A Unified Successive Pseudo-Convex Approximation Framework

Yang Yang and Marius Pesavento

***Abstract*—In this paper, we propose a successive pseudo-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, energy efficiency maximization in massive MIMO systems and LASSO in sparse signal recovery.**

***Index Terms*—Energy efficiency, exact line search, LASSO, massive MIMO, MIMO broadcast channel, nonconvex optimization, nondifferentiable optimization, pseudo-convex optimization, successive convex approximation.**

I. Introduction

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

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

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.

Manuscript received April 26, 2016; revised November 13, 2016 and February 23, 2016; accepted February 23, 2017. Date of publication May 02, 2016; date of current version August 05, 2016. The work of Y. Yang was supported by the Seventh Framework Programme for Research of the European Commission under Grant ADEL-619647, the EXPRESS project within the DFG priority program CoSIP (DFG-SPP 1798), and the framework of the Horizon 2020 project FANTASTIC-5G (ICT-671660). The work of M. Pesavento was supported by the Seventh Framework Programme for Research of the European Commission under Grant ADEL 619647 and the EXPRESS project within the DFG priority program CoSIP (DFG-SPP 1798). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Anthony So.

Y. Yang is with Intel Deutschland GmbH, Neubiberg 85579, Germany (email: yang1.yang@intel.com).

M. Pesavento is with the Communication Systems Group, Technische Universität Darmstadt , Darmstadt 64283, Germany (email: pesavento@nt.tu-darmstadt.de).

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

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

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

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

Problems (1) and (3) are of the same form as the objective functions are differentiable and the constraint sets are convex.

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 for the differentiable problem (1) [13] and the proximal gradient method for the nondifferentiable problem (2) [14, 15], 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], and it 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

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

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, 16].

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 $f(\mathbf{x}_1, \ldots, \mathbf{x}_K)$ [17], 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, 18].

A recent progress in parallel algorithms has been made in [8, 9, 16, 19, 20], 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, 16, 19, 20] 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, 19, 20] 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, 16, 19, 20] suffer from two limitations. Firstly, the approximate function must be strictly or 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 (except that in [16]) require the use of a decreasing stepsize. On the one hand, a slow decay of the stepsize is preferable to make notable progress and to achieve satisfactory convergence speed; on the other hand, theoretical convergence is guaranteed only when the stepsize decays fast enough. In practice, it is a difficult task on its own to find a decay rate for the stepsize that provides a good trade-off between convergence speed and convergence guarantee, and current practices mainly rely on heuristics [19].

The contribution of this paper consists in the development of a novel iterative pseudo-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, 8, 9, 19], but also opens new possibilities for constructing approximate problems that are easier to solve. For example, in the MIMO BC sum-rate maximization problems, 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, 18] and decreasing stepsizes [8, 19] are used. For example, the exact line search based on the bisection method can be used when $f(\mathbf{x})$ is convex. 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 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 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-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: the sum rate maximization problem of MIMO BC, the energy efficiency maximization of a massive MIMO system to illustrate the advantage of the proposed approximate function, and the LASSO problem to illustrate the advantage of the proposed stepsize. The paper is concluded in Sec. V.

Notation: We use x, $\mathbf{x}$ and $\mathbf{X}$ to denote a 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 $\mathbf{x}$ where $\mathbf{x} = (x_k)_{k=1}^K$, and $\mathbf{x}_{-k}$ denotes all elements of $\mathbf{x}$ except x_k: $\mathbf{x}_{-k} = (x_j)_{j=1,j\neq k}^K$. 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 $[\mathbf{a}, \mathbf{b}]$: $[\mathbf{x}]_{\mathbf{a}}^{\mathbf{b}} \triangleq \max(\min(\mathbf{x}, \mathbf{b}), \mathbf{a})$. 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 a vector with all elements 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 $\mathbf{y} \in \mathcal{X}$ is a stationary point of problem (1) if

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

Condition (5) is the necessary condition for local 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 [21, 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., $f(\mathbf{x}^{t+1}) < f(\mathbf{x}^t)$ for arbitrary functions $f(\mathbf{x})$ 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 $\alpha \in [0,1]$:

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

It is strictly quasi-convex if the above inequality is satisfied with strict inequality whenever $h(\mathbf{x}) \neq h(\mathbf{y})$.

Pseudo-convex function. A function $h(\mathbf{x})$ is pseudo-convex if [22, Sec. 9.3]

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

Combining this with (5) implies that any stationary point of a pseudo-convex function is a global minimizer [22, Th. 9.3.3].

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

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

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 [22, Th. 9.3.5].

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

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

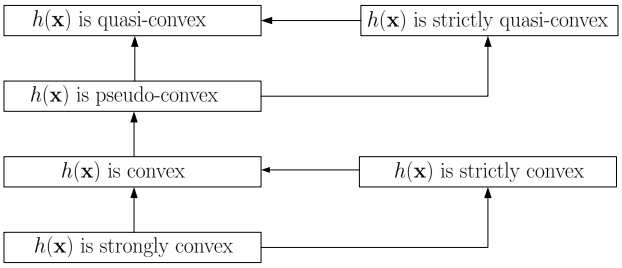

Figure 1. Relationship of functions with different degree of convexity

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 for some positive constant a:

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

The relationship of functions with different degree of convexity is summarized in Fig. 1 where the arrow denotes implication in the direction of the arrow.

III. The Proposed Successive Pseudo-Convex Approximation Algorithm

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 might even exhibit closed-form solutions.

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

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

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)**.** *Provided that Assumptions (A1)-(A3) are satisfied: (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 $\mathbb{B}\mathbf{x}^t - \mathbf{x}^t$. In other words, γ^t is the (globally) optimal point of the following optimization problem:

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

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 $[0, 1]$:

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

If $0 \le \gamma^t = \gamma^\star \le 1$, 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-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 [23]. 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:

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

Algorithm 1 The successive pseudo-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 $t \leftarrow t + 1$.

Note that the existence of a finite m_t satisfying (14) is always guaranteed if $\mathbb{B}\mathbf{x}^t - \mathbf{x}^t$ is a descent direction at $\mathbf{x}^t$ and $\nabla f(\mathbf{x}^t)^T(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t) < 0$ [13].

The successive line search (14) ensures that the stepsize $\gamma^t = \beta^{m_t}$ is sufficiently small to yield a decrease in the objective function, but not too small so that sufficient decrease can be achieved. In this procedure, $\nabla f(\mathbf{x}^t)$ must be calculated and the function $f(\mathbf{x}^t + \beta^m(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t))$ are evaluated several times; see [24, Sec. 3.1] for more details.

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. 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 [25]. 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 $-1 \le x \le 1$ and the point $x^t = 0$ in iteration t. Choosing the approximate function $\tilde{f}(x; x^t) = x^3$, which is strictly 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 in the sense that inequality (11) in Proposition 1 is violated.

Similar to convex optimization problems, any stationary point of the pseudo-convex optimization problem (8) is also globally optimal [22, Th. 9.3.3]. Some important instances of pseudo-convex optimization problems can be solved by interior point methods. For example, the fractional function (convex function divided by a concave function) is one of the most popular examples of pseudo-convex functions. This particular pseudo-convex optimization problem can be solved by the interior point method proposed in [26], whose complexity is the same as that of interior point methods designed for convex optimization problems [23, Ch. 11]. More generally, a pseudo-convex function is also a quasi-convex function, so problem (8) can be solved by the bisection method, solving a convex feasibility problem at each step, relying on standard convex optimization techniques [23, Sec. 4.2.5].

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:

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

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 second inequality comes from the definition of successive line search [cf. (14)] and 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 this subsection, we propose a successive convex approximation framework for the nondifferentiable problem (2), following the previous line of argument. To ease the readers, the development is divided into three steps: we start by applying the proposed Algorithm 1 to problem (3), an equivalent reformulation of (2), and then the update in each iteration is further refined, and we finally remove the auxiliary variables.

Step 1: 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 (A2)-(A3) as well as the following convexity assumption (A1') which is stricter than Assumption (A1):

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

Then the approximation of problem (3) around $(\mathbf{x}^t, y^t)$ is

$$(\mathbb{B}\mathbf{x}^t, y^\star(\mathbf{x}^t)) \triangleq \underset{(\mathbf{x},y):\mathbf{x}\in\mathcal{X}, g(\mathbf{x})\leq y}{\arg\min} \tilde{f}(\mathbf{x};\mathbf{x}^t) + y. \tag{18}$$

That is, we only need to replace the differentiable function $f(\mathbf{x})$ by its (convex) approximate function $\tilde{f}(\mathbf{x};\mathbf{x}^t)$. Since the approximate function in (18) is convex, it is sufficient to verify Assumption (A3) only:

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

Based on the exact line search, the stepsize γ^t in this case is given as

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

where $y^t \geq g(\mathbf{x}^t)$. 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.

