in predetermined stepsizes, e.g., decreasing stepsizes and constant stepsizes. Besides, even in cases where the line search procedure induces additional signaling, the burden is often fully amortized by the significant increase in the convergence rate.

The rest of the paper is organized as follows. In Sec. II we introduce the mathematical background. The novel iterative method is proposed and its convergence is analyzed in Sec. III; its connection to several existing descent direction algorithms is presented there. In Sec. IV, several applications are considered: sum rate maximization problem of MIMO BC to illustrate the advantage of the proposed approximate function, and LASSO to illustrate the advantage of the proposed stepsize. The paper is finally concluded in Sec. V.

Notation: We use x, $\mathbf{x}$ and $\mathbf{X}$ to denote scalar, vector and matrix, respectively. We use X_{jk} to denote the (j,k)-th element of $\mathbf{X}$; x_k is the k-th element of $\mathbf{x}$, and $\mathbf{x}=(x_k)_{k=1}^K$. We denote $\mathbf{x}^{-1}$ as the element-wise inverse of $\mathbf{x}$, i.e., $(\mathbf{x}^{-1})_k = 1/x_k$. Notation $\mathbf{x}\circ\mathbf{y}$ and $\mathbf{X}\otimes\mathbf{Y}$ denotes the Hadamard product between $\mathbf{x}$ and $\mathbf{y}$, and the Kronecker product between $\mathbf{X}$ and $\mathbf{Y}$, respectively. The operator $[\mathbf{x}]_{\mathbf{a}}^{\mathbf{b}}$ returns the element-wise projection of $\mathbf{x}$ onto $[\mathbf{a},\mathbf{b}]$: $[\mathbf{x}]_{\mathbf{a}}^{\mathbf{b}} \triangleq \max(\min(\mathbf{x},\mathbf{b}),\mathbf{a})$, and $[\mathbf{x}]^+ \triangleq [\mathbf{x}]_{\mathbf{0}}$. We denote $\lceil x \rceil$ as the smallest integer that is larger than or equal to x. We denote $\mathbf{d}(\mathbf{X})$ as the vector that consists of the diagonal elements of $\mathbf{X}$ and $\mathrm{diag}(\mathbf{x})$ is a diagonal matrix whose diagonal elements are as same as $\mathbf{x}$. We use $\mathbf{1}$ to denote the vector whose elements are equal to 1.

II. Preliminaries on Descent Direction Method and Convex Functions

In this section, we introduce the basic definitions and concepts that are fundamental in the development of the mathematical formalism used in the rest of the paper.

Stationary point. A point $\mathbf{y}$ is a stationary point of (1) if

$$(\mathbf{x}-\mathbf{y})^T\nabla f(\mathbf{y}) \geq 0,\ \forall \mathbf{x}\in\mathcal{X}. \tag{5}$$

Condition (5) is the necessary condition for local (and also global) optimality of the variable $\mathbf{y}$. For nonconvex problems, where global optimality conditions are difficult to establish, the computation of stationary points of the optimization problem (1) is generally desired. If (1) is convex, stationary points coincide with (globally) optimal points and condition (5) is also sufficient for $\mathbf{y}$ to be (globally) optimal.

Descent direction. The vector $\mathbf{d}^t$ is a descent direction of the function $f(\mathbf{x})$ at $\mathbf{x}=\mathbf{x}^t$ if

$$\nabla f(\mathbf{x}^t)^T\mathbf{d}^t < 0. \tag{6}$$

If (6) is satisfied, the function $f(\mathbf{x})$ can be decreased when $\mathbf{x}$ is updated from $\mathbf{x}^t$ along direction $\mathbf{d}^t$. This is because in the Taylor expansion of $f(\mathbf{x})$ around $\mathbf{x} = \mathbf{x}^t$ is given by:

$$f(\mathbf{x}^t + \gamma \mathbf{d}^t) = f(\mathbf{x}^t) + \gamma \nabla f(\mathbf{x}^t)^T \mathbf{d}^t + o(\gamma),$$

where the first order term is negative in view of (6). For sufficiently small γ, the first order term dominates all higher order terms. More rigorously, if $\mathbf{d}^t$ is a descent direction, there exists a $\bar{\gamma}^t > 0$ such that [20, 8.2.1]

$$f(\mathbf{x}^t + \gamma \mathbf{d}^t) < f(\mathbf{x}^t), \forall \gamma \in (0, \bar{\gamma}^t).$$

Note that the converse is not necessarily true, i.e., $f(\mathbf{x}^{t+1}) < f(\mathbf{x}^t)$ does not necessarily imply that $\mathbf{x}^{t+1} - \mathbf{x}^t$ is a descent direction of $f(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}^t$.

Quasi-convex function. A function $h(\mathbf{x})$ is quasi-convex if for any $\alpha \in [0, 1]$:

$$h((1-\alpha)\mathbf{x} + \alpha \mathbf{y}) \leq \max(h(\mathbf{x}), h(\mathbf{y})), \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}.$$

A locally optimal point $\mathbf{y}$ of a quasi-convex function $h(\mathbf{x})$ over a convex set $\mathcal{X}$ is also globally optimal, i.e.,

$$h(\mathbf{x}) \geq h(\mathbf{y}), \forall \mathbf{x} \in \mathcal{X}.$$

Pseudo-convex function. A function $h(\mathbf{x})$ is pseudo-convex if [21]

$$\nabla h(\mathbf{x})^T (\mathbf{y} - \mathbf{x}) \geq 0 \Longrightarrow h(\mathbf{y}) \geq h(\mathbf{x}), \ \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}.$$

Another equivalent definition of pseudo-convex functions is also useful in our context [21]:

$$h(\mathbf{y}) < h(\mathbf{x}) \Longrightarrow \nabla h(\mathbf{x})^T (\mathbf{y} - \mathbf{x}) < 0. \tag{7}$$

In other words, $h(\mathbf{y}) < h(\mathbf{x})$ implies that $\mathbf{y} - \mathbf{x}$ is a descent direction of $h(\mathbf{x})$. A pseudo-convex function is also quasi-convex [21, Th. 9.3.5], and thus any locally optimal points of pseudo-convex functions are also globally optimal.

Convex function. A function $h(\mathbf{x})$ is convex if

$$h(\mathbf{y}) \geq h(\mathbf{x}) + \nabla h(\mathbf{x})^T (\mathbf{y} - \mathbf{x}), \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}.$$

It is strictly convex if the above inequality is satisfied with strict inequality whenever $\mathbf{x} \neq \mathbf{y}$. It is easy to see that a convex function is pseudo-convex.

Strongly convex functions. A function $h(\mathbf{x})$ is strongly convex with constant a if

$$h(\mathbf{y}) \geq h(\mathbf{x}) + \nabla h(\mathbf{x})^T (\mathbf{y} - \mathbf{x}) + \tfrac{a}{2} \|\mathbf{x} - \mathbf{y}\|_2^2, \forall \mathbf{x}, \mathbf{y} \in \mathcal{X},$$

for some positive constant a. The relationship of functions with different degree of convexity is summarized in Fig. 1.

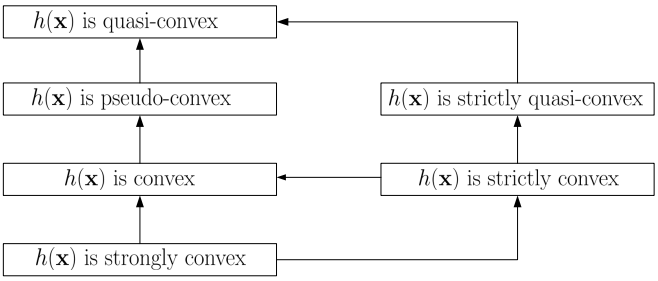

Figure 1. Relationship of functions with different degree of convexity

III. The Proposed Iterative Convex Approximation Method

In this section, we propose an iterative algorithm that solves (1) as a sequence of successively refined approximate problems, each of which is much easier to solve than the original problem (1), e.g., the approximate problem can be decomposed into independent subproblems that even exhibits a closed-form solution.

In iteration t, let $\tilde{f}(\mathbf{x}; \mathbf{x}^t)$ be the approximate function of $f(\mathbf{x})$ around the point $\mathbf{x}^t$. Then the approximate problem is

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \tilde{f}(\mathbf{x}; \mathbf{x}^t) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X}, \end{aligned} \tag{8}$$

and its optimal point and solution set is denoted as $\mathbb{B}\mathbf{x}^t$ and $\mathcal{S}(\mathbf{x}^t)$, respectively:

$$\mathbb{B}\mathbf{x}^t \in \mathcal{S}(\mathbf{x}^t) \triangleq \left\{ \mathbf{x}^\star \in \mathcal{X} : \ \tilde{f}(\mathbf{x}^\star; \mathbf{x}^t) = \min_{\mathbf{x} \in \mathcal{X}} \tilde{f}(\mathbf{x}; \mathbf{x}^t) \right\}. \tag{9}$$

We assume that the approximate function $\tilde{f}(\mathbf{x}; \mathbf{y})$ satisfies the following technical conditions:

(A1) The approximate function $\tilde{f}(\mathbf{x}; \mathbf{y})$ is pseudo-convex in $\mathbf{x}$ for any given $\mathbf{y} \in \mathcal{X}$;

(A2) The approximate function $\tilde{f}(\mathbf{x}; \mathbf{y})$ is continuously differentiable in $\mathbf{x}$ for any given $\mathbf{y} \in \mathcal{X}$ and continuous in $\mathbf{y}$ for any $\mathbf{x} \in \mathcal{X}$;

(A3) The gradient of $\tilde{f}(\mathbf{x}; \mathbf{y})$ and the gradient of $f(\mathbf{x})$ are identical at $\mathbf{x} = \mathbf{y}$ for any $\mathbf{y} \in \mathcal{X}$, i.e., $\nabla_{\mathbf{x}} \tilde{f}(\mathbf{y}; \mathbf{y}) = \nabla_{\mathbf{x}} f(\mathbf{y})$;

Based on (9), we define the mapping $\mathbb{B}\mathbf{x}$ that is used to generate the sequence of points in the proposed algorithm:

$$\mathcal{X} \ni \mathbf{x} \longmapsto \mathbb{B}\mathbf{x} \in \mathcal{X}. \tag{10}$$

Given the mapping $\mathbb{B}\mathbf{x}$, the following properties hold.

Proposition 1 (Stationary point and descent direction)**.** *(i) A point $\mathbf{y}$ is a stationary point of (1) if and only if $\mathbf{y} \in \mathcal{S}(\mathbf{y})$ defined in (9); (ii) If $\mathbf{y}$ is not a stationary point of (1), then $\mathbb{B}\mathbf{y} - \mathbf{y}$ is a descent direction of $f(\mathbf{x})$:*

$$\nabla f(\mathbf{y})^T (\mathbb{B}\mathbf{y} - \mathbf{y}) < 0. \tag{11}$$

Proof: See Appendix A. ■

If $\mathbf{x}^t$ is not a stationary point, according to Proposition 1, we define the vector update $\mathbf{x}^{t+1}$ in the $(t+1)$-th iteration as:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \gamma^t (\mathbb{B}\mathbf{x}^t - \mathbf{x}^t), \tag{12}$$

where $\gamma^t \in (0, 1]$ is an appropriate stepsize that can be determined by either the exact line search (also known as the minimization rule) or the successive line search (also known as the Armijo rule). Since $\mathbf{x}^t \in \mathcal{X}$, $\mathbb{B}\mathbf{x}^t \in \mathcal{X}$ and $\gamma^t \in (0, 1]$, it follows from the convexity of $\mathcal{X}$ that $\mathbf{x}^{t+1} \in \mathcal{X}$ for all t.

Exact line search. The stepsize is selected such that the function $f(\mathbf{x})$ is decreased to the largest extent along the descent direction $\mathbb{B}\mathbf{x}^t - \mathbf{x}^t$:

$$\gamma^t \in \underset{0 \leq \gamma \leq 1}{\arg\min} \ f(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)). \tag{13}$$

Algorithm 1 The iterative convex approximation algorithm for differentiable problem (1)

Data: $t = 0$ and $\mathbf{x}^0 \in \mathcal{X}$.
Repeat the following steps until convergence:

S1: Compute $\mathbb{B}\mathbf{x}^t$ using (9).
S2: Compute γ^t by the exact line search (13) or the successive line search (14).
S3: Update $\mathbf{x}^{t+1}$ according to (12) and set $t \leftarrow t+1$.

With this stepsize rule, it is easy to see that if $\mathbf{x}^t$ is not a stationary point, then $f(\mathbf{x}^{t+1}) < f(\mathbf{x}^t)$.

In the special case that $f(\mathbf{x})$ in (1) is convex and $\gamma^\star$ nulls the gradient of $f(\mathbf{x}^t+\gamma(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))$, i.e., $\nabla_\gamma f(\mathbf{x}^t+\gamma^\star(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))=0$, then γ^t in (13) is simply the projection of $\gamma^\star$ onto the interval $[0,1]$:

$$\gamma^t = [\gamma^\star]_0^1 = \begin{cases} 1, & \text{if } \nabla_\gamma f(\mathbf{x}^t+\gamma(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))|_{\gamma=1} \geq 0, \\ 0, & \text{if } \nabla_\gamma f(\mathbf{x}^t+\gamma(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))|_{\gamma=0} \leq 0, \\ \gamma^\star, & \text{otherwise.} \end{cases}$$

If $0 \leq \gamma^t = \gamma^\star \leq 1$, the constrained optimization problem in (13) is essentially unconstrained. In some applications it is possible to compute $\gamma^\star$ analytically, e.g., if $f(\mathbf{x})$ is quadratic as in the LASSO problem (Sec. IV-B and Sec. IV-C). Otherwise, for general convex functions, $\gamma^\star$ can be found efficiently by the bisection method as follows. Restricting the function $f(\mathbf{x})$ to a line $\mathbf{x}^t+\gamma(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)$, the new function $f(\mathbf{x}^t+\gamma(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))$ is convex in γ [22]. It thus follows that $\nabla_\gamma f(\mathbf{x}^t+\gamma(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)) < 0$ if $\gamma < \gamma^\star$ and $\nabla_\gamma f(\mathbf{x}^t+\gamma(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)) > 0$ if $\gamma > \gamma^\star$. 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), set $\gamma_{\text{mid}} = (\gamma_{\text{low}}+\gamma_{\text{up}})/2$ and refine γ_{low} and γ_{up} according to the following rule:

$$\begin{cases} \gamma_{\text{low}} = \gamma_{\text{mid}}, & \text{if } \nabla_\gamma f(\mathbf{x}^t+\gamma_{\text{mid}}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)) > 0, \\ \gamma_{\text{up}} = \gamma_{\text{mid}}, & \text{if } \nabla_\gamma f(\mathbf{x}^t+\gamma_{\text{mid}}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)) < 0. \end{cases}$$

The procedure is repeated for finite times until the gap $\gamma_{\text{up}} - \gamma_{\text{low}}$ is smaller than a prescribed precision.

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 (13), the successive line search can instead be employed: given scalars $0<\alpha<1$ and $0<\beta<1$, 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:

$$f(\mathbf{x}^t+\beta^m(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)) \leq f(\mathbf{x}^t)+\alpha\beta^m\nabla f(\mathbf{x}^t)^T(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t). \tag{14}$$

Note that the existence of a finite m_t satisfying (14) is always guaranteed if $\nabla f(\mathbf{x}^t)^T(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t) < 0$ [13], i.e., from Proposition 1 inequality (14) always admits a solution.

The algorithm is formally summarized in Algorithm 1 and its convergence properties are given in the following theorem.

Theorem 2 (Convergence to a stationary point)**.** *Consider the sequence $\{\mathbf{x}^t\}$ generated by Algorithm 1. Provided that Assumptions (A1)-(A3) as well as the following assumptions are satisfied:*

(A4) The solution set $\mathcal{S}(\mathbf{x}^t)$ is nonempty for $t=1,2,\ldots$;

(A5) Given any convergent subsequence $\{\mathbf{x}^t\}_{t\in\mathcal{T}}$ where $\mathcal{T} \subseteq \{1,2,\ldots\}$, the sequence $\{\mathbb{B}\mathbf{x}^t\}_{t\in\mathcal{T}}$ is bounded.

Then any limit point of $\{\mathbf{x}^t\}$ is a stationary point of (1).

Proof: See Appendix B. ■

In the following we discuss some properties of the proposed Algorithm 1.

On the conditions (A1)-(A5). The only requirement on the convexity of the approximate function $\tilde{f}(\mathbf{x};\mathbf{x}^t)$ is that it is pseudo-convex, cf. (A1). To the best of our knowledge, these are the weakest conditions for descent direction methods available in the literature. As a result, it enables the construction of new approximate functions that can often be optimized more easily or even in closed-form, resulting in a significant reduction of the computational cost if the approximate problems must otherwise only be optimized by iterative algorithms as in standard solvers [23]. Assumptions (A2)-(A3) represent standard conditions for successive convex approximation techniques and are satisfied for many existing approximation functions, cf. Sec. III-B. Sufficient conditions for Assumptions (A4)-(A5) are that either the feasible set $\mathcal{X}$ in (8) is bounded or the approximate function in (8) is strongly convex [24]. We show that how these assumptions are satisfied in popular applications considered in Sec. IV.

On the pseudo-convexity of the approximate function. Assumption (A1) is tight in the sense that if it is not satisfied, Proposition 1 may not hold. Consider the following simple example: $f(x)=x^3$, where $-1 \leq x \leq 1$ and the point $x^t=0$ at iteration t. Choosing the approximate function $\tilde{f}(x;x^t)=x^3$, which is quasi-convex but not pseudo-convex, all assumptions except (A1) are satisfied. It is easy to see that $\mathbb{B}x^t=-1$, however $(\mathbb{B}x^t-x^t)\nabla f(x^t)=(-1-0)\cdot 0=0$, and thus $\mathbb{B}x^t-x^t$ is not a descent direction, i.e., inequality (11) in Proposition 1 is violated.

On the stepsize. The stepsize can be determined in a more straightforward way if $\tilde{f}(\mathbf{x};\mathbf{x}^t)$ is a global upper bound of $f(\mathbf{x})$ that is exact at $\mathbf{x}=\mathbf{x}^t$, i.e., assume that

(A6) $\tilde{f}(\mathbf{x};\mathbf{x}^t) \geq f(\mathbf{x})$ and $\tilde{f}(\mathbf{x}^t;\mathbf{x}^t)=f(\mathbf{x}^t)$,

then Algorithm 1 converges under the choice $\gamma^t=1$ which results in the update $\mathbf{x}^{t+1}=\mathbb{B}\mathbf{x}^t$. To see this, we first remark that $\gamma^t=1$ must be an optimal point of the following problem:

$$1 \in \operatorname*{argmin}_{0\leq\gamma\leq 1} \tilde{f}(\mathbf{x}^t+\gamma(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t);\mathbf{x}^t), \tag{15}$$

otherwise the optimality of $\mathbb{B}\mathbf{x}^t$ is contradicted, cf. (9). At the same time, it follows from Proposition 1 that $\nabla\tilde{f}(\mathbf{x}^t;\mathbf{x}^t)^T(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)<0$. The successive line search over $\tilde{f}(\mathbf{x}^t+\gamma(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))$ thus yields a nonnegative and finite integer m_t such that for some $0<\alpha<1$ and $0<\beta<1$:

$$\begin{aligned} \tilde{f}(\mathbb{B}\mathbf{x}^t;\mathbf{x}^t) &\leq \tilde{f}(\mathbf{x}^t+\beta^{m_t}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t);\mathbf{x}^t) \\ &\leq \tilde{f}(\mathbf{x}^t)+\alpha\beta^{m_t}\nabla\tilde{f}(\mathbf{x}^t;\mathbf{x}^t)^T(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t) \\ &= f(\mathbf{x}^t)+\alpha\beta^{m_t}\nabla f(\mathbf{x}^t)^T(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t), \end{aligned} \tag{16}$$

where the last equality follows from Assumptions (A3) and (A6). Invoking Assumption (A6) again, we obtain

$$f(\mathbf{x}^{t+1}) \leq f(\mathbf{x}^t)+\alpha\beta^{m_t}\nabla f(\mathbf{x}^t)^T(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)\big|_{\mathbf{x}^{t+1}=\mathbb{B}\mathbf{x}^t}. \tag{17}$$

The proof of Theorem 2 can be used verbatim to prove the convergence of Algorithm 1 with a constant stepsize $\gamma^t = 1$.

A. Nondifferentiable optimization problems

In the following we show that the proposed Algorithm 1 can also be applied to solve problem (3), and its equivalent formulation (2) which contains a nondifferentiable objective function. Suppose that $\tilde{f}(\mathbf{x};\mathbf{x}^t)$ is an approximate function of $f(\mathbf{x})$ in (3) around $\mathbf{x}^t$ and it satisfies Assumptions (A1)-(A3). Then the approximation of problem (3) around $\mathbf{x}^t$ is

$$\begin{aligned} \underset{\mathbf{x},y}{\text{minimize}} \quad & \tilde{f}(\mathbf{x};\mathbf{x}^t) + y \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X},\, g(\mathbf{x}) \le y. \end{aligned} \tag{18}$$

To see this, it is sufficient to verify Assumption (A3) only:

$$\begin{aligned} \nabla_{\mathbf{x}}(\tilde{f}(\mathbf{x}^t;\mathbf{x}^t) + y^t) &= \nabla_{\mathbf{x}}(f(\mathbf{x}^t) + y^t), \\ \nabla_{y}(\tilde{f}(\mathbf{x}^t;\mathbf{x}^t) + y^t) &= \nabla_{y}(f(\mathbf{x}^t) + y^t) = 1. \end{aligned}$$

We denote its optimal points $\mathbf{x}$ and y of (18) as $\mathbb{B}\mathbf{x}^t$ and $y^\star(\mathbf{x}^t)$, respectively. Then it is easy to infer from (18) that $y^\star(\mathbf{x}^t) = g(\mathbb{B}\mathbf{x}^t)$. Based on the exact line search, the stepsize γ^t in this case is given as

$$\gamma^t \in \underset{0\le\gamma\le 1}{\operatorname{argmin}} \big\{ f(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) + y^t + \gamma(y^\star(\mathbf{x}^t) - y^t)) \big\}. \tag{19}$$

Then the variables $\mathbf{x}^{t+1}$ and y^{t+1} are defined as follows:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \gamma^t(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t), \tag{20a}$$

$$y^{t+1} = y^t + \gamma^t(y^\star(\mathbf{x}^t) - y^t). \tag{20b}$$

The convergence of Algorithm 1 with $(\mathbb{B}\mathbf{x}^t, y^\star(\mathbf{x}^t))$ and γ^t given by (18)-(19) directly follows from Theorem 2.

The point y^{t+1} given in (20b) can be further refined. Note that $\mathbb{B}\mathbf{x}^t$ is independent of y^t and

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

where the first and the second inequality comes from the fact that $y^t \ge g(\mathbf{x}^t)$ and Jensen's inequality of convex functions $g(\mathbf{x})$ [22], respectively. Since $y^{t+1} \ge g(\mathbf{x}^{t+1})$ by definition, the point $(\mathbf{x}^{t+1}, g(\mathbf{x}^{t+1}))$ always yields a lower value of $f(\mathbf{x}) + y$ than $(\mathbf{x}^{t+1}, y^{t+1})$ while $(\mathbf{x}^{t+1}, g(\mathbf{x}^{t+1}))$ is still a feasible point for problem (3). The update (20b) is then replaced by the following enhanced rule:

$$y^{t+1} = g(\mathbf{x}^{t+1}). \tag{21}$$

Algorithm 1 with $\mathbb{B}\mathbf{x}^t$ given in (20a) and y^{t+1} given in (21) still converges to a stationary point of (3).