Step 2: The point y^{t+1} given in (20b) can be further refined:

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

where the first and the second inequality comes from the fact that $y^t \geq g(\mathbf{x}^t)$ as well as $y^\star(\mathbf{x}^t) = g(\mathbb{B}\mathbf{x}^t)$ and Jensen's inequality of convex functions $g(\mathbf{x})$ [23], respectively. Since $y^{t+1} \geq g(\mathbf{x}^{t+1})$, 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:

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

while $y^0 = g(\mathbf{x}^0)$. 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).

Step 3: The notation in (18)-(19) can be simplified by removing the auxiliary variable $\mathbf{y}$: $\mathbb{B}\mathbf{x}^t$ in (18) can be equivalently written as

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

and combining (19) and (21) yields

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

In 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:

Algorithm 2 The successive 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 $t \leftarrow t + 1$.

$$f(\mathbf{x}^t + \beta^m(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) - f(\mathbf{x}^t) \leq \beta^m\big(\alpha\nabla f(\mathbf{x}^t)^T(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t) + (\alpha - 1)(g(\mathbb{B}\mathbf{x}^t) - g(\mathbf{x}^t))\big). \tag{24}$$

Based on the above steps, the proposed algorithm for the nondifferentiable problem (2) is 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) [27, Rule E], i.e.,

$$\min_{0\leq\gamma\leq 1} f(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) + g(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)).$$

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-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 its computational complexity and signaling exchange (when implemented in a distributed manner) is thus lower than state-of-the-art techniques (for example [27, Rule A'], [28, Equations (9)-(10)], [19, Remark 4] and [29, 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 and New Algorithms

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. We also show that the convergence conditions of existing algorithms can be greatly relaxed by exploiting the proposed framework.

Conditional gradient method: In this iterative algorithm for problem (1), 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 iterative algorithm for problem (1), $\mathbb{B}\mathbf{x}^t$ 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:

$$\tilde{f}(\mathbf{x}; \mathbf{x}^t) = \nabla f(\mathbf{x}^t)^T(\mathbf{x} - \mathbf{x}^t) + \frac{1}{2s^t}\left\|\mathbf{x} - \mathbf{x}^t\right\|_2^2, \tag{26}$$

which is the first-order approximation of $f(\mathbf{x})$ augmented by a quadratic regularization term that is introduced to improve the numerical stability [17]. A generalization of (26) is to replace the quadratic term by $(\mathbf{x}-\mathbf{x}^t)\mathbf{H}^t(\mathbf{x}-\mathbf{x}^t)$ where $\mathbf{H}^t \succ \mathbf{0}$ [28].

Proximal gradient method: If $f(\mathbf{x})$ is convex and has a Lipschitz continuous gradient with a constant L, the proximal gradient method for problem (2) has the following form [14, Sec. 4.2]:

$$\begin{aligned}\mathbf{x}^{t+1} &= \arg\min_{\mathbf{x}} \left\{s^t g(\mathbf{x}) + \frac{1}{2}\left\|\mathbf{x} - (\mathbf{x}^t - s^t\nabla f(\mathbf{x}^t))\right\|^2\right\} \\ &= \arg\min_{\mathbf{x}} \left\{\nabla f(\mathbf{x}^t)(\mathbf{x} - \mathbf{x}^t) + \frac{1}{2s^t}\left\|\mathbf{x} - \mathbf{x}^t\right\|^2 + g(\mathbf{x})\right\},\end{aligned} \tag{27}$$

where $s^t > 0$. In the context of the proposed framework (22), the update (27) is equivalent to defining $\tilde{f}(\mathbf{x}; \mathbf{x}^t)$ as follows:

$$\tilde{f}(\mathbf{x}; \mathbf{x}^t) = \nabla f(\mathbf{x})^T(\mathbf{x} - \mathbf{x}^t) + \frac{1}{2s^t}\left\|\mathbf{x} - \mathbf{x}^t\right\|^2 \tag{28}$$

and setting the stepsize $\gamma^t = 1$ for all t. According to Theorem 2 and the discussion following Assumption (A6), the proposed algorithm converges under a constant unit stepsize if $\tilde{f}(\mathbf{x}; \mathbf{x}^t)$ is a global upper bound of $f(\mathbf{x})$, which is indeed the case when $s^t \leq 1/L$ in view of the descent lemma [13, Prop. A.24].

Jacobi algorithm: In problem (1), 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]

$$\tilde{f}(\mathbf{x}; \mathbf{x}^t) = \textstyle\sum_{k=1}^K \left(f(\mathbf{x}_k, \mathbf{x}^t_{-k}) + \frac{\tau_k}{2}\left\|\mathbf{x}_k - \mathbf{x}_k^t\right\|_2^2\right), \tag{29}$$

where $\tau_k \geq 0$ for $k = 1, \ldots, K$. The k-th component function $f(\mathbf{x}_k, \mathbf{x}^t_{-k}) + \frac{\tau_k}{2}\left\|\mathbf{x}_k - \mathbf{x}_k^t\right\|_2^2$ in (29) is obtained from the original function $f(\mathbf{x})$ by fixing all variables except $\mathbf{x}_k$, i.e., $\mathbf{x}_{-k} = \mathbf{x}^t_{-k}$, and further adding a quadratic regularization term. Since $\tilde{f}(\mathbf{x}; \mathbf{x}^t)$ in (29) is convex, Assumption (A1) is satisfied. Based on the observations that

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

we conclude that Assumption (A3) is satisfied by the choice of the approximate function in (29). The resulting approximate problem is given by

$$\begin{array}{ll}\underset{\mathbf{x}=(\mathbf{x}_k)_{k=1}^K}{\text{minimize}} & \sum_{k=1}^K (f(\mathbf{x}_k, \mathbf{x}^t_{-k}) + \frac{\tau_k}{2}\left\|\mathbf{x}_k - \mathbf{x}_k^t\right\|^2) \\ \text{subject to} & \mathbf{x} \in \mathcal{X}.\end{array} \tag{30}$$

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 (31).

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 $t \leftarrow t + 1$.

This is commonly known as the Jacobi algorithm. The structure inside the constraint set $\mathcal{X}$, if any, may be exploited to solve (30) 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 (30) is convex, primal and dual decomposition techniques can readily be used to solve (30) efficiently [30].

To guarantee the convergence, the condition proposed in [9] is that $\tau_k > 0$ for all k in (29) unless $f(\mathbf{x})$ is strongly convex in each $\mathbf{x}_k$. However, the strong convexity of $f(\mathbf{x})$ in each $\mathbf{x}_k$ is a strong assumption that cannot always be satisfied and the additional quadratic regularization term that is required may destroy the convenient structure that could otherwise be exploited, as we show through an example application in the MIMO BC in Sec. IV-A. In the case $\tau_k = 0$, convergence of the Jacobi algorithm (30) is only proved when $f(\mathbf{x})$ is jointly convex in $(\mathbf{x}_1, \ldots, \mathbf{x}_K)$ and the stepsize is inversely proportional to the number of variables K [2, 10, 13], namely, $\gamma^t = 1/K$. However, the resulting convergence speed is usually slow when K is large, as it tends to be conservative.

With the technical assumptions specified in Theorem 2, the existing convergence conditions can be relaxed. In particular, the convergence of the Jacobi algorithm with the approximate problem (30) and successive line search is guaranteed even when $\tau_k = 0$. This is because $\tilde{f}(\mathbf{x}; \mathbf{x}^t)$ in (29) is already convex when $\tau_k = 0$ for all k and it naturally satisfies the pseudo-convexity assumption specified by Assumption (A1).

In the case that the constraint set $\mathcal{X}$ has a Cartesian product structure (4), the subproblem (30) is naturally decomposed into K sub-problems, one for each variable, which are then solved *in parallel*. In this case, the requirement in the convexity of $f(\mathbf{x})$ in each $\mathbf{x}_k$ can even be relaxed to pseudo-convexity only (although the sum function $\sum_{k=1}^K f(\mathbf{x}_k, \mathbf{x}_{-k}^t)$ is not necessarily pseudo-convex in $\mathbf{x}$ as pseudo-convexity is not preserved under nonnegative weighted sum operator), and this leads to the following update: $\mathbb{B}\mathbf{x}^t = (\mathbb{B}_k \mathbf{x}^t)_{k=1}^K$ and

$$\mathbb{B}_k \mathbf{x}^t \in \arg\min_{\mathbf{x}_k \in \mathcal{X}_k} f(\mathbf{x}_k, \mathbf{x}_{-k}^t),\ k = 1, \ldots, K, \qquad (31)$$

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 proposed Jacobi algorithm is formally summarized in Algorithm 3 and its convergence is proved in Theorem 3.

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 proposed convergence condition specified in Theorem 3 relaxes those in [8, 19]: $f(\mathbf{x})$ only needs to be pseudo-convex in each variable $\mathbf{x}_k$ 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. We will show in Sec. IV-B by an example application of the energy efficiency maximization problem in massive MIMO systems how the weak assumption on the approximate function's convexity proposed in Theorem 2-3 can be exploited to the largest extent.