Based on (21), the notations in (18)-(19) can be simplified by removing the auxiliary variable $\mathbf{y}$: (18) and (19) is respectively equivalent to

$$\mathbb{B}\mathbf{x}^t = \underset{\mathbf{x}\in\mathcal{X}}{\arg\min} \big\{ \tilde{f}(\mathbf{x};\mathbf{x}^t) + g(\mathbf{x}) \big\} \tag{22}$$

and

$$\gamma^t \in \underset{0\le\gamma\le 1}{\operatorname{argmin}} \big\{ f(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) + \gamma(g(\mathbb{B}\mathbf{x}^t) - g(\mathbf{x}^t)) \big\}. \tag{23}$$

Algorithm 2 The iterative convex approximation algorithm for nondifferentiable problem (2)

Data: $t = 0$ and $\mathbf{x}^0 \in \mathcal{X}$.

Repeat the following steps until convergence:

S1: Compute $\mathbb{B}\mathbf{x}^t$ using (22).

S2: Compute γ^t by the exact line search (23) or the successive line search (24).

S3: Update $\mathbf{x}^{t+1}$ according to

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

Set $t \leftarrow t + 1$.

In the context of the successive line search, customizing the general definition (14) for problem (2) yields the choice $\gamma^t = \beta^{m_t}$ with m_t being the smallest integer that satisfies the inequality:

$$\begin{aligned} &f(\mathbf{x}^t + \beta^m(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) - f(\mathbf{x}^t) \le \\ &\quad \beta^m\big(\alpha\nabla f(\mathbf{x}^t)^T(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t) + (\alpha-1)(g(\mathbb{B}\mathbf{x}^t) - g(\mathbf{x}^t))\big). \end{aligned} \tag{24}$$

Based on the derivations above, the proposed algorithm for the nondifferentiable problem (2) is formally summarized in Algorithm 2.

It is much easier to calculate γ^t according to (23) than in state-of-the-art techniques that directly carry out the exact line search over the original nondifferentiable objective function in (2) [25, Rule E], i.e., $\min_{0\le\gamma\le 1} f(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) + g(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t))$. This is because the objective function in (23) is differentiable in γ while state-of-the-art techniques involve the minimization of a nondifferentiable function. If $f(\mathbf{x})$ exhibits a specific structure such as in quadratic functions, γ^t can even be calculated in closed-form. This property will be exploited to develop fast and easily implementable algorithm for the popular LASSO problem in Sec. IV-B and Sec. IV-C.

In the proposed successive line search, the left hand side of (24) depends on $f(\mathbf{x})$ while the right hand side is linear in β^m. The proposed variation of the successive line search thus involves only the evaluation of the differentiable function $f(\mathbf{x})$ and it outperforms, from the perspective of both computational complexity and signaling exchange when implemented in a distributed manner, state-of-the-art techniques (for example [25, Rule A'], [17, Remark 4] and [26, Algorithm 2.1]) in which the whole nondifferentiable function $f(\mathbf{x}) + g(\mathbf{x})$ must be repeatedly evaluated (for different m) and compared with a certain benchmark before m_t is found.

B. Special Cases of the Proposed Method

In this subsection, we interpret some existing methods in the context of Algorithm 1 and show that they can be considered as special cases of the proposed algorithm.

Conditional gradient method: In this method, the approximate function is given as the first-order approximation of $f(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}^t$ [13, Sec. 2.2.2], i.e.,

$$\tilde{f}(\mathbf{x};\mathbf{x}^t) = \nabla f(\mathbf{x}^t)^T(\mathbf{x} - \mathbf{x}^t). \tag{25}$$

Then the stepsize is selected by either the exact line search or the successive line search.

Gradient projection method: In this method, $\mathbb{B}\mathbf{x}^t$ is given by [13, Sec. 2.3]

$$\mathbb{B}\mathbf{x}^t = \left[\mathbf{x}^t - s^t \nabla f(\mathbf{x}^t)\right]_{\mathcal{X}},$$

where $s^t > 0$ and $[\mathbf{x}]_{\mathcal{X}}$ denotes the projection of $\mathbf{x}$ onto $\mathcal{X}$. This is equivalent to defining $\tilde{f}(\mathbf{x};\mathbf{x}^t)$ in (9) as follows:

$$\tilde{f}(\mathbf{x};\mathbf{x}^t) = \nabla f(\mathbf{x}^t)^T(\mathbf{x}-\mathbf{x}^t) + \tfrac{1}{2s^k}\left\|\mathbf{x}-\mathbf{x}^t\right\|_2^2,$$

which is the first-order approximation of $f(\mathbf{x})$ augmented by a quadratic regularization term that is introduced to improve the numerical stability [15]. Then the stepsize is selected by either the exact or the successive line search.

Jacobi algorithm: If $f(\mathbf{x})$ is convex in each $\mathbf{x}_k$ where $k = 1,\ldots,K$ (but not necessarily jointly convex in $(\mathbf{x}_1,\ldots,\mathbf{x}_K)$), the approximate function is defined as [8]

$$\tilde{f}(\mathbf{x};\mathbf{x}^t) = \textstyle\sum_{k=1}^K \big(f(\mathbf{x}_k,\mathbf{x}_{-k}^t) + \frac{\tau_k}{2}\left\|\mathbf{x}_k-\mathbf{x}_k^t\right\|_2^2\big), \quad (26)$$

where $\tau_k \geq 0$ for $k = 1,\ldots,K$. The k-th component function $f(\mathbf{x}_k,\mathbf{x}_{-k}^t) + \frac{\tau_k}{2}\|\mathbf{x}_k-\mathbf{x}_k^t\|_2^2$ in (26) is obtained from the original function $f(\mathbf{x})$ by fixing all variables except $\mathbf{x}_k$, i.e., $\mathbf{x}_{-k} = \mathbf{x}_{-k}^t$, and further adding an (optional) quadratic regularization term. Since $\tilde{f}(\mathbf{x};\mathbf{x}^t)$ in (26) is convex, Assumption (A1) is satisfied. Based on the observations that

$$\begin{aligned}\nabla_{\mathbf{x}_k}\tilde{f}(\mathbf{x}^t;\mathbf{x}^t) &= \nabla_{\mathbf{x}_k}\big(f(\mathbf{x}_k,\mathbf{x}_{-k}^t) + \tfrac{\tau_k}{2}\left\|\mathbf{x}_k-\mathbf{x}_k^t\right\|_2^2\big)\big|_{\mathbf{x}_k=\mathbf{x}_k^t}\\ &= \nabla_{\mathbf{x}_k} f(\mathbf{x}_k,\mathbf{x}_{-k}^t) + \tau_k(\mathbf{x}_k-\mathbf{x}_k^t)\big|_{\mathbf{x}_k=\mathbf{x}_k^t}\\ &= \nabla_{\mathbf{x}_k} f(\mathbf{x}^t),\end{aligned}$$

we conclude that Assumption (A3) is satisfied by the choice of the approximate function in (26). The resulting approximate problem is given by

$$\begin{array}{ll}\underset{\mathbf{x}=(\mathbf{x}_k)_{k=1}^K}{\text{minimize}} & \sum_{k=1}^K f(\mathbf{x}_k,\mathbf{x}_{-k}^t)\\ \text{subject to} & \mathbf{x}\in\mathcal{X},\end{array} \quad (27)$$

while the stepsizes are then determined by line search.

To guarantee the convergence, the condition proposed in [9] is that $\tau_k > 0$ for all k in (26). However, this may destroy the convenient structure that could otherwise be exploited. In contrast to this, in the case $\tau_k = 0$, problem (27) may exhibit a closed-form solution. In the proposed method, the convergence is guaranteed even when $\tau_k = 0$ in (26) since $\tilde{f}(\mathbf{x};\mathbf{x}^t)$ in (26) is convex when $\tau_k = 0$ for all k and it naturally satisfies the pseudo-convexity assumption specified by Assumption (A1). We will show by an example application in the MIMO BC in Sec. IV-A that the proposed relaxation in approximate function yields new approximate problems that are much easier to solve.

The structure inside the constraint set $\mathcal{X}$, if any, may be exploited to solve (27) even more efficiently. For example, the constraint set $\mathcal{X}$ consists of separable constraints in the form of $\sum_{k=1}^K h_k(\mathbf{x}_k) \leq 0$ for some convex functions $h_k(\mathbf{x}_k)$. Since subproblem (27) is convex, primal and dual decomposition techniques can readily be used to solve (27) efficiently [27] (such an example is studied in Sec. IV-A).

Algorithm 3 The Jacobi algorithm for problem (4)

Data: $t = 0$ and $\mathbf{x}_k^0 \in \mathcal{X}_k$ for all $k = 1,\ldots,K$.
Repeat the following steps until convergence:

S1: For $k = 1,\ldots,K$, compute $\mathbb{B}_k\mathbf{x}^t$ using (28).

S2: Compute γ^t by the exact line search (13) or the successive line search (14).

S3: Update $\mathbf{x}^{t+1}$ according to

$$\mathbf{x}_k^{t+1} = \mathbf{x}_k^t + \gamma^t(\mathbb{B}_k\mathbf{x}^t - \mathbf{x}_k^t), \forall k = 1,\ldots,K.$$

Set $t \leftarrow t+1$.

In the case that the constraint set $\mathcal{X}$ has a Cartesian product structure (4), the subproblem (27) is naturally decomposed into K sub-problems, one for each variable, which are then solved *in parallel*. This is commonly known as the Jacobi update: $\mathbb{B}\mathbf{x}^t = (\mathbb{B}_k\mathbf{x}^t)_{k=1}^K$ and

$$\mathbb{B}_k\mathbf{x}^t \in \underset{\mathbf{x}_k\in\mathcal{X}_k}{\arg\min}\ f(\mathbf{x}_k,\mathbf{x}_{-k}^t),\ k = 1,\ldots,K, \quad (28)$$

where $\mathbb{B}_k\mathbf{x}^t$ can be interpreted as variable $\mathbf{x}_k$'s best-response to other variables $\mathbf{x}_{-k} = (\mathbf{x}_j)_{j\neq k}$ when $\mathbf{x}_{-k} = \mathbf{x}_{-k}^t$. The Jacobi algorithm is formally summarized in Algorithm 3.

If $f(\mathbf{x})$ is only pseudo-convex (but not necessarily convex) in each $\mathbf{x}_k$, the approximate function $\sum_{k=1}^K f(\mathbf{x}_k,\mathbf{x}_{-k}^t)$ is not necessarily pseudo-convex in $\mathbf{x}$, but Algorithm 3 stills converges as we show in the following theorem.

Theorem 3. *Consider the sequence $\{\mathbf{x}^t\}$ generated by Algorithm 3. Provided that $f(\mathbf{x})$ is pseudo-convex in $\mathbf{x}_k$ for all $k = 1,\ldots,K$ and Assumptions (A4)-(A5) are satisfied. Then any limit point of the sequence generated by Algorithm 3 is a stationary point of (4).*

Proof: See Appendix C. ∎

The convergence condition specified in Theorem 3 relaxes those in [8, 17]: $f(\mathbf{x})$ only needs to be pseudo-convex in each $\mathbf{x}_k$ and no regularization term is needed (i.e., $\tau_k = 0$). To the best of our knowledge, this is the weakest convergence condition on Jacobi algorithms available in the literature.