IV. Example Applications

In this section, we illustrate through several important applications in communication networks and signal processing the *new* algorithms motivated by the proposed framework.

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 [31]

$$\begin{aligned} \underset{(\mathbf{Q}_k)_{k=1}^K}{\text{maximize}} \quad & \log\big|\mathbf{I} + \textstyle\sum_{k=1}^K \mathbf{H}_k \mathbf{Q}_k \mathbf{H}_k^H\big| \\ \text{subject to} \quad & \mathbf{Q}_k \succeq \mathbf{0},\ \forall k, \textstyle\sum_{k=1}^K \text{tr}(\mathbf{Q}_k) \leq P, \end{aligned} \qquad (32)$$

where P is the power budget at the base station.

Problem (32) is a convex problem whose solution cannot be expressed in closed-form and can only be found iteratively. To apply Algorithm 1, we invoke (29)-(30) and the approximate problem at the t-th iteration is

$$\begin{aligned} \underset{(\mathbf{Q}_k)_{k=1}^K}{\text{maximize}} \quad & \textstyle\sum_{k=1}^K \log\big|\mathbf{R}_k(\mathbf{Q}_{-k}^t) + \mathbf{H}_k \mathbf{Q}_k \mathbf{H}_k^H\big| \\ \text{subject to} \quad & \mathbf{Q}_k \succeq \mathbf{0},\ \forall k,\ \textstyle\sum_{k=1}^K \text{tr}(\mathbf{Q}_k) \leq P, \end{aligned} \qquad (33)$$

where $\mathbf{R}_k(\mathbf{Q}_{-k}^t) \triangleq \mathbf{I} + \sum_{j \neq k} \mathbf{H}_j \mathbf{Q}_j^t \mathbf{H}_j^H$. 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 (33) is compact, the approximate problem (33) has a solution and Assumptions (A4)-(A5) are satisfied.

Problem (33) is convex and the sum-power constraint coupling $\mathbf{Q}_1, \ldots, \mathbf{Q}_K$ is separable, so dual decomposition techniques can be used [30]. In particular, the constraint set has a nonempty interior, so strong duality holds and (33) can

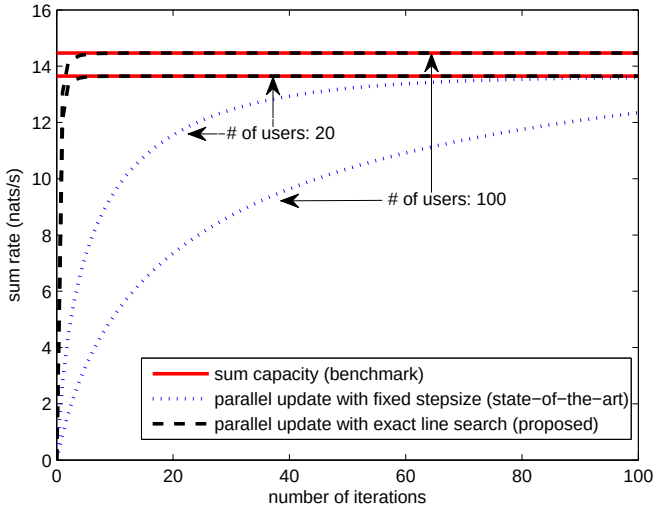

Figure 2. MIMO BC: sum-rate versus the number of iterations.

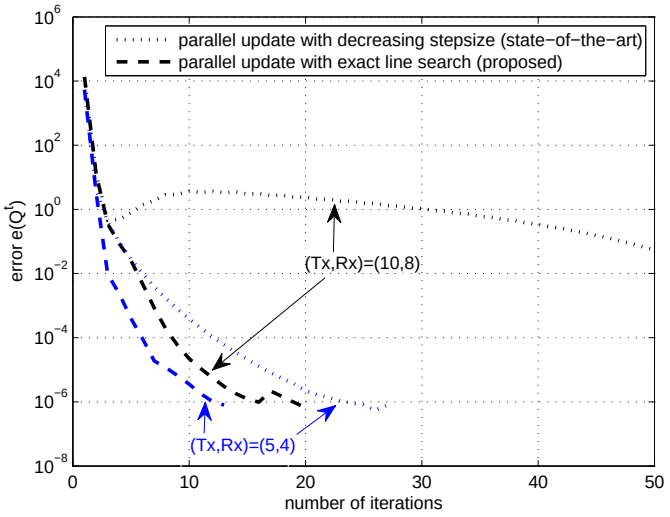

Figure 3. MIMO BC: error $e(\mathbf{Q}^t) = \Re(\mathrm{tr}(\nabla f(\mathbf{Q}^t)(\mathbb{B}\mathbf{Q}^t - \mathbf{Q}^t)))$ versus the number of iterations.

be solved from the dual domain by relaxing the sum-power constraint into the Lagrangian [23]:

$$\mathbb{B}\mathbf{Q}^t = \underset{(\mathbf{Q}_k \succeq \mathbf{0})_{k=1}^K}{\arg\max} \left\{ \begin{array}{l} \sum_{k=1}^K \log \left| \mathbf{R}_k(\mathbf{Q}^t_{-k}) + \mathbf{H}_k \mathbf{Q}_k \mathbf{H}_k^H \right| \\ -\lambda^\star (\sum_{k=1}^K \mathrm{tr}(\mathbf{Q}_k) - P) \end{array} \right\}. \tag{34}$$

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: $\lambda^\star \geq 0$, $\sum_{k=1}^K \mathrm{tr}(\mathbb{B}_k \mathbf{Q}^t) - P \leq 0$, $\lambda^\star (\sum_{k=1}^K \mathrm{tr}(\mathbb{B}_k \mathbf{Q}^t) - P) = 0$, and can be found efficiently using the bisection method.

The problem in (34) 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

$$\mathbb{B}_k \mathbf{Q}^t = \underset{\mathbf{Q}_k \succeq \mathbf{0}}{\arg\max} \left\{ \log \left| \mathbf{R}_k(\mathbf{Q}^t_{-k}) + \mathbf{H}_k \mathbf{Q}_k \mathbf{H}_k^H \right| - \lambda^\star \mathrm{tr}(\mathbf{Q}_k) \right\}, \tag{35}$$

and $\mathbb{B}_k \mathbf{Q}^t$ exhibits a closed-form expression based on the waterfilling solution [2]. Thus problem (33) 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.

We remark that when the channel matrices $\mathbf{H}_k$ are rank deficient, which is the case when $\mathbf{H}_k$ is a fat matrix, e.g., the base station has a large-scale antenna array, problem (33) is convex but not strongly convex, but the proposed algorithm with the approximate problem (33) still converges. However, if the approximate function in [8] is used [cf. (29)], an additional quadratic regularization term must be included into (33) (and thus (35)) to make the approximate problem strongly convex, 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 (33) and stepsize based on the exact line search, and compare it with the iterative algorithm proposed in [2, 18], which uses the same approximate problem (33) 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 which 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 (32) 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 (33) and 0.0018 seconds to perform the exact line search (the software/hardware environment is further specified in Sec. IV-C). 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 algorithm of [20], which uses the approximate problem (33) but with an additional quadratic regularization term, cf. (29), 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 [20] 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 (33), while the approximate problem proposed in [20] does not exhibit a closed-form solution and can only be solved iteratively. Specifically, it takes CVX (version 2.0 [32]) 21.1785 seconds (based on the dual approach (35) where $\lambda^\star$ is found by bisection). Therefore, the overall complexity per iteration of the proposed algorithm is much lower than that of [20].

B. Energy Efficiency Maximization in Massive MIMO Systems

In this subsection, we study the energy efficiency maximization problem in massive MIMO systems to illustrate the advantage of the relaxed convexity requirement of the approximate function in the proposed iterative optimization approach: according to Assumption (A1), $\tilde{f}(\mathbf{x};\mathbf{x}^t)$ only needs to exhibit the pseudo-convexity property rather than the convexity or strong convexity property that is conventionally required.

Consider a multi-cell network with K cells and each cell serves one user. The base stations have a large-scale antenna systems and the user is equipped with a single antenna. The achievable transmission rate for user k in the uplink can be formulated into the following general form[1]:

$$r_k(\mathbf{p}) \triangleq \log\left(1+\frac{w_{kk}p_k}{\sigma_k^2+\phi_k p_k+\sum_{j\neq k} w_{kj}p_j}\right), \quad (36)$$

where p_k is the transmission power for user k, σ_k^2 is the covariance of the additive noise at the base station k, ϕ_k is a positive constant that depends on the channel conditions only, and w_{kj} is the effective channel from user j to base station k, which depends on the real channel condition and the receive beamforming at base station k. In particular, $\phi_k p_k$ accounts for the hardware impairments in the large-scale antenna array, and $\sum_{j\neq k} w_{kj}p_j$ accounts for the inter-user interference [34].

In 5G wireless communication networks, the energy efficiency is a key performance indicator. To address this issue, we look for the optimal power allocation that maximizes the energy efficiency:

$$\begin{aligned} \underset{\mathbf{p}}{\text{maximize}} \quad & \frac{\sum_{k=1}^K r_k(\mathbf{p})}{P_c+\sum_{k=1}^K p_k} \\ \text{subject to} \quad & \underline{\mathbf{p}} \le \mathbf{p} \le \bar{\mathbf{p}}, \end{aligned} \quad (37)$$

where P_c is a positive constant representing the total circuit power dissipated in the network, and $\underline{\mathbf{p}} = (\underline{p}_k)_{k=1}^K$ ($\overline{\mathbf{p}} = (\overline{p}_k)_{k=1}^K$) specifies the lower (upper) bound constraint.

Problem (37) is nonconvex and NP-hard [5]. Therefore we aim at finding a stationary point of (37) using the proposed framework. To begin with, we propose the following approximate function at $\mathbf{p}=\mathbf{p}^t$:

$$\tilde{f}(\mathbf{p};\mathbf{p}^t)=\frac{\sum_{k=1}^K \tilde{r}_k(p_k;\mathbf{p}^t)}{P_c+\sum_{k=1}^K p_k}, \quad (38)$$

[1]We assume a single resource block. However, the extension to multiple resource blocks is straightforward; see [33] for more details.

where

$$\tilde{r}_k(p_k;\mathbf{p}^t) \triangleq r_k(p_k,\mathbf{p}_{-k}^t)+\sum_{j\neq k}(p_k-p_k^t)\nabla_{p_k} r_j(\mathbf{p}^t). \quad (39)$$

Note that the approximate function $\tilde{f}(\mathbf{p};\mathbf{p}^t)$'s numerator consists of K component functions, one for each variable p_k, and the k-th component function is constructed as follows: since $r_k(\mathbf{p})$ is concave in p_k (shown in the right column of this page) but $r_j(\mathbf{p})$ is not concave in p_k (as a matter of fact, it is convex in p_k), the concave function $r_k(p_k,\mathbf{p}_{-k}^t)$ is preserved in $\tilde{r}_k(p_k;\mathbf{p}^t)$ in (39) with $\mathbf{p}_{-k}$ fixed to be $\mathbf{p}_{-k}^t$ while the nonconcave functions $\{r_j(\mathbf{p})\}$ are linearized w.r.t. p_k at $\mathbf{p}=\mathbf{p}^t$. In this way, the partial concavity in the nonconcave function $\sum_{j=1}^K r_j(\mathbf{p})$ is preserved in $\tilde{f}(\mathbf{p};\mathbf{p}^t)$. Similarly, since $P_c+\sum_{j=1}^K p_j$ in the denominator is linear in $\mathbf{p}$, we leave it unchanged. Note that the division operator in the original problem (37) is kept in the approximate function $\tilde{f}(\mathbf{p};\mathbf{p}^t)$ (38). Although it will destroy the concavity (recall that a concave function divided by a linear function is no longer a concave function), the pseudo-concavity of $\tilde{f}(\mathbf{p};\mathbf{p}^t)$ required in Assumption (A1) is still preserved, because i) the function $r_k(p_k,\mathbf{p}_{-k}^t)$ is concave (this can be verified by checking the second order derivative of $r_k(p_k,\mathbf{p}_{-k}^t)$ w.r.t. p_k; details are presented in [33] and omitted here due to the page limit), and the function $\sum_{k=1}^K \tilde{r}_k(p_k;\mathbf{p}^t)$ is thus concave in $\mathbf{p}$; ii) The nonnegative concave function $\sum_{k=1}^K \tilde{r}_k(p_k;\mathbf{p}^t)$ divided by a positive convex function $P_c+\sum_{k=1}^K p_k$ is pseudo-concave [22].

Assumption (A2) is satisfied because both $\tilde{r}_k(p_k;\mathbf{p}^t)$ and $p_k+P_c+\sum_{j\neq k} p_j^t$ are continuously differentiable for any $\mathbf{p}^t \ge \mathbf{0}$. Then we verify Assumption (A3) in Theorem 3, namely the gradient of the approximate function and that of the original objective function are identical at $\mathbf{p}=\mathbf{p}^t$:

$$\begin{aligned} &\nabla_{p_k}\tilde{f}(\mathbf{p};\mathbf{p}^t)\Big|_{\mathbf{p}=\mathbf{p}^t} \\ &= \frac{\nabla_{p_k}\tilde{r}_k(\mathbf{p}^t)(p_c+\mathbf{1}^T\mathbf{p}^t)-\sum_{j=1}^K \tilde{r}_j(\mathbf{p}^t)}{(P_c+\mathbf{1}^T\mathbf{p}^t)^2} \\ &= \frac{\nabla_{p_k}(\sum_{j=1}^K r_j(\mathbf{p}^t))(p_c+\mathbf{1}^T\mathbf{p}^t)-\sum_{j=1}^K r_j(\mathbf{p}^t)}{(P_c+\mathbf{1}^T\mathbf{p}^t)^2} \\ &= \nabla_{p_k}\left(\frac{\sum_{j=1}^K r_j(\mathbf{p})}{P_c+\sum_{j=1}^K p_j}\right)\Bigg|_{\mathbf{p}=\mathbf{p}^t}, \forall\, k, \end{aligned}$$

where we have used the facts that $\nabla_{p_k}\tilde{r}_k(\mathbf{p}^t) = \nabla_{p_k}(\sum_{j=1}^K r_j(\mathbf{p}^t))$ and $\tilde{r}_k(\mathbf{p}^t)=r_k(\mathbf{p}^t)$ for all k.

Given the approximate function (38), the approximate problem in iteration t is thus

$$\mathbb{B}\mathbf{p}^t = \underset{\underline{\mathbf{p}}\le\mathbf{p}\le\bar{\mathbf{p}}}{\arg\max}\ \frac{\sum_{k=1}^K \tilde{r}_k(p_k;\mathbf{p}^t)}{P_c+\sum_{k=1}^K p_k}. \quad (40)$$

Since the feasible set $[\underline{\mathbf{p}},\overline{\mathbf{p}}]$ is bounded, Assumptions (A4) and (A5) are also satisfied. Since the objective function in (37) is nonconcave, it may not be computationally affordable to perform the exact line search. Instead, the successive line search can be applied to calculate the stepsize. The convergence of the proposed algorithm with approximate problem (40) and successive line search follows from Theorem 3.

$$p_k(\lambda^{t,\tau}) = \left[\mathtt{int}_k(\mathbf{p}^t) \frac{\sqrt{(2\phi_k + w_{kk})^2 - 4\phi_k \left(\frac{w_{kk}}{(\pi_k(\mathbf{p}^t) - \lambda^{t,\tau})\mathtt{int}_k(\mathbf{p}^t)} + 1 \right)} - 1}{2\phi_k(\pi_k(\mathbf{p}^t) - \lambda^{t,\tau})(\phi_k + w_{kk})} \right]_{\underline{p}_k}^{\overline{p}_k}, \tag{43}$$

The numerator function and the denominator function in (40) is concave and linear, respectively, so the optimization problem in (40) is a fractional programming problem and can be solved by the Dinkelbach's algorithm [34, Algorithm 5]: given $\lambda^{t,\tau}$ ($\lambda^{t,0}$ can be set to 0), the following optimization problem in iteration $\tau + 1$ is solved:

$$\mathbf{p}(\lambda^{t,\tau}) \triangleq \underset{\underline{\mathbf{p}} \le \mathbf{p} \le \overline{\mathbf{p}}}{\arg\max} \sum_{k=1}^{K} \tilde{r}_k(p_k; \mathbf{p}^t) - \lambda^{t,\tau} \left(P_c + \sum_{k=1}^{K} p_k \right), \tag{41}$$

The optimization problem in (41) can be decomposed into scalar subproblems that can be solved in parallel:

$$p_k(\lambda^{t,\tau}) = \underset{\underline{p}_k \le p_k \le \bar{p}_k}{\arg\max}\ \tilde{r}_k(p_k; \mathbf{p}^t) - \lambda^{t,\tau} p_k,\ k = 1, \ldots, K, \tag{42a}$$

The variable $\lambda^{t,\tau}$ is then updated in iteration $\tau + 1$ as

$$\lambda^{t,\tau+1} = \frac{\sum_{k=1}^{K} \tilde{r}_k(p_k(\lambda^{t,\tau}); \mathbf{p}^t)}{P_c + \sum_{k=1}^{K} p_k(\lambda^{t,\tau})}. \tag{42b}$$

It follows from the convergence properties of the Dinkelbach's algorithm that

$$\lim_{\tau \to \infty} \mathbf{p}(\lambda^{t,\tau}) = \mathbb{B}\mathbf{p}^t$$

at a superlinear convergence rate. Note that problem (42a) can be solved in closed-form, as $p_k(\lambda^{t,\tau})$ is simply the projection of the point that sets the gradient of the objective function in (42a) to zero onto the interval $[\underline{p}_k, \overline{p}_k]$. It can be verified that finding that point is equivalent to finding the root of a polynomial with order 2 and it thus admits a closed-form expression. We omit the detailed derivations and directly give the expression of $p_k(\lambda^{t,\tau})$ in (43) at the top of this page, where $\pi_k(\mathbf{p}^t) \triangleq \sum_{j \neq k} \nabla_{p_k} r_j(\mathbf{p}^t)$ and $\mathtt{int}_k(\mathbf{p}^t) \triangleq \sigma_k^2 + \sum_{j \neq k} w_{kj} p_j^t$.