DC algorithm: If the objective function in (1) is the difference of two convex functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$:

$$f(\mathbf{x}) = f_1(\mathbf{x}) - f_2(\mathbf{x}),$$

the following approximate function can be used:

$$\tilde{f}(\mathbf{x};\mathbf{x}^t) = f_1(\mathbf{x}) - (f_2(\mathbf{x}^t) + \nabla f_2(\mathbf{x}^t)^T(\mathbf{x}-\mathbf{x}^t)).$$

Since $f_2(\mathbf{x})$ is convex and $f_2(\mathbf{x}) \geq f_2(\mathbf{x}^t) + \nabla f_2(\mathbf{x}^t)^T(\mathbf{x}-\mathbf{x}^t)$, Assumption (A6) is satisfied and the a constant unit stepsize can be chosen. Such a choice outperforms the algorithm proposed in [8] which uses decreasing stepsizes instead, i.e., it avoids the difficulty of finding a good decreasing rate and generally yields faster convergence.

IV. Example Applications

In this section, we apply the proposed Algorithms 1-2 to solve important practical problems that are of broad and fundamental interest to illustrate their advantages.

A. MIMO Broadcast Channel Capacity Computation

In this subsection, we study the MIMO BC capacity computation problem to illustrate the advantage of the proposed approximate function.

Consider a MIMO BC where the channel matrix characterizing the transmission from the base station to user k is denoted by $\mathbf{H}_k$, the transmit covariance matrix of the signal from the base station to user k is denoted as $\mathbf{Q}_k$, and the noise at each user k is an additive independent and identically distributed Gaussian vector with unit variance on each of its elements. Then the sum capacity of the MIMO BC is [28]

$$\begin{aligned} \underset{\{\mathbf{Q}_k\}}{\text{maximize}} \quad & \log\left|\mathbf{I}+\sum_{k=1}^K \mathbf{H}_k\mathbf{Q}_k\mathbf{H}_k^H\right| \\ \text{subject to} \quad & \mathbf{Q}_k \succeq \mathbf{0},\ k=1,\ldots,K, \sum_{k=1}^K \text{tr}(\mathbf{Q}_k) \le P, \end{aligned} \quad (29)$$

where P is the power budget at the base station.

Problem (29) is a convex problem whose solution cannot be expressed in closed-form and can only be found iteratively. To apply Algorithm 1, we invoke (26)-(27) and the approximate problem at the t-th iteration is

$$\begin{aligned} \underset{\{\mathbf{Q}_k\}}{\text{maximize}} \quad & \sum_{k=1}^K \log\left|\mathbf{R}_k(\mathbf{Q}^t_{-k})+\mathbf{H}_k\mathbf{Q}_k\mathbf{H}_k^H\right| \\ \text{subject to} \quad & \mathbf{Q}_k \succeq \mathbf{0},\ k=1,\ldots,K,\ \sum_{k=1}^K \text{tr}(\mathbf{Q}_k) \le P, \end{aligned} \quad (30)$$

where $\mathbf{R}_k(\mathbf{Q}^t_{-k}) \triangleq \mathbf{I}+\sum_{j\neq k}\mathbf{H}_j\mathbf{Q}_j^t\mathbf{H}_j^H$. The approximate function is concave in $\mathbf{Q}$ and differentiable in both $\mathbf{Q}$ and $\mathbf{Q}^t$, and thus Assumptions (A1)-(A3) are satisfied. Since the constraint set in (30) is compact, the approximate problem (30) has a solution and Assumptions (A4)-(A5) are satisfied.

Problem (30) is convex and the sum-power constraint coupling $\mathbf{Q}_1,\ldots,\mathbf{Q}_K$ is separable, so dual decomposition techniques can be used [27]. In particular, the constraint set has a nonempty interior, so strong duality holds and (30) can be solved from the dual domain by relaxing the sum-power constraint into the Lagrangian [22]:

$$\mathbb{B}\mathbf{Q}^t = \underset{(\mathbf{Q}_k\succeq\mathbf{0})_{k=1}^K}{\arg\max} \left\{ \begin{array}{l} \sum_{k=1}^K \log\left|\mathbf{R}_k(\mathbf{Q}^t_{-k})+\mathbf{H}_k\mathbf{Q}_k\mathbf{H}_k^H\right| \\ -\lambda^\star(\sum_{k=1}^K \text{tr}(\mathbf{Q}_k)-P) \end{array} \right\}. \quad (31)$$

where $\mathbb{B}\mathbf{Q}^t = (\mathbb{B}_k\mathbf{Q}^t)_{k=1}^K$ and $\lambda^\star$ is the optimal Lagrange multiplier that satisfies the following conditions: $\lambda^\star \ge 0$, $\sum_{k=1}^K \text{tr}(\mathbb{B}_k\mathbf{Q}^t)-P \le 0$, $\lambda^\star(\sum_{k=1}^K \text{tr}(\mathbb{B}_k\mathbf{Q}^t)-P)=0$, and can be found efficiently using the bisection method .

The problem in (31) is uncoupled among different variables $\mathbf{Q}_k$ in both the objective function and the constraint set, so it can be decomposed into a set of smaller subproblems which are solved in parallel: $\mathbb{B}\mathbf{Q}^t = (\mathbb{B}_k\mathbf{Q}^t)_{k=1}^K$ and

$$\mathbb{B}_k\mathbf{Q}^t = \underset{\mathbf{Q}_k\succeq\mathbf{0}}{\arg\max}\left\{\log\left|\mathbf{R}_k(\mathbf{Q}^t_{-k})+\mathbf{H}_k\mathbf{Q}_k\mathbf{H}_k^H\right| - \lambda^\star\text{tr}(\mathbf{Q}_k)\right\}, \quad (32)$$

and $\mathbb{B}_k\mathbf{Q}^t$ exhibits a closed-form expression based on the waterfilling solution [2]. Thus problem (30) also has a closed-form solution up to a Lagrange multiplier that can be found efficiently using the bisection method. With the update direction $\mathbb{B}\mathbf{Q}^t-\mathbf{Q}^t$, the base station can implement the exact line search to determine the stepsize using the bisection method described after (13) in Sec. III.

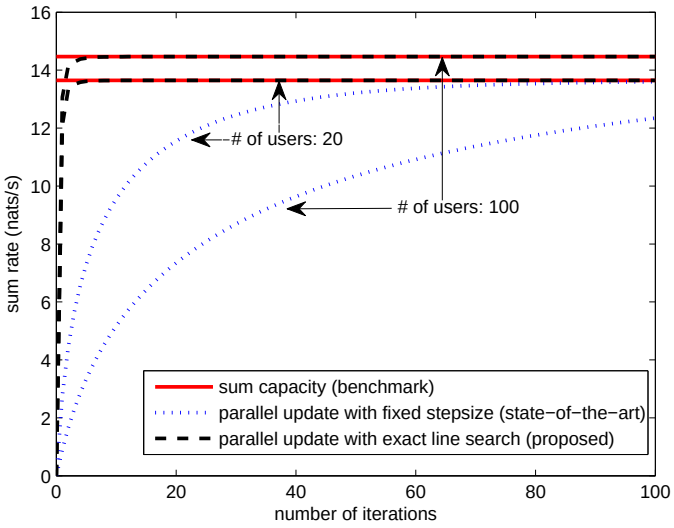

Figure 2. MIMO BC: sum-rate versus the number of iterations.

We remark that when the channel matrices $\mathbf{H}_k$ are rank deficient, problem (30) is convex but not strongly convex, but the proposed algorithm with the approximate problem (30) still converges. However, if the approximate function in [8] is used, an additional quadratic regularization term must be included into (30) (and thus (32)) to make the approximate problem strongly convex and to guarantee t, cf. (26), but the resulting approximate problem no longer exhibits a closed-form solution and thus are much more difficult to solve.

Simulations. The parameters are set as follows. The number of users is $K=20$ and $K=100$, the number of transmit and receive antenna is (5,4), and $P=10$ dB. The simulation results are averaged over 20 instances.

We apply Algorithm 1 with approximate problem (30) and stepsize based on the exact line search, and compare it with the iterative algorithm proposed in [2, 16], which uses the same approximate problem (30) but with a fixed stepsize $\gamma^t=1/K$ (K is the number of users). It is easy to see from Fig. 2 that the proposed method converges very fast (in less than 10 iterations) to the sum capacity, while the method of [2] requires many more iterations. This is due to the benefit of the exact line search applied in our algorithm over the fixed stepsize tends to be overly conservative. Employing the exact line search adds complexity as compared to the simple choice of a fixed stepsize, however, since the objective function of (29) is concave, the exact line search consists in maximizing a differentiable concave function with a scalar variable, and it can be solved efficiently by the bisection method with affordable cost. More specifically, it takes 0.0023 seconds to solve problem (30) and 0.0018 seconds to perform the exact line search (the software/hardware environment is further specified in Sec. IV-B). Therefore, the overall CPU time (time per iteration$\times$number of iterations) is still dramatically decreased due to the notable reduction in the number of iterations. Besides, in contrast to the method of [2], increasing the number of users K does not slow down the convergence, so the proposed algorithm is scalable in large networks.

We also compare the proposed algorithm with the iterative

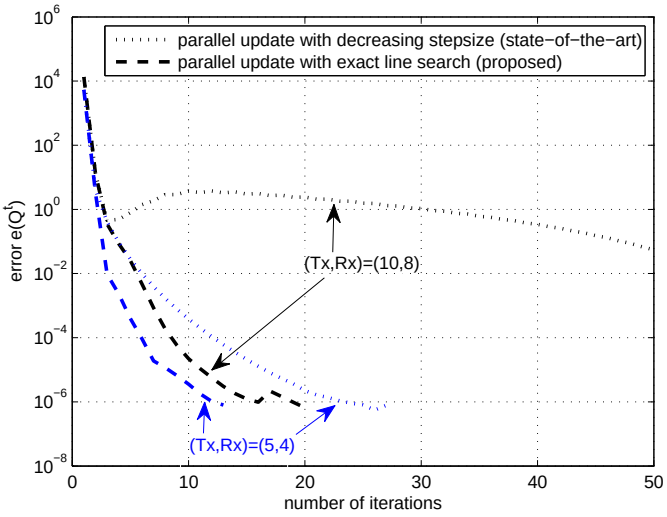

Figure 3. MIMO BC: error $e(\mathbf{Q}^t) = \Re(\mathrm{tr}(\nabla f(\mathbf{Q}^t) \bullet (\mathbb{B}\mathbf{Q}^t - \mathbf{Q}^t)))$ versus the number of iterations.

algorithm of [18], which uses the approximate problem (30) but with an additional quadratic regularization term, cf. (26), where $\tau_k = 10^{-5}$ for all k, and decreasing stepsizes $\gamma^{t+1} = \gamma^t(1-d\gamma^t)$ where $d = 0.01$ is the so-called decreasing rate that controls the rate of decrease in the stepsize. We can see from Fig. 3 that the convergence behavior of [18] is rather sensitive to the decreasing rate d. The choice $d = 0.01$ performs well when the number of transmit and receive antennas is 5 and 4, respectively, but it is no longer a good choice when the number of transmit and receive antenna increases to 10 and 8, respectively. A good decreasing rate d is usually dependent on the problem parameters and no general rule performs equally well for all choices of parameters.

We remark once again that the complexity of each iteration of the proposed algorithm is very low because of the existence of a closed-form solution to the approximate problem (30), while the approximate problem proposed in [18] does not exhibit a closed-form solution and can only be solved iteratively. Specifically, it takes `CVX` (version 2.0 [29]) 21.1785 seconds (based on the dual approach (32) where $\lambda^\star$ is found by bisection). Therefore, the overall complexity per iteration of the proposed algorithm is much lower than that of [18].

B. LASSO

In this subsection and the following subsection, we study the LASSO problem to illustrate the advantage of the proposed line search method for nondifferentiable optimization problems.