We finally remark that the approximate function in (38) is constructed in the same spirit as [8, 9, 35] by keeping as much concavity as possible, namely, $r_k(\mathbf{p})$ in the numerator and $P_c + \sum_{j=1}^{K} p_j$ in the denominator, and linearizing the non-concave functions only, namely, $\sum_{j \neq k} r_j(\mathbf{p})$ in the numerator. Besides this, the division operator is also kept. Therefore, the proposed algorithm is of a best-response nature and expected to converge faster than gradient based algorithms which linearizes the objective function in (37) completely. However, the convergence of the proposed algorithm with the approximate problem (40) cannot be derived from existing works, since the approximate function presents only a weak form of convexity, namely, the pseudo-convexity, which is much weaker than those required in state-of-the-art convergence analysis, e.g., uniform strong convexity [8].

Simulations. The number of antennas at the BS in each cell is $M = 50$, and the channel from user j to cell k is $\mathbf{h}_{kj} \in \mathbb{C}^{M \times 1}$. We assume a similar setup as [34]: $w_{kk} = \left| \mathbf{h}_{kk}^H \mathbf{h}_{kk} \right|^2$, $w_{kj} = \left| \mathbf{h}_{kk}^H \mathbf{h}_{kj} \right|^2 + \epsilon \mathbf{h}_{kk}^H \mathbf{D}_j \mathbf{h}_{kk}$ for $j \neq k$ and $\phi_k = \epsilon \mathbf{h}_{kk}^H \mathbf{D}_k \mathbf{h}_{kk}$, where $\epsilon = 0.01$ is the error magnitude of hardware impairments at the BS and $\mathbf{D}_j = \mathrm{diag}(\{|h_{jj}(m)|^2\}_{m=1}^M)$. The noise covariance $\sigma_k^2 = 1$, and the hardware dissipated power p_c is 10dBm, while $\underline{p}_k$ is -10dBm and $\overline{p}_k$ is 10dBm for all users. The benchmark algorithm is [34, Algorithm 1], which successively maximizes the following lower bound function of the objective function in (37), which is tight at $\mathbf{p} = \mathbf{p}^t$:

$$\begin{aligned} \underset{\mathbf{q}}{\text{maximize}}\ & \frac{\sum_{k=1}^{K} b_k^t + a_k^t \log w_{kk}}{P_c + \sum_{k=1}^{K} e^{q_k}} + \\ & \frac{\sum_{k=1}^{K} a_k^t (q_k - \log(\sigma_k^2 + \phi_k e^{q_k} + \sum_{j \neq k} w_{kj} e^{q_j}))}{P_c + \sum_{k=1}^{K} e^{q_k}} \\ \text{subject to}\ & \log(\underline{p}_k) \le q_k \le \log(\overline{p}_k),\ k = 1, \ldots, K, \end{aligned} \tag{44}$$

where $a_k^t \triangleq \mathrm{sinr}_k(\mathbf{p}^t)/(1 + \mathrm{sinr}_k(\mathbf{p}^t))$, $b_k^t \triangleq \log(1 + \mathrm{sinr}_k(\mathbf{p}^t)) - \log(\mathrm{sinr}_k(\mathbf{p}^t))\mathrm{sinr}_k(\mathbf{p}^t)/(1 + \mathrm{sinr}_k(\mathbf{p}^t))$, and $\mathrm{sinr}_k(\mathbf{p}) \triangleq w_{kk} p^t / (\sigma_k^2 + \phi_k p_k + \sum_{j \neq k} w_{kj} p_j)$. Denote the optimal variable of (44) as $\mathbf{q}^t$ (which can be found by the Dinkelbach's algorithm); then the variable $\mathbf{p}$ is updated as $p_k^{t+1} = e^{q_k^t}$ for all $k = 1, \ldots, K$. We thus coin [34, Algorithm 1] as the successive lower bound maximization (SLBM) method.

In Fig. 4, we compare the convergence behavior of the proposed Successive Pseudo-Convex Approximation Algorithm (SPCAA) and the SLBM method in terms of both the number of iterations (the upper subplots) and the CPU time (the lower subplots), for two different number of users: $K = 10$ in the left column and $K = 50$ in the right column. It is obvious that the convergence speed of the proposed algorithm in terms of the number of iterations is comparable to that of the SLBM method. However, we remark that the approximate problem (40) of the proposed algorithm is superior to that of the SLBM method in the following aspects:

Firstly, the approximate problem of the proposed algorithm consists of independent subproblems that can be solved in parallel, cf. (41), while each subproblem has a closed-form solution, cf. (42)-(43). However, the optimization variable in the approximate problem of the SLBM method (44) is a vector $\mathbf{q} \in \mathbb{R}^{K \times 1}$ and the approximate problem can only be solved by a general purpose solver.

In the simulations, we use the Matlab optimization toolbox to solve (44) and the iterative update specified in (42)-(43) to solve (40), where the stopping criterion for (42) is $\left\| \boldsymbol{\lambda}^{t,\tau} \right\|_\infty \le 10^{-5}$. The upper subplots in Fig. 4 show that the numbers of iterations required for convergence are approximately the same for the SLBM method when $K = 10$ and when $K = 50$. However, we see from the lower subplots

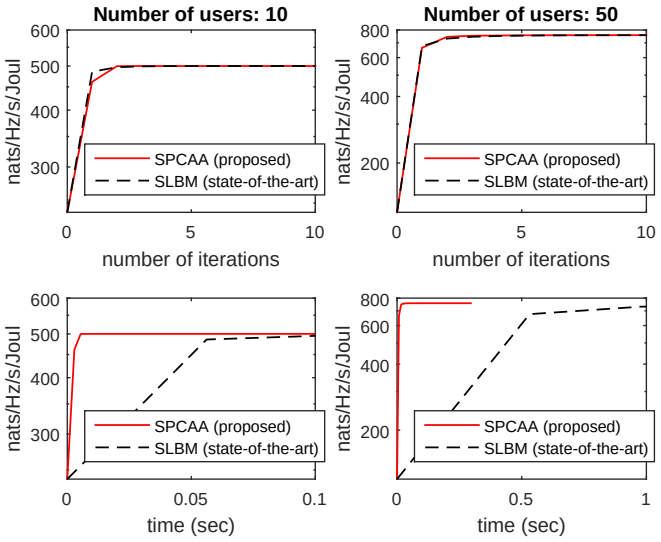

Figure 4. Energy Efficiency Maximization: achieved energy efficiency versus the number of iterations and CPU time

in Fig. 4 that the CPU time of each iteration of the SLBM method is dramatically increased when K is increased from 10 to 50. On the other hand, the CPU time of the proposed algorithm is not notably changed because the operations are parallelizable[2] and the required CPU time is thus not affected by the problem size.

Secondly, since a variable substitution $q_k = e^{p_k}$ is adopted in the SLBM method (we refer to [34] for more details), the lower bound constraint $\underline{p}_k = 0$ (which corresponds to $q_k = -\infty$) cannot be handled by the SLBM method numerically which may impair the applicability of the SLBM method.

C. LASSO

In this 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, 36, 37]:

$$\underset{\mathbf{x}}{\text{minimize}} \quad \tfrac{1}{2}\left\|\mathbf{A}\mathbf{x}-\mathbf{b}\right\|_2^2 + \mu\left\|\mathbf{x}\right\|_1, \tag{45}$$

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

$$f(\mathbf{x}) \triangleq \tfrac{1}{2}\left\|\mathbf{A}\mathbf{x}-\mathbf{b}\right\|_2^2, \quad \text{and} \quad g(\mathbf{x}) \triangleq \mu\left\|\mathbf{x}\right\|_1. \tag{46}$$

Problem (45) 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 (29), the approximate problem is

$$\mathbb{B}\mathbf{x}^t = \arg\min_{\mathbf{x}} \left\{\textstyle\sum_{k=1}^K f(x_k, \mathbf{x}_{-k}^t) + g(\mathbf{x})\right\}. \tag{47}$$

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 (47) reduces to K independent scalar subproblems 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 [37] and

$$\mathbf{r}(\mathbf{x}) \triangleq \mathbf{d}(\mathbf{A}^T\mathbf{A}) \circ \mathbf{x} - \mathbf{A}^T(\mathbf{A}\mathbf{x}-\mathbf{b}), \tag{48}$$

or more compactly:

$$\mathbb{B}\mathbf{x}^t = (\mathbb{B}_k\mathbf{x}^t)_{k=1}^K = \mathbf{d}(\mathbf{A}^T\mathbf{A})^{-1} \circ \mathcal{S}_{\mu\mathbf{1}}(\mathbf{r}(\mathbf{x}^t)). \tag{49}$$

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

$$\begin{aligned}\gamma^t &= \arg\min_{0\le\gamma\le 1} \left\{f(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) + \gamma\big(g(\mathbb{B}\mathbf{x}^t) - g(\mathbf{x}^t)\big)\right\} \\ &= \arg\min_{0\le\gamma\le 1} \left\{\begin{array}{c}\frac{1}{2}\left\|\mathbf{A}(\mathbf{x}^t + \gamma(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) - \mathbf{b}\right\|_2^2 \\ +\gamma\,\mu\big(\left\|\mathbb{B}\mathbf{x}^t\right\|_1 - \left\|\mathbf{x}^t\right\|_1\big)\end{array}\right\} \\ &= \left[-\frac{(\mathbf{A}\mathbf{x}^t-\mathbf{b})^T\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t) + \mu(\left\|\mathbb{B}\mathbf{x}^t\right\|_1 - \left\|\mathbf{x}^t\right\|_1)}{(\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))^T(\mathbf{A}(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))}\right]_0^1.\end{aligned} \tag{50}$$

The exact line search consists in solving a convex quadratic optimization problem with a scalar variable and a bound constraint, so the problem exhibits a closed-form solution (50). Therefore, both the update direction and stepsize can be calculated in closed-form. We name the proposed update (49)-(50) as Soft-Thresholding with Exact Line search Algorithm (STELA).

Since $f(x_k, \mathbf{x}_{-k}^t)$ is strongly convex, Assumptions (A4) and (A5) are satisfied and the proposed update (49)-(50) converges. The proposed update 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 (49). This is in the same spirit as [19, 38] and the convergence speed is generally faster than BCD [39] or the gradient-based update [40]. Secondly, the proposed exact line search (50) 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 procedures proposed in [19, 28, 38] where either decreasing stepsizes are used [19] or the line search is over the original nondifferentiable objective function in (45) [28, 38]:

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

Computational complexity. The computational overhead associated with the proposed exact line search (50) can

[2]By stacking the $p_k(\lambda^{t,\tau})$'s into the vector form $\mathbf{p}(\lambda^{t,\tau}) = (p_k(\lambda^{t,\tau}))_{k=1}^K$ we can see that only element wise operations between vectors and matrix vector multiplications are involved. The simulations on which Fig. 4 are based are not performed in a real parallel computing environment with K processors, but only make use of the efficient linear algebraic implementations available in Matlab which already implicitly admits a certain level of parallelism.

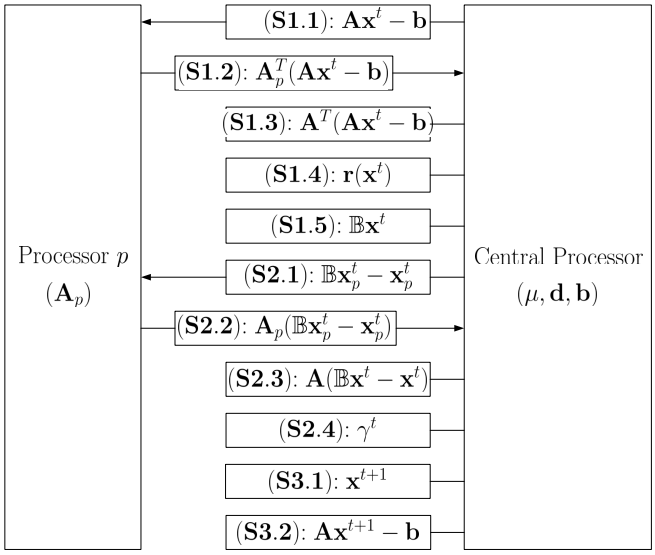

Figure 5. Operation flow and signaling exchange between local processor p and the central processor. A solid line indicates the computation that is locally performed by the central/local processor, and a solid line with an arrow indicates signaling exchange between the central and local processor and the direction of the signaling exchange.

significantly be reduced if (50) is carefully implemented as outlined in the following. The most complex operation in (50) 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 is computed in order to determine the best-response in (49). 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:

$$\begin{aligned}\mathbf{A}\mathbf{x}^{t+1} - \mathbf{b} &= \mathbf{A}(\mathbf{x}^t + \gamma^t(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t)) - \mathbf{b} \\ &= (\mathbf{A}\mathbf{x}^t - \mathbf{b}) + \gamma^t\mathbf{A}(\mathbb{B}\mathbf{x}^t - \mathbf{x}^t), \qquad (51)\end{aligned}$$

where only an additional vector addition is involved. As a result, the stepsize (50) does not incur any additional matrix-vector multiplications, but only affordable vector-vector multiplications.

Signaling exchange. When $\mathbf{A}$ cannot 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:

$$\mathbf{A}\mathbf{x} = \textstyle\sum_{p=1}^P \mathbf{A}_p\mathbf{x}_p, \qquad (52a)$$

$$\mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) = \left(\mathbf{A}_p^T(\mathbf{A}\mathbf{x} - \mathbf{b})\right)_{p=1}^P, \qquad (52b)$$

$$\mathbf{d}(\mathbf{A}^T\mathbf{A}) = (\mathbf{d}(\mathbf{A}_p^T\mathbf{A}_p))_{p=1}^P. \qquad (52c)$$

where $\mathbf{x}_p \in \mathbb{R}^{K_p}$. The central processor computes the best-response $\mathbb{B}\mathbf{x}^t$ in (49) and the stepsize γ^t in (50). The decomposition in (52) enables us to analyze the signaling exchange between local processor p and the central processor involved in (49) and (50)[3].

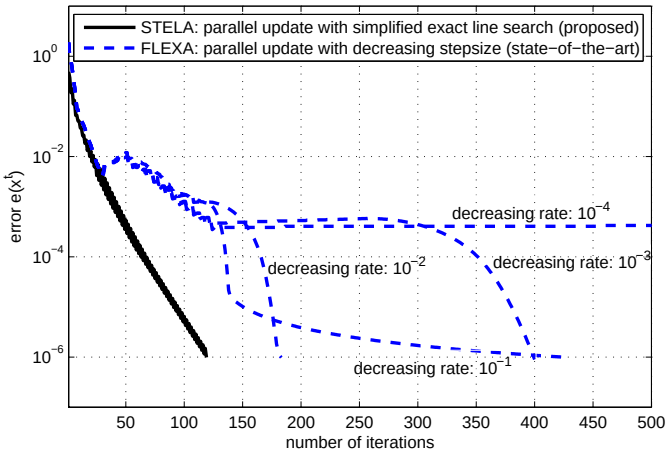

Figure 6. Convergence of STELA (proposed) and FLEXA (state-of-the-art) for LASSO: error versus the number of iterations.

The signaling exchange is summarized in Fig. 5. Firstly, the central processor sends $\mathbf{A}\mathbf{x}^t - \mathbf{b}$ to the local processors (**S1.1**)[4], and each local processor p for $p = 1, \ldots, P$ 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 (52b) and calculates $\mathbf{r}(\mathbf{x}^t)$ as in (48) (**S1.4**) and then $\mathbb{B}\mathbf{x}^t$ as in (49) (**S1.5**). Then the central processor sends $\mathbb{B}\mathbf{x}_p^t - \mathbf{x}_p^t$ to the local processor p for $p = 1, \ldots, P$ (**S2.1**), and each local 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 (52a), calculates γ^t as in (50) (**S2.4**), and updates $\mathbf{x}^{t+1}$ (**S3.1**) and $\mathbf{A}\mathbf{x}^{t+1} - \mathbf{b}$ (**S3.2**) according to (51). From Fig. 5 we observe that the exact line search (50) 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. (51).

We finally remark that the proposed successive line search can also 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. 6 the proposed algorithm STELA with FLEXA [19] in terms of the error criterion $e(\mathbf{x}^t)$ defined as:

$$e(\mathbf{x}^t) \triangleq \left\|\nabla f(\mathbf{x}^t) - \left[\nabla f(\mathbf{x}^t) - \mathbf{x}^t\right]_{-\mu\mathbf{1}}^{\mu\mathbf{1}}\right\|_2. \qquad (53)$$

Note that $\mathbf{x}^\star$ is a solution of (45) if and only if $e(\mathbf{x}^\star) = 0$ [29]. FLEXA is implemented as outlined in [19]; however, the selective update scheme [19] 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 $\gamma^{t+1} = \gamma^t(1 - \min(1, 10^{-4}/e(\mathbf{x}^t))d\gamma^t)$ [19], where d is the decreasing

[3]Updates (49) and (50) 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 (50) does not incur additional signaling compared with predetermined stepsizes.