LASSO is an important and widely studied problem in sparse signal recovery [11, 12, 30, 31]:

$$\underset{\mathbf{x}}{\text{minimize}} \quad \tfrac{1}{2}\left\|\mathbf{A}\mathbf{x}-\mathbf{b}\right\|_2^2 + \mu\left\|\mathbf{x}\right\|_1, \tag{33}$$

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 (33) is an instance of the general problem structure defined in (2) with the following decomposition:

$$f(\mathbf{x}) \triangleq \tfrac{1}{2}\left\|\mathbf{A}\mathbf{x}-\mathbf{b}\right\|_2^2, \quad \text{and} \quad g(\mathbf{x}) \triangleq \mu\left\|\mathbf{x}\right\|_1. \tag{34}$$

Problem (33) is convex, but its objective function is nondifferentiable and it does not have a closed-form solution. To apply Algorithm 2, the scalar decomposition $\mathbf{x} = (x_k)_{k=1}^K$ is adopted. Recalling (22) and (26), the approximate problem is

$$\mathbb{B}\mathbf{x}^t = \arg\min_{\mathbf{x}} \left\{\textstyle\sum_{k=1}^K f(x_k, \mathbf{x}_{-k}^t) + g(\mathbf{x})\right\}. \tag{35}$$

Note that $g(\mathbf{x})$ can be decomposed among different components of $\mathbf{x}$, i.e., $g(\mathbf{x}) = \sum_{k=1}^K g(x_k)$, so the vector problem (35) reduces to K independent scalar problem that can be solved in parallel:

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

where $d_k(\mathbf{A}^T\mathbf{A})$ is the k-th diagonal element of $\mathbf{A}^T\mathbf{A}$, $\mathcal{S}_{\mathbf{a}}(\mathbf{b}) \triangleq [\mathbf{b}-\mathbf{a}]^+ - [-\mathbf{b}-\mathbf{a}]^+$ is the so-called soft-thresholding operator [31] and

$$\mathbf{r}(\mathbf{x}) \triangleq \mathbf{d}(\mathbf{A}^T\mathbf{A}) \circ \mathbf{x} - \mathbf{A}^T(\mathbf{A}\mathbf{x}-\mathbf{b}), \tag{36}$$

or more compactly:

$$\mathbb{B}\mathbf{x}^t = (\mathbb{B}_k\mathbf{x}^t)_{k=1}^K = \mathbf{d}(\mathbf{A}^T\mathbf{A})^{-1} \circ \mathcal{S}_{\mu\mathbf{1}}(\mathbf{r}(\mathbf{x}^t)). \tag{37}$$

Thus the update direction exhibits a closed-form expression. The stepsize based on the proposed exact line search (23) is

$$\begin{aligned}\gamma^t &= \arg\min_{0\le\gamma\le 1} \left\{f(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) + \gamma\big(g(\mathbb{B}\mathbf{x}^t) - g(\mathbf{x}^t)\big)\right\} \\ &= \arg\min_{0\le\gamma\le 1} \left\{\begin{array}{l}\frac{1}{2}\left\|\mathbf{A}(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) - \mathbf{b}\right\|_2^2 \\ \quad + \gamma\,\mu\big(\left\|\mathbb{B}\mathbf{x}^t\right\|_1 - \left\|\mathbf{x}^t\right\|_1\big)\end{array}\right\} \\ &= \left[-\frac{(\mathbf{A}\mathbf{x}^t-\mathbf{b})^T\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t) + \mu(\left\|\mathbb{B}\mathbf{x}^t\right\|_1 - \left\|\mathbf{x}^t\right\|_1)}{(\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))^T(\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))}\right]_0^1.\end{aligned} \tag{38}$$

The exact line search consists in a convex quadratic optimization problem with a scalar variable and a bound constraint, so it exhibits a closed-form solution (38). Therefore, both the update direction and stepsize can be calculated in closed-form. We name the proposed update (37)-(38) as Soft-Thresholding with Exact Line search Algorithm (`STELA`).

The proposed update (37)-(38) has several desirable features that make it appealing in practice. Firstly, in each iteration, all elements are updated in parallel based on the nonlinear best-response (37). This is in the same spirit as [17] and the convergence speed is generally faster than BCD [32] or the gradient-based update [33]. Secondly, the proposed exact line search (38) not only yields notable progress in each iteration but also enjoys an easy implementation given the closed-form expression. The convergence speed is thus further enhanced as compared to the procedure proposed in [17] where decreasing stepsizes are used.

Computational complexity. The computational overhead associated with the proposed exact line search (38) can significantly be reduced if (38) is carefully implemented as outlined in the following. The most complex operation in (38) 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 the one hand, the term $\mathbf{A}\mathbf{x}^t - \mathbf{b}$ is already available from $\mathbf{r}(\mathbf{x}^t)$, which

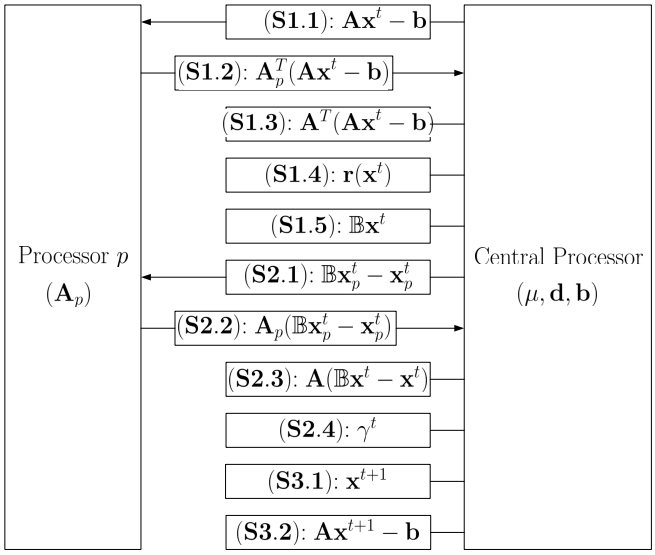

Figure 4. Operation flow and signaling exchange between local processor p and the central processor. A solid line indicates the computation that is locally performed by the central/local processor, and a solid line with an arrow indicates signaling exchange between the central and local processor and the direction of the signaling exchange.

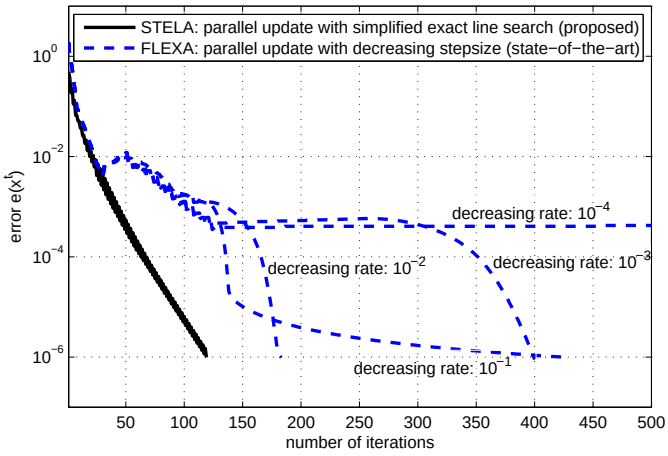

Figure 5. Convergence of STELA (proposed) and FLEXA (state-of-the-art) for LASSO: error versus the number of iterations.

is computed in order to determine the best-response in (37). On the other hand, the matrix-vector multiplication $\mathbf{A}(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)$ is also required for the computation of $\mathbf{A}\mathbf{x}^{t+1} - \mathbf{b}$ as it can alternatively be computed as:

$$\begin{aligned}\mathbf{A}\mathbf{x}^{t+1} - \mathbf{b} &= \mathbf{A}(\mathbf{x}^t + \gamma^t(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) - \mathbf{b} \\ &= (\mathbf{A}\mathbf{x}^t - \mathbf{b}) + \gamma^t\mathbf{A}(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t), \qquad (39)\end{aligned}$$

where only vector addition is involved. As a result, the stepsize (38) does not incur any 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 processing unit, a parallel architecture can be employed. Assume there are $P+1$ ($P \geq 2$) processors. We label the first P processors as local processors and the last one as the central processor, and partition $\mathbf{A}$ as

$$\mathbf{A} = [\mathbf{A}_1,\ \mathbf{A}_2,\ \ldots\ ,\mathbf{A}_P],$$

where $\mathbf{A}_p \in \mathbb{R}^{N\times K_p}$ and $\sum_{p=1}^P K_p = K$. Matrix $\mathbf{A}_p$ is stored and processed in the local processor p, and the following computations are decomposed among the local processors:

$$\mathbf{A}\mathbf{x} = \textstyle\sum_{p=1}^P \mathbf{A}_p\mathbf{x}_p, \qquad (40a)$$

$$\mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) = \left(\mathbf{A}_p^T(\mathbf{A}\mathbf{x} - \mathbf{b})\right)_{p=1}^P, \qquad (40b)$$

$$\mathbf{d}(\mathbf{A}^T\mathbf{A}) = (\mathbf{d}(\mathbf{A}_p^T\mathbf{A}_p))_{p=1}^P. \qquad (40c)$$

where $\mathbf{x}_p \in \mathbb{R}^{K_p}$. The central processor computes the best-response $\mathbb{B}\mathbf{x}^t$ in (37) and the stepsize γ^t in (38). The decomposition in (40) enables us to analyze the signaling exchange between local processor p and the central processor involved in (37) and (38)[1].

[1]Updates (37) and (38) can also be implemented by a parallel architecture without a central processor. In this case, the signaling is exchanged mutually between every two of the local processors, but the analysis is similar and the conclusion to be drawn remains same: the proposed exact line search (38) does not incur additional signaling compared with predetermined stepsizes.

The signaling exchange is summarized in Fig. 4. Firstly, the central processor sends $\mathbf{A}\mathbf{x}^t - \mathbf{b}$ to each local processor p (**S1.1**)[2], and the local processor p first computes $\mathbf{A}_p^T(\mathbf{A}\mathbf{x}^t - \mathbf{b})$ and then sends it back to the central processor (**S1.2**), which forms $\mathbf{A}^T(\mathbf{A}\mathbf{x}^t - \mathbf{b})$ (**S1.3**) as in (40b) and calculates $\mathbf{r}(\mathbf{x}^t)$ as in (36) (**S1.4**) and then $\mathbb{B}\mathbf{x}^t$ as in (37) (**S1.5**). Then the central processor sends $\mathbb{B}\mathbf{x}_p^t - \mathbf{x}_p^t$ to each processor p (**S2.1**), and each processor first computes $\mathbf{A}_p(\mathbb{B}\mathbf{x}_p^t - \mathbf{x}_p^t)$ and then sends it back to the central processor (**S2.2**), which forms $\mathbf{A}(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)$ (**S2.3**) as in (40a), calculates γ^t as in (38) (**S2.4**), and updates $\mathbf{x}^{t+1}$ (**S3.1**) and $\mathbf{A}\mathbf{x}^{t+1} - \mathbf{b}$ (**S3.2**) according to (39). From Fig. 4 we observe that the exact line search (38) does *not* incur any additional signaling compared with that of predetermined stepsizes (e.g., constant and decreasing stepsize), because the signaling exchange in **S2.1-S2.2** has also to be carried out in the computation of $\mathbf{A}\mathbf{x}^{t+1} - \mathbf{b}$ in **S3.2**, cf. (39).

We finally remark that the proposed successive line search can be applied and it exhibits a closed-form expression as well. However, since the exact line search yields faster convergence, we omit the details at this point.

Simulations. We first compare in Fig. 5 the proposed algorithm STELA with FLEXA [17] in terms of the error criterion $e(\mathbf{x}^t)$ defined as:

$$e(\mathbf{x}^t) \triangleq \left\|\nabla f(\mathbf{x}^t) - \left[\nabla f(\mathbf{x}^t) - \mathbf{x}^t\right]_{-\mu\mathbf{1}}^{\mu\mathbf{1}}\right\|_2. \qquad (41)$$

Note that $\mathbf{x}^\star$ is a solution of (33) if and only if $e(\mathbf{x}^\star) = 0$ [26]. FLEXA is implemented as outlined in [17]; however, the selective update scheme [17] is not implemented in FLEXA because it is also applicable for STELA and it cannot eliminate the slow convergence and sensitivity of the decreasing stepsize. We also remark that the stepsize rule for FLEXA is $\gamma^{t+1} = \gamma^t(1-\min(1, 10^{-4}/e(\mathbf{x}^t))d\gamma^t)$ [17], where d is the decreasing rate and $\gamma^0 = 0.9$. The code and the data generating the figure can be downloaded online [34].

Note that the error $e(\mathbf{x}^t)$ plotted in Fig. 5 does not necessarily decrease monotonically while the objective function $f(\mathbf{x}^t) + g(\mathbf{x}^t)$ always does. This is because STELA and FLEXA are descent direction methods. For FLEXA, when

[2]$\mathbf{x}^0$ is set to $\mathbf{x}^0 = \mathbf{0}$, so $\mathbf{A}\mathbf{x}^0 - \mathbf{b} = -\mathbf{b}$.

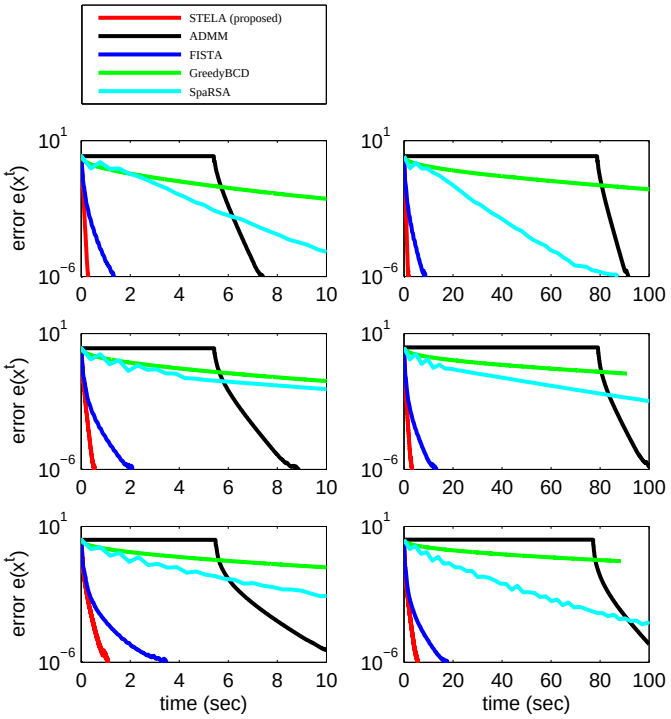

Figure 6. Time versus error of different algorithms for LASSO. In the left and right column, the dimension of $\mathbf{A}$ is 2000×4000 and 5000×10000, respectively. In the higher, middle and lower column, the density of $\mathbf{x}_{\text{true}}$ is 0.1, 0.2 and 0.4.

the decreasing rate is low ($d = 10^{-4}$), no improvement is observed after 100 iterations. As a matter of fact, the stepsize in those iterations is so large that the function value is actually dramatically increased, and thus the associated iterations are discarded in Fig. 5. A 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, the choice $d = 10^{-2}$ performs well, but the value of a good decreasing rate is parameters dependent (e.g., $\mathbf{A}$, $\mathbf{b}$ and μ) and no general rule performs equally well for all choices of parameters. By comparison, the proposed algorithm `STELA` is fast to converge and exhibits stable performance.

We also compare in Fig. 6 the proposed algorithm `STELA` with other competitive algorithms in literature: `FISTA` [31], `ADMM` [12], `GreedyBCD` [35] and `SpaRSA` [36]. We simulated `GreedyBCD` of [35] because it exhibits guaranteed convergence. The dimension of $\mathbf{A}$ is 2000×4000 (the left column of Fig. 6) and 5000×10000 (the right column). It is generated by the Matlab command `randn` with each row being normalized to unity. The density (the proportion of nonzero elements) of the sparse vector $\mathbf{x}_{\text{true}}$ is 0.1 (the upper row of Fig. 6), 0.2 (the middle row) and 0.4 (the lower row). The vector $\mathbf{b}$ is generated as $\mathbf{b} = \mathbf{A}\mathbf{x}_{\text{true}} + \mathbf{e}$ where $\mathbf{e}$ is drawn from a Gaussian white distribution with variance 10^{-4}. The regularization gain μ is set to $\mu = 0.1 \left\|\mathbf{A}^T\mathbf{b}\right\|_\infty$, which allows $\mathbf{x}_{\text{true}}$ to be recovered to a high accuracy [36].

The simulations are carried out under Matlab R2012a on a PC equipped with an operating system of Windows 7 64-bit Home Premium Edition, an Intel i5-3210 2.50GHz CPU, and a 8GB RAM. All of the Matlab codes are available online [34]. The comparison is made in terms of CPU time that is required until either a given error bound $e(\mathbf{x}^t) \le 10^{-6}$ is reached or the maximum number of iterations, namely, 2000, is reached. 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 the proposed algorithm `STELA`, $\mathbf{d}(\mathbf{A}^T\mathbf{A})$ is computed[3] in the initialization stage since it is required in the iterative variable update in the formal stage, cf. (37). The simulation results are averaged over 20 instances.

We observe from Fig. 6 that the proposed algorithm `STELA` converges faster than all competing algorithms. Some further observations are in order.

- 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 increases negligibly.
- The proposed algorithm `STELA` scales relatively well with the problem dimension. When the dimension of $\mathbf{A}$ is increased from 2000×4000 (the left column) to 5000×10000 (the right column), the CPU time is only marginally increased.
- The initialization stage of `ADMM` is time consuming because of some expensive matrix operations as, 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 [12, Sec. 6.4]. Furthermore, the CPU time of the initialization stage of `ADMM` is increased dramatically when the dimension of $\mathbf{A}$ is increased from 2000×4000 to 5000×10000.
- `SpaRSA` performs 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.
- The asymptotic convergence speed of `GreedyBCD` is slow, because only one variable is updated in each iteration.

C. Nonconvex LASSO

In this subsection, we show that the fast convergence behavior observed in convex problems studied in Sec. IV-A and Sec. IV-B extends to nonconvex optimization problems.

We consider the following nonconvex LASSO problem introduced in [17]:

$$\underset{\mathbf{x}}{\text{minimize}} \quad \tfrac{1}{2}\left\|\mathbf{A}\mathbf{x} - \mathbf{b}\right\|_2^2 - \tfrac{1}{2}c\left\|\mathbf{x}\right\|_2^2 + \mu\left\|\mathbf{x}\right\|_1, \qquad (42)$$

where $\mathbf{A} \in \mathbb{R}^{N\times N}$ (with $N \ll K$), $\mathbf{b} \in \mathbb{R}^{N\times 1}$, $c > 0$, and $\mu > 0$ are given parameters. Problem (42) is an instance of (2) with the following decomposition:

$$f(\mathbf{x}) \triangleq \tfrac{1}{2}\left\|\mathbf{A}\mathbf{x} - \mathbf{b}\right\|_2^2 - \tfrac{1}{2}c\left\|\mathbf{x}\right\|_2^2, \text{ and } g(\mathbf{x}) \triangleq \mu\left\|\mathbf{x}\right\|_1. \quad (43)$$

Since the minimum eigenvalue of $\mathbf{A}^T\mathbf{A}$ is 0 and $\mathbf{A}^T\mathbf{A} - c\mathbf{I} \not\succeq \mathbf{0}$, $f(\mathbf{x})$ defined in (43) is nonconvex and so is problem (42). To apply Algorithm 2, we adopt the following approximate function:

$$\tilde{f}(\mathbf{x};\mathbf{x}^t) = \textstyle\sum_{k=1}^K (f_1(x_k, \mathbf{x}_{-k}^t) + \nabla_k f_2(\mathbf{x}^t)(x_k - x_k^t) + g(x_k)), \qquad (44)$$

[3]The Matlab command is `sum(A.^2,1)`, so matrix-matrix multiplication between $\mathbf{A}^T$ and $\mathbf{A}$ is not required.

where $f_1(\mathbf{x}) \triangleq \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2$ and $f_2(\mathbf{x}) \triangleq -\frac{1}{2}c\|\mathbf{x}\|_2^2$. Note that different from the approximation function (35) used in LASSO, the approximate function of $f(\mathbf{x})$ in (44) comes from preserving the convex part of $f(\mathbf{x})$, namely, $f_1(\mathbf{x})$, while linearizing the nonconvex part of $f(\mathbf{x})$, namely, $f_2(\mathbf{x})$.

Since the approximate function (44) can be decomposed among different elements of $\mathbf{x}$ and problem (42) is unconstrained, the approximate problem consists of K scalar problems, which are then solved in parallel:

$$\begin{aligned}\mathbb{B}_k\mathbf{x}^t &= \arg\min_{x_k}\{f_1(x_k,\mathbf{x}_{-k}^t)+\nabla_k f_2(\mathbf{x}^t)(x_k-x_k^t)+g(x_k)\}\\ &= d_k(\mathbf{A}^T\mathbf{A})^{-1}\mathcal{S}_\mu(r_k(\mathbf{x}^t)+cx_k^t),\ k=1,\ldots,K,\end{aligned}$$

where $\mathcal{S}_{\mathbf{a}}(\mathbf{b})$ and $\mathbf{r}(\mathbf{x})$ is defined in (36), or more compactly:

$$\mathbb{B}\mathbf{x}^t = \mathbf{d}(\mathbf{A}^T\mathbf{A})^{-1}\circ\mathcal{S}_{\mu\mathbf{1}}(\mathbf{r}(\mathbf{x}^t)+c\mathbf{x}^t). \tag{45}$$

The stepsize based on the proposed exact line search (23) is

$$\begin{aligned}\gamma^t &= \arg\min_{0\le\gamma\le1}\{f(\mathbf{x}^t+\gamma(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))+\gamma(g(\mathbb{B}\mathbf{x}^t)-g(\mathbf{x}^t))\}\\ &= \arg\min_{0\le\gamma\le1}\left\{\begin{array}{l}\frac{1}{2}(\|\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)\|_2^2-c\|\mathbb{B}\mathbf{x}^t-\mathbf{x}^t\|_2^2)\cdot\gamma^2\\ +((\mathbf{A}\mathbf{x}^t-\mathbf{b})^T\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)-c\mathbf{x}^t)\cdot\gamma\\ +(\mu(\|\mathbb{B}\mathbf{x}^t\|_1-\|\mathbf{x}^t\|_1))\cdot\gamma\end{array}\right\},\end{aligned} \tag{46}$$

which consists in minimizing a quadratic function with a scalar variable subject to a bound constraint.

Suppose $\|\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)\|_2^2-c\|\mathbb{B}\mathbf{x}^t-\mathbf{x}^t\|_2^2<0$. Then the objective function in (46) is concave. Since $\mathbb{B}\mathbf{x}^t-\mathbf{x}^t$ is a descent direction, it must be that $\gamma^t>0$ and thus

$$\nabla_\gamma f(\mathbf{x}^t+\gamma(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))+\gamma(g(\mathbb{B}\mathbf{x}^t)-g(\mathbf{x}^t))\big|_{\gamma=0}<0.$$

In this case, γ^t in (46) is given as follows:

$$\gamma^t=1,\ \text{if}\ \left\|\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)\right\|_2^2-c\left\|\mathbb{B}\mathbf{x}^t-\mathbf{x}^t\right\|_2^2<0. \tag{47a}$$

If, on the other hand, $\|\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)\|_2^2-c\|\mathbb{B}\mathbf{x}^t-\mathbf{x}^t\|_2^2\ge0$, the optimization problem in (46) is convex and

$$\gamma^t=\left[-\frac{(\mathbf{A}\mathbf{x}^t-\mathbf{b})^T\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)-c\mathbf{x}^t+\mu(\|\mathbb{B}\mathbf{x}^t\|_1-\|\mathbf{x}^t\|_1)}{\|\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t)\|_2^2-c\|\mathbb{B}\mathbf{x}^t-\mathbf{x}^t\|_2^2}\right]_0^1. \tag{47b}$$

Therefore, both the update direction and the stepsize can be found by closed-form expressions, namely, (45) and (47).