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

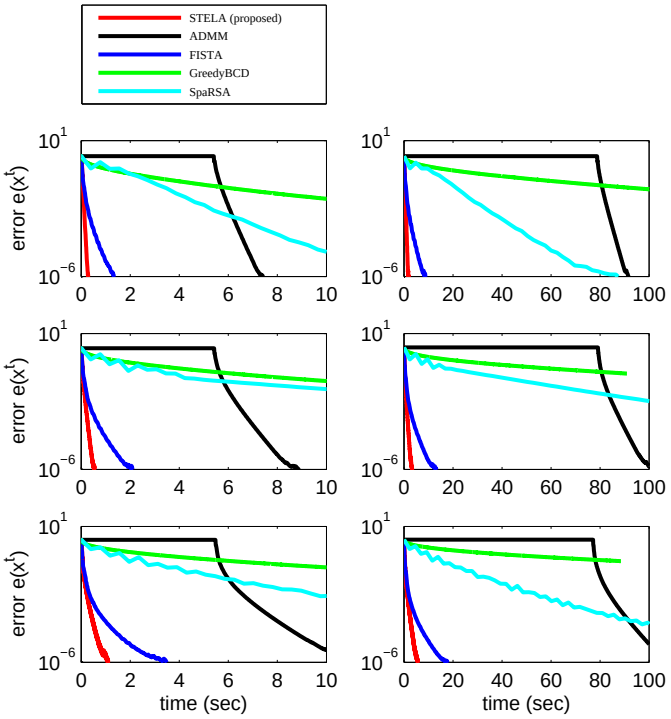

Figure 7. Time versus error of different algorithms for LASSO. In the left and right column, the dimension of $\mathbf{A}$ is 2000×4000 and 5000×10000, respectively. In the higher, middle and lower column, the density of $\mathbf{x}_{\text{true}}$ is 0.1, 0.2 and 0.4, respectively.

rate and $\gamma^0 = 0.9$. The code and the data generating the figure can be downloaded online [41].

Note that the error $e(\mathbf{x}^t)$ plotted in Fig. 6 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 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. 6. A similar behavior is also observed for $d = 10^{-3}$, until the stepsize becomes sufficiently small. 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 depends on the parameter setup (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 without requiring any parameter tuning.

We also compare in Fig. 7 the proposed algorithm `STELA` with other competitive algorithms in literature: `FISTA` [37], `ADMM` [12], `GreedyBCD` [42] and `SpaRSA` [43]. We simulated `GreedyBCD` of [42] because it exhibits guaranteed convergence. The dimension of $\mathbf{A}$ is 2000×4000 (the left column of Fig. 7) 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. 7), 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 an i.i.d. Gaussian 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 [43].

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 [41]. The comparison is made in terms of CPU time that is required until 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 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[5] in the initialization stage since it is required in the iterative variable update in the formal stage, cf. (49). The simulation results are averaged over 20 instances.

We observe from Fig. 7 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 in the case 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.

To further evaluate the performance of the proposed algorithm `STELA`, we test it on the benchmarking platform developed by the Optimization Group from Department of Mathematics at Technische Universität Darmstadt[6] and compare it with different algorithms in various setups (data set, problem dimension, etc.) for the basis pursuit problem [44]:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{x}\|_1 \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}. \end{aligned}$$

To adapt `STELA` for the basis pursuit problem, we use the

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

[6]Project website: http://wwwopt.mathematik.tu-darmstadt.de/spear/

augmented Lagrangian approach [13, 45]:

$$\begin{aligned}\mathbf{x}^{t+1} &= \|\mathbf{x}\|_1 + (\boldsymbol{\lambda}^t)^T(\mathbf{A}\mathbf{x}-\mathbf{b}) + \frac{c^t}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2,\\ \boldsymbol{\lambda}^{t+1} &= \boldsymbol{\lambda}^t + c^t(\mathbf{A}\mathbf{x}^{t+1}-\mathbf{b}),\end{aligned}$$

where $c^{t+1}=\min(2c^t,10^2)$ ($c^0=10/\left\|\mathbf{A}^T\mathbf{b}\right\|_\infty$), $\mathbf{x}^{t+1}$ is computed by STELA and this process is repeated until $\boldsymbol{\lambda}^t$ converges. The numerical results summarized in [46] show that, although STELA must be called multiple times before the Lagrange multiplier $\boldsymbol{\lambda}$ converges, the proposed algorithm for the basis pursuit problem based on STELA is very competitive in terms of running time and robust in the sense that it solved all problem instances in the test platform database.

V. Concluding Remarks

In this paper, we have proposed a novel iterative algorithm based on convex approximation. The most critical 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. After the first submission of this work, the proposed framework has been further customized to solve other emerging applications such as channel estimation in array signal processing [47] and symbol detection in massive MIMO systems [48].

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:

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

Using Assumption (A3), we get

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

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

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

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

$$\nabla f(\mathbf{y})^T(\mathbf{x}-\mathbf{y})=\nabla \tilde{f}(\mathbf{y};\mathbf{y})^T(\mathbf{x}-\mathbf{y})\geq 0,\ \forall \mathbf{x}\in\mathcal{X}, \quad (54)$$

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

$$\tilde{f}(\mathbb{B}\mathbf{y};\mathbf{y})=\tilde{f}(\mathbf{y};\mathbf{y}), \quad (55a)$$

or

$$\tilde{f}(\mathbb{B}\mathbf{y};\mathbf{y})<\tilde{f}(\mathbf{y};\mathbf{y}), \quad (55b)$$

If (55a) 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 (55b) can be true. We know from the pseudo-convexity of $\tilde{f}(\mathbf{x};\mathbf{y})$ in $\mathbf{x}$ (cf. Assumption (A1)) and (55b) that $\mathbb{B}\mathbf{y}\neq\mathbf{y}$ and

$$\nabla \tilde{f}(\mathbf{y};\mathbf{y})^T(\mathbb{B}\mathbf{y}-\mathbf{y})=\nabla f(\mathbf{y})^T(\mathbb{B}\mathbf{y}-\mathbf{y})<0, \quad (56)$$

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:

$$\nabla \tilde{f}(\mathbb{B}\mathbf{x}^t;\mathbf{x}^t)^T(\mathbf{x}-\mathbb{B}\mathbf{x}^t)\geq 0,\ \forall \mathbf{x}\in\mathcal{X}. \quad (57)$$

If (55a) 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 (54) (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

$$f(\mathbf{x}^{t+1})\leq f(\mathbf{x}^t). \quad (58)$$

It is easy to see (58) holds for the exact line search as well.

If (55b) 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

$$f(\mathbf{x}^{t+1})=f(\mathbf{x}^t+\gamma^t(\mathbb{B}\mathbf{x}^t-\mathbf{x}^t))<f(\mathbf{x}^t). \quad (59)$$

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 (58) and (59) 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

$$f\left(\lim_{\mathcal{T}_1\ni t\to\infty}\mathbf{x}^t\right)=f\left(\lim_{\mathcal{T}_2\ni t\to\infty}\mathbf{x}^t\right). \quad (60)$$

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), 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 it follows from the Maximum Theorem in [49, Ch. VI.3] that $\lim_{\mathcal{T}_s \ni t\to\infty} \mathbb{B}\mathbf{x}^t \in \mathcal{S}(\mathbf{y})$. Since both $f(\mathbf{x})$ and $\nabla f(\mathbf{x})$ are continuous, applying the Maximum Theorem again implies there is a $\mathcal{T}_{s'}$ such that $\mathcal{T}_{s'} \subseteq \mathcal{T}_s (\subseteq \mathcal{T})$ and $\left\{\mathbf{x}^{t+1}\right\}_{t\in\mathcal{T}_{s'}}$ 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 (59) that $f(\mathbf{y}') < f(\mathbf{y})$, but this would contradict (60). 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

$$\nabla_k f(\mathbb{B}_k\mathbf{y}, \mathbf{y}_{-k})^T(\mathbf{x}_k - \mathbb{B}_k\mathbf{y}) \geq 0, \forall\, \mathbf{x}_k \in \mathcal{X}_k. \tag{61}$$

Firstly, there must exist an index j such that

$$f(\mathbb{B}_j\mathbf{y}, \mathbf{y}_{-j}) < f(\mathbf{y}_j, \mathbf{y}_{-j}), \tag{62}$$

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 (62) that

$$\nabla_j f(\mathbf{y})^T(\mathbb{B}_j\mathbf{y} - \mathbf{y}_j) < 0. \tag{63}$$

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

$$\nabla_k f(\mathbf{y}_k, \mathbf{y}_{-k})^T(\mathbb{B}_k\mathbf{y} - \mathbf{y}_k) \geq 0. \tag{64}$$

Similarly, setting $\mathbf{x}_k = \mathbf{y}_k$ in (61) yields

$$\nabla_k f(\mathbb{B}_k\mathbf{y}, \mathbf{y}_{-k})^T(\mathbf{y}_k - \mathbb{B}_k\mathbf{y}) \geq 0. \tag{65}$$

Adding (64) and (65), 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 (65) 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

$$\begin{aligned} &\nabla_k f(\mathbf{y})^T(\mathbb{B}_k\mathbf{y} - \mathbf{y}_k) \leq \\ &-(\nabla_k f(\mathbb{B}_k\mathbf{y}, \mathbf{y}_{-k}) - \nabla_k f(\mathbf{y}))^T(\mathbb{B}_k\mathbf{y} - \mathbf{y}_k) \leq 0. \end{aligned} \tag{66}$$

Adding (63) and (66) 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 (31) 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 Trans. Inf. 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 Trans. Inf. 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 Trans. Inf. 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 Trans. Signal Process.*, vol. 51, no. 11, pp. 2839–2848, Nov. 2003.