Simulations. We test the convergence speed and scalability of STELA for the nonconvex LASSO problem (42), where the update direction and the stepsize is given by (45) and (47), respectively. The parameters $\mathbf{A}$, $\mathbf{b}$, and μ are generated in the same way as the LASSO problem in Sec. IV-B, except that the rows of $\mathbf{A}$ are not normalized. The density of $\mathbf{x}_{\text{true}}$ is 0.2, and $c=N/200$. The dimension of $\mathbf{A}$ is 2000×4000, 5000×10000, and 10000×20000, respectively.

From Fig. 7, we observe that, as expected, the objective function value is monotonically decreasing, because STELA is an instance of Algorithm 2 which is essentially an iterative descent direction method. The convergence to a (local) minimum is observed in less than 10 iterations, even when the dimension of $\mathbf{A}$ is as large as 10000×20000. More importantly, by comparing the different curves in Fig. 7, the convergence speed seems independent of the dimension of $\mathbf{A}$, so the proposed STELA scales very well. Furthermore, due to the closed-form updates, the iteration complexity is extremely low. Therefore, the fast convergence, easy implementation and scalability observed in LASSO in Sec. IV-B extend to nonconvex optimization problems as well.

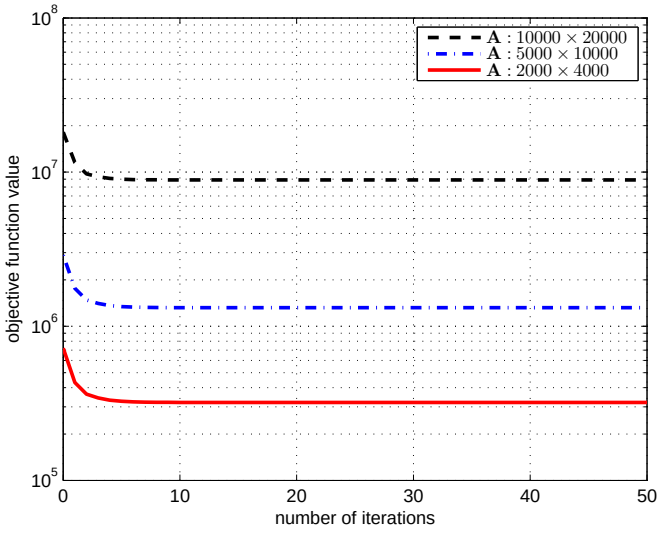

Figure 7. Convergence of STELA for nonconvex LASSO: the objective value versus the number of iterations.

V. Concluding Remarks

In this paper, we have proposed a novel iterative algorithm based on convex approximation. The only requirement on the approximate function is that it is pseudo-convex. On the one hand, the relaxation of the assumptions on the approximate functions can make the approximate problems much easier to solve. We show by a counter-example that the assumption on pseudo-convexity is tight in the sense that when it is violated, the algorithm may not converge. On the another hand, the stepsize based on the exact/successive line search yields notable progress in each iteration. Additional structures can be exploited to assist with the selection of the stepsize, so that the algorithm can be further accelerated. The advantages and benefits of the proposed algorithm have been demonstrated using prominent applications in communication networks and signal processing, and they are also numerically consolidated. The proposed algorithm can readily be applied to solve other problems as well, such as portfolio optimization [10].

Appendix A
Proof of Proposition 1

Proof: i) Firstly, suppose $\mathbf{y}$ is a stationary point of (1); it satisfies the first-order optimality condition:

$$\nabla f(\mathbf{y})^T(\mathbf{x}-\mathbf{y})\ge0,\ \forall\mathbf{x}\in\mathcal{X}.$$

Using Assumption (A3), we get

$$\nabla\tilde{f}(\mathbf{y};\mathbf{y})^T(\mathbf{x}-\mathbf{y})\ge0,\ \forall\mathbf{x}\in\mathcal{X}.$$

Since $\tilde{f}(\bullet;\mathbf{y})$ is pseudo-convex, the above condition implies

$$\tilde{f}(\mathbf{x};\mathbf{y})\ge\tilde{f}(\mathbf{y};\mathbf{y}),\ \forall\mathbf{x}\in\mathcal{X}.$$

That is, $\tilde{f}(\mathbf{y};\mathbf{y}) = \min_{\mathbf{x}\in\mathcal{X}} \tilde{f}(\mathbf{x};\mathbf{y})$ and $\mathbf{y} \in \mathcal{S}(\mathbf{y})$.

Secondly, suppose $\mathbf{y} \in \mathcal{S}(\mathbf{y})$. We readily get

$$\nabla f(\mathbf{y})^T(\mathbf{x}-\mathbf{y}) = \nabla \tilde{f}(\mathbf{y};\mathbf{y})^T(\mathbf{x}-\mathbf{y}) \geq 0,\ \forall \mathbf{x} \in \mathcal{X}, \quad (48)$$

where the equality and inequality comes from Assumption (A3) and the first-order optimality condition, respectively, so $\mathbf{y}$ is a stationary point of (1).

ii) From the definition of $\mathbb{B}\mathbf{x}$, it is either

$$\tilde{f}(\mathbb{B}\mathbf{y};\mathbf{y}) = \tilde{f}(\mathbf{y};\mathbf{y}), \quad (49a)$$

or

$$\tilde{f}(\mathbb{B}\mathbf{y};\mathbf{y}) < \tilde{f}(\mathbf{y};\mathbf{y}), \quad (49b)$$

If (49a) is true, then $\mathbf{y} \in \mathcal{S}(\mathbf{y})$ and, as we have just shown, it is a stationary point of (1). So only (49b) can be true. We know from the pseudo-convexity of $\tilde{f}(\mathbf{x};\mathbf{y})$ in $\mathbf{x}$ (cf. Assumption (A1)) and (49b) that $\mathbb{B}\mathbf{y} \neq \mathbf{y}$ and

$$\nabla \tilde{f}(\mathbf{y};\mathbf{y})^T(\mathbb{B}\mathbf{y}-\mathbf{y}) = \nabla f(\mathbf{y})^T(\mathbb{B}\mathbf{y}-\mathbf{y}) < 0, \quad (50)$$

where the equality comes from Assumption (A3). ■

APPENDIX B
PROOF OF THEOREM 2

Proof: Since $\mathbb{B}\mathbf{x}^t$ is the optimal point of (8), it satisfies the first-order optimality condition:

$$\nabla \tilde{f}(\mathbb{B}\mathbf{x}^t;\mathbf{x}^t)^T(\mathbf{x}-\mathbb{B}\mathbf{x}^t) \geq 0,\ \forall \mathbf{x} \in \mathcal{X}. \quad (51)$$

If (49a) is true, then $\mathbf{x}^t \in \mathcal{S}(\mathbf{x}^t)$ and it is a stationary point of (1) according to Proposition 1 (i). Besides, it follows from (48) (with $\mathbf{x} = \mathbb{B}\mathbf{x}^t$ and $\mathbf{y} = \mathbf{x}^t$) that $\nabla f(\mathbf{x}^t)^T(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t) \geq 0$. Note that equality is actually achieved, i.e.,

$$\nabla f(\mathbf{x}^t)^T(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t) = 0$$

because otherwise $\mathbb{B}\mathbf{x}^t - \mathbf{x}^t$ would be an ascent direction of $\tilde{f}(\mathbf{x};\mathbf{x}^t)$ at $\mathbf{x} = \mathbf{x}^t$ and the definition of $\mathbb{B}\mathbf{x}^t$ would be contradicted. Then from the definition of the successive line search, we can readily infer that

$$f(\mathbf{x}^{t+1}) \leq f(\mathbf{x}^t). \quad (52)$$

It is easy to see (52) holds for the exact line search as well.

If (49b) is true, $\mathbf{x}^t$ is not a stationary point and $\mathbb{B}\mathbf{x}^t - \mathbf{x}^t$ is a strict descent direction of $f(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}^t$ according to Proposition 1 (ii): $f(\mathbf{x})$ is strictly decreased compared with $f(\mathbf{x}^t)$ if $\mathbf{x}$ is updated at $\mathbf{x}^t$ along the direction $\mathbb{B}\mathbf{x}^t - \mathbf{x}^t$. From the definition of the successive line search, there always exists a γ^t such that $0 < \gamma^t \leq 1$ and

$$f(\mathbf{x}^{t+1}) = f(\mathbf{x}^t + \gamma^t(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) < f(\mathbf{x}^t). \quad (53)$$

This strict decreasing property also holds for the exact line search because it is the stepsize that yields the largest decrease, which is always larger than or equal to that of the successive line search.

We know from (52) and (53) that $\{f(\mathbf{x}^t)\}$ is a monotonically decreasing sequence and it thus converges. Besides, for any two (possibly different) convergent subsequences $\{\mathbf{x}^t\}_{t\in\mathcal{T}_1}$ and $\{\mathbf{x}^t\}_{t\in\mathcal{T}_2}$, the following holds:

$$\lim_{t\to\infty} f(\mathbf{x}^t) = \lim_{\mathcal{T}_1 \ni t\to\infty} f(\mathbf{x}^t) = \lim_{\mathcal{T}_2 \ni t\to\infty} f(\mathbf{x}^t).$$

Since $f(\mathbf{x})$ is a continuous function, we infer from the preceding equation that

$$f\left(\lim_{\mathcal{T}_1 \ni t\to\infty} \mathbf{x}^t\right) = f\left(\lim_{\mathcal{T}_2 \ni t\to\infty} \mathbf{x}^t\right). \quad (54)$$

Now consider any convergent subsequence $\{\mathbf{x}^t\}_{t\in\mathcal{T}}$ with limit point $\mathbf{y}$, i.e., $\lim_{\mathcal{T}\ni t\to\infty} \mathbf{x}^t = \mathbf{y}$. To show that $\mathbf{y}$ is a stationary point, we first assume the contrary: $\mathbf{y}$ is not a stationary point. Since $\tilde{f}(\mathbf{x};\mathbf{x}^t)$ is continuous in both $\mathbf{x}$ and $\mathbf{x}^t$ by Assumption (A2) and $\{\mathbf{B}\mathbf{x}^t\}_{t\in\mathcal{T}}$ is bounded by Assumption (A5), it follows from [24, Th. 1] that there exists a sequence $\{\mathbb{B}\mathbf{x}^t\}_{t\in\mathcal{T}_s}$ with $\mathcal{T}_s \subseteq \mathcal{T}$ such that it converges and $\lim_{\mathcal{T}_s\ni t\to\infty} \mathbb{B}\mathbf{x}^t \in \mathcal{S}(\mathbf{y})$. Since both $f(\mathbf{x})$ and $\nabla f(\mathbf{x})$ are continuous, applying [24, Th. 1] again implies there is a $\mathcal{T}_{s'}$ such that $\mathcal{T}_{s'} \subseteq \mathcal{T}_s (\subseteq \mathcal{T})$ and $\{\mathbf{x}^{t+1}\}_{t\in\mathcal{T}_{s'}}$ converges to $\mathbf{y}'$ defined as:

$$\mathbf{y}' \triangleq \mathbf{y} + \rho(\mathbb{B}\mathbf{y} - \mathbf{y}),$$

where ρ is the stepsize when either the exact or successive line search is applied to $f(\mathbf{y})$ along the direction $\mathbb{B}\mathbf{y} - \mathbf{y}$. Since $\mathbf{y}$ is not a stationary point, it follows from (53) that $f(\mathbf{y}') < f(\mathbf{y})$, but this would contradict (54). Therefore $\mathbf{y}$ is a stationary point, and the proof is completed. ■

APPENDIX C
PROOF OF THEOREM 3

Proof: We first need to show that Proposition 1 still holds.