[5] Z.-Q. Luo and S. Zhang, "Dynamic Spectrum Management: Complexity and Duality," *IEEE J. Sel. Topics Signal Process.*, 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 Trans. Inf. 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 Trans. Signal Process.*, 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 Trans. Signal Process.*, 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 J. Sel. Areas Commun.*, 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 Trans. Signal Process.*, 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 J. Sel. Topics Signal Process.*, 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] N. Parikh and S. Boyd, "Proximal Algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[15] P. L. Combettes and J.-C. Pesquet, "Proximal Splitting Methods in Signal Processing," in *Fixed-Point Algorithms for Inverse Problems*

in Science and Engineering, ser. Springer Optimization and Its Applications, H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, Eds. New York, NY: Springer New York, 2011, vol. 49, pp. 185–212.
[16] 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, 2013.
[17] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: Numerical methods*. Prentice Hall, 1989.
[18] P. He, "Correction of Convergence Proof for Iterative Water-Filling in Gaussian MIMO Broadcast Channels," *IEEE Trans. Inf. Theory*, vol. 57, no. 4, pp. 2539–2543, Apr. 2011.
[19] F. Facchinei, G. Scutari, and S. Sagratella, "Parallel Selective Algorithms for Nonconvex Big Data Optimization," *IEEE Trans. Signal Process.*, vol. 63, no. 7, pp. 1874–1889, Nov. 2015.
[20] G. Scutari, F. Facchinei, and L. Lampariello, "Parallel and Distributed Methods for Constrained Nonconvex Optimization-Part I: Theory," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 1929–1944, April 2017.
[21] J. M. Ortega and W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*. Academic, New York, 1970.
[22] O. L. Mangasarian, *Nonlinear programming*. McGraw-Hill, 1969.
[23] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge Univ Pr, 2004.
[24] J. Nocedal and S. J. Wright, *Numerical optimization*, 2nd ed. Springer, 2006.
[25] 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.
[26] R. W. Freund and F. Jarre, "An interior-point method for fractional programs with convex constraints," *Mathematical Programming*, vol. 67, no. 1, pp. 407–440, 1994.
[27] 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.
[28] P. Tseng and S. Yun, "A coordinate gradient descent method for nonsmooth separable minimization," *Mathematical Programming*, vol. 117, no. 1, pp. 387–423, 2009.
[29] 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
[30] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.
[31] W. Yu and J. M. Cioffi, "Sum Capacity of Gaussian Vector Broadcast Channels," *IEEE Trans. Inf. Theory*, vol. 50, no. 9, pp. 1875–1892, Sep. 2004.
[32] M. Grant and S. Boyd, "CVX: Matlab Software for Disciplined Convex Programming, version 2.0 beta," http://cvxr.com/, 2012.
[33] Y. Yang and M. Pesavento, "A Parallel Algorithm for Energy Efficiency Maximization in Massive MIMO Networks," in *Proc. IEEE GLOBECOM*, 2016.
[34] A. Zappone, L. Sanguinetti, G. Bacci, E. Jorswieck, and M. Debbah, "Energy-Efficient Power Control: A Look at 5G Wireless Technologies," *IEEE Trans. Signal Process.*, vol. 64, no. 7, pp. 1668–1683, April 2016.
[35] Y. Yang, G. Scutari, D. P. Palomar, and M. Pesavento, "A parallel decomposition method for nonconvex stochastic multi-agent optimization problems," *IEEE Trans. Signal Process.*, vol. 64, no. 11, pp. 2949–2964, June 2016.
[36] 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.
[37] 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.
[38] M. Elad, "Why simple shrinkage is still relevant for redundant representations?" *IEEE Trans. Inf. Theory*, vol. 52, no. 12, pp. 5559–5569, 2006.
[39] 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.
[40] 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 J. Sel. Topics Signal Process.*, vol. 1, no. 4, pp. 586–597, Dec. 2007.
[41] Y. Yang, http://www.nts.tu-darmstadt.de/home_nts/staff_nts/former_staff_nts/mitarbeiterdetails_32448.en.jsp.
[42] 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.
[43] S. Wright, R. Nowak, and M. Figueiredo, "Sparse Reconstruction by Separable Approximation," *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2479–2493, Jul. 2009.
[44] D. Lorenz, M. Pfetsch, and A. Tillmann, "Solving Basis Pursuit: Heuristic Optimality Check and Solver Comparison," *ACM Transactions on Mathematical Software*, vol. 41, no. 2, pp. 1–29, 2015.
[45] R. T. Rockafellar, "Augmented Lagrangians and Applications of the Proximal Point Algorithm in Convex Programming," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 97–116, 1976.
[46] J. Kuske and A. Tillmann, "Solving basis pursuit: Update," Tech. Rep., April 2016. [Online]. Available: http://www.mathematik.tu-darmstadt.de/~tillmann/docs/SolvingBPupdateApr2016.pdf
[47] C. Steffens, Y. Yang, and M. Pesavento, "Multidimensional sparse recovery for MIMO channel parameter estimation," in *Proc. European Signal Processing Conference (EUSIPCO)*, 2016.
[48] G. Hegde, Y. Yang, C. Steffens, and M. Pesavento, "Parallel low-complexity M-PSK detector for large-scale MIMO systems," in *Proc. IEEE sensor array and Multichannel signal processing workshop (SAM)*, 2016.
[49] C. Berge, *Topological Spaces: Including a Treatment of Multi-Valued Functions, Vector Spaces and Convexity*. Dover Publications, 1997.

Yang Yang received the B.S. degree in School of Information Science and Engineering, Southeast University, Nanjing, China, in 2009, and the Ph.D. degree in Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology. From Nov. 2013 to Nov. 2015 he had been a postdoctoral research associate at the Communication Systems Group, Darmstadt University of Technology, Darmstadt, Germany. He joined Intel Deutschland GmbH as a research scientist in Dec. 2015.

His research interests are in distributed solution methods in convex optimization, nonlinear programming, and game theory, with applications in communication networks, signal processing, and financial engineering.

Marius Pesavento (M'00) received the Dipl-Ing. and M.Eng. degrees from Ruhr-Universität Bochum, Bochum, Germany, and McMaster University, Hamilton, ON, Canada, in 1999 and 2000, respectively, and the Dr.-Ing. degree in electrical engineering from Ruhr-Universitat Bochum in 2005. Between 2005 and 2007, he was a Research Engineer at FAG Industrial Services GmbH, Aachen, Germany. From 2007 to 2009, he was the Director of the Signal Processing Section at MIMOon GmbH, Duisburg, Germany. In 2010, he became an Assistant Professor for Robust Signal Processing and a Full Professor for Communication Systems in 2013, at the Department of Electrical Engineering and Information Technology, Technische Universität Darmstadt, Darmstadt, Germany. His research interests include robust signal processing and adaptive beamforming, high-resolution sensor array processing, multiantenna and multiuser communication systems, distributed, sparse, and mixed-integer optimization techniques for signal processing and communications, statistical signal processing, spectral analysis, and parameter estimation. He has received the 2003 ITG/VDE Best Paper Award, the 2005 Young Author Best Paper Award of the IEEE Transactions on Signal Processing, and the 2010 Best Paper Award of the CROWNCOM conference. He is a Member of the Editorial Board of the EURASIP Signal Processing Journal, and served as an Associate Editor for the IEEE Transactions on Signal Processing in the terms 2012-2016. He is a Member of the Sensor Array and Multichannel Technical Committee of the IEEE Signal Processing Society, and the Special Area Teams "Signal Processing for Communications and Networking" and "Signal Processing for Multisensor Systems"of the EURASIP.