(i) We prove $\mathbf{y}$ is a stationary point of (4) if and only if $\mathbf{y}_k \in \arg\min_{\mathbf{x}_k\in\mathcal{X}_k} f(\mathbf{x}_k, \mathbf{y}_{-k})$ for all k.

Suppose $\mathbf{y}$ is a stationary point of (4), it satisfies the first-order optimality condition:

$$\nabla f(\mathbf{y})^T(\mathbf{x}-\mathbf{y}) = \textstyle\sum_{k=1}^K \nabla_k f(\mathbf{y})^T(\mathbf{x}_k - \mathbf{y}_k) \geq 0, \forall \mathbf{x} \in \mathcal{X},$$

and it is equivalent to

$$\nabla_k f(\mathbf{y})^T(\mathbf{x}_k - \mathbf{y}_k) \geq 0, \forall \mathbf{x}_k \in \mathcal{X}_k.$$

Since $f(\mathbf{x})$ is pseudo-convex in $\mathbf{x}_k$, the above condition implies $f(\mathbf{y}_k, \mathbf{y}_{-k}) = \min_{\mathbf{x}_k\in\mathcal{X}_k} f(\mathbf{x}_k, \mathbf{y}_{-k})$ for all $k = 1, \ldots, K$.

Suppose $\mathbf{y}_k \in \arg\min_{\mathbf{x}_k\in\mathcal{X}_k} f(\mathbf{x}_k, \mathbf{y}_{-k})$ for all $k = 1, \ldots, K$. The first-order optimality conditions yields

$$\nabla_k f(\mathbf{y})^T(\mathbf{x}_k - \mathbf{y}_k) \geq 0, \forall \mathbf{x}_k \in \mathcal{X}_k.$$

Adding the above inequality for all $k = 1, \ldots, K$ yields

$$\nabla f(\mathbf{y})^T(\mathbf{x}-\mathbf{y}) \geq 0, \forall \mathbf{x} \in \mathcal{X}.$$

Therefore, $\mathbf{y}$ is a stationary point of (4).

(ii) We prove that if $\mathbf{y}$ is not a stationary point of (4), then $\nabla f(\mathbf{y})^T(\mathbb{B}\mathbf{y} - \mathbf{y}) < 0$.

It follows from the optimality of $\mathbb{B}_k\mathbf{x}$ that

$$f(\mathbb{B}_k\mathbf{y}, \mathbf{y}_{-k}) \leq f(\mathbf{y}_k, \mathbf{y}_{-k}),$$

and

$$\nabla_k f(\mathbb{B}_k\mathbf{y}, \mathbf{y}_{-k})^T(\mathbf{x}_k - \mathbb{B}_k\mathbf{y}) \geq 0, \forall \mathbf{x}_k \in \mathcal{X}_k. \quad (55)$$

Firstly, there must exist an index j such that

$$f(\mathbb{B}_j\mathbf{y}, \mathbf{y}_{-j}) < f(\mathbf{y}_j, \mathbf{y}_{-j}), \quad (56)$$

otherwise $\mathbf{y}$ would be a stationary point of (4). Since $f(\mathbf{x})$ is pseudo-convex in $\mathbf{x}_k$ for $k = 1, \ldots, K$, it follows from (56) that

$$\nabla_j f(\mathbf{y})^T (\mathbb{B}_j \mathbf{y} - \mathbf{y}_j) < 0. \quad (57)$$

Secondly, for any index k such that $f(\mathbb{B}_k \mathbf{y}, \mathbf{y}_{-k}) = f(\mathbf{y}_k, \mathbf{y}_{-k})$, $\mathbf{y}_k$ minimizes $f(\mathbf{x}_k, \mathbf{y}_{-k})$ over $\mathbf{x}_k \in \mathcal{X}_k$ and $\nabla_k f(\mathbf{y}_k, \mathbf{y}_{-k})^T (\mathbf{x}_k - \mathbf{y}_k) \geq 0$ for any $\mathbf{x}_k \in \mathcal{X}$. Setting $\mathbf{x}_k = \mathbb{B}_k \mathbf{y}$ yields

$$\nabla_k f(\mathbf{y}_k, \mathbf{y}_{-k})^T (\mathbb{B}_k \mathbf{y} - \mathbf{y}_k) \geq 0. \quad (58)$$

Similarly, setting $\mathbf{x}_k = \mathbf{y}_k$ in (55) yields

$$\nabla_k f(\mathbb{B}_k \mathbf{y}, \mathbf{y}_{-k})^T (\mathbf{y}_k - \mathbb{B}_k \mathbf{y}) \geq 0. \quad (59)$$

Adding (58) and (59), we can infer that $(\nabla_k f(\mathbf{y}) - \nabla_k f(\mathbb{B}_k \mathbf{y}, \mathbf{y}_{-k}))^T (\mathbf{y}_k - \mathbb{B}_k \mathbf{y}) \geq 0$. Therefore, we can rewrite (59) as follows

$$\begin{aligned} 0 &\leq \nabla_k f(\mathbb{B}_k \mathbf{y}, \mathbf{y}_{-k})^T (\mathbf{y}_k - \mathbb{B}_k \mathbf{y}) \\ &= (\nabla_k f(\mathbb{B}_k \mathbf{y}, \mathbf{y}_{-k}) - \nabla_k f(\mathbf{y}) + \nabla_k f(\mathbf{y}))^T (\mathbf{y}_k - \mathbb{B}_k \mathbf{y}), \end{aligned}$$

and thus

$$\begin{aligned} &\nabla_k f(\mathbf{y})^T (\mathbb{B}_k \mathbf{y} - \mathbf{y}_k) \leq \\ &-(\nabla_k f(\mathbb{B}_k \mathbf{y}, \mathbf{y}_{-k}) - \nabla_k f(\mathbf{y}))^T (\mathbb{B}_k \mathbf{y} - \mathbf{y}_k) \leq 0. \end{aligned} \quad (60)$$

Adding (57) and (60) over all $k = 1, \ldots, K$ yields

$$\nabla f(\mathbf{y})^T (\mathbb{B}\mathbf{y} - \mathbf{y}) = \textstyle\sum_{k=1}^K \nabla_k f(\mathbf{y})^T (\mathbb{B}_k \mathbf{y} - \mathbf{y}_k) < 0.$$

That is, $\mathbb{B}\mathbf{y} - \mathbf{y}$ is a descent direction of $f(\mathbf{x})$ at the point $\mathbf{y}$.

The proof of Theorem 2 can then be used verbatim to prove the convergence of the algorithm with the approximate problem (28) and the exact/successive line search. ■

REFERENCES

[1] W. Yu, W. Rhee, S. Boyd, and J. Cioffi, "Iterative Water-Filling for Gaussian Vector Multiple-Access Channels," *IEEE Transactions on Information Theory*, vol. 50, no. 1, pp. 145–152, Jan. 2004.

[2] N. Jindal, W. Rhee, S. Vishwanath, S. Jafar, and A. Goldsmith, "Sum Power Iterative Water-Filling for Multi-Antenna Gaussian Broadcast Channels," *IEEE Transactions on Information Theory*, vol. 51, no. 4, pp. 1570–1580, Apr. 2005.

[3] W. Yu, "Sum-capacity computation for the Gaussian vector broadcast channel via dual decomposition," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 754–759, Feb. 2006.

[4] S. Ye and R. Blum, "Optimized signaling for MIMO interference systems with feedback," *IEEE Transactions on Signal Processing*, vol. 51, no. 11, pp. 2839–2848, Nov. 2003.

[5] Z.-Q. Luo and S. Zhang, "Dynamic Spectrum Management: Complexity and Duality," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 1, pp. 57–73, Feb. 2008.

[6] 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.

[7] Q. Shi, M. Razaviyayn, Z.-Q. Luo, and C. He, "An Iteratively Weighted MMSE Approach to Distributed Sum-Utility Maximization for a MIMO Interfering Broadcast Channel," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4331–4340, Sep. 2011.

[8] G. Scutari, F. Facchinei, P. Song, D. P. Palomar, and J.-S. Pang, "Decomposition by Partial Linearization: Parallel Optimization of Multi-Agent Systems," *IEEE Transactions on Signal Processing*, vol. 62, no. 3, pp. 641–656, Feb. 2014.

[9] Y. Yang, G. Scutari, P. Song, and D. P. Palomar, "Robust MIMO Cognitive Radio Systems Under Interference Temperature Constraints," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 11, pp. 2465–2482, Nov. 2013.

[10] Y. Yang, F. Rubio, G. Scutari, and D. P. Palomar, "Multi-Portfolio Optimization: A Potential Game Approach," *IEEE Transactions on Signal Processing*, vol. 61, no. 22, pp. 5590–5602, Nov. 2013.

[11] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An Interior-Point Method for Large-Scale l_1-Regularized Least Squares," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 606–617, Dec. 2007.

[12] 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*, vol. 3, no. 1, 2010.

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

[14] 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, Jun. 2013.

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

[16] P. He, "Correction of Convergence Proof for Iterative Water-Filling in Gaussian MIMO Broadcast Channels," *IEEE Transactions on Information Theory*, vol. 57, no. 4, pp. 2539–2543, Apr. 2011.

[17] 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.

[18] G. Scutari, F. Facchinei, L. Lampariello, , and P. Song, "Distributed Methods for Constrained Nonconvex Multi-Agent Optimization-Part I: Theory," Oct. 2014, submitted to *IEEE Transactions on Signal Processing*. [Online]. Available: http://arxiv.org/abs/1410.4754

[19] R. Horst and N. Thoai, "DC programming: overview," *Journal of Optimization Theory and Applications*, vol. 103, no. 1, pp. 1–43, 1999.

[20] J. M. Ortega and W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*. Academic, New York, 1970.

[21] O. L. Mangasarian, *Nonlinear programming*. McGraw-Hill, 1969.

[22] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge Univ Pr, 2004.

[23] J. F. Sturm, "Using SeDuMi 1.02: A Matlab toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11, no. 1-4, pp. 625–653, Jan. 1999.

[24] S. M. Robinson and R. H. Day, "A sufficient condition for continuity of optimal sets in mathematical programming," *Journal of Mathematical Analysis and Applications*, vol. 45, no. 2, pp. 506–511, Feb. 1974.

[25] 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.

[26] R. H. Byrd, J. Nocedal, and F. Oztoprak, "An Inexact Successive Quadratic Approximation Method for Convex L-1 Regularized Optimization," Sep. 2013. [Online]. Available: http://arxiv.org/abs/1309.3529

[27] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.

[28] W. Yu and J. M. Cioffi, "Sum Capacity of Gaussian Vector Broadcast Channels," *IEEE Transactions on Information Theory*, vol. 50, no. 9, pp. 1875–1892, Sep. 2004.

[29] M. Grant and S. Boyd, "CVX: Matlab Software for Disciplined Convex Programming, version 2.0 beta," http://cvxr.com/, 2012.

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

[31] 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.

[32] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani, "Pathwise coordinate optimization," *The Annals of Applied Statistics*, vol. 1, no. 2, pp. 302–332, Dec. 2007.

[33] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, "Gradient Projection for Sparse Reconstruction: Application to Compressed Sensing and Other Inverse Problems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 586–597, Dec. 2007.

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

[35] Z. Peng, M. Yan, and W. Yin, "Parallel and distributed sparse optimization," *2013 Asilomar Conference on Signals, Systems and Computers*, pp. 659–646, Nov. 2013.

[36] 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.