A Novel Iterative Convex Approximation Method
Yang Yang and Marius Pesavento

Abstract—In this paper, we propose a novel iterative algorithm based on convex approximation for a large class of possibly nonconvex optimization problems. The stationary points of the original optimization problems are found by solving a sequence of successively refined approximate problems, each of which is presumably much easier to solve than the original problem. To achieve convergence, the approximate problem only needs to be pseudo-convex while the stepsizes are determined by the exact or successive line search. The proposed method not only includes as special cases a number of existing methods, for example, the gradient method, the block coordinate descent (Gauss-Seidel) 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 nonsmooth optimization problems, which is to perform line search over a properly constructed differentiable function and is thus easier to implement than state-of-the-art techniques that directly operate on the original nonsmooth objective function. The advantages of the proposed algorithm are shown, both theoretically and numerically, by several example applications, namely, MIMO broadcast channel capacity computation, multi-portfolio optimization and LASSO in sparse signal recovery.

I. INTRODUCTION

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

\[
\begin{aligned}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in \mathcal{X},
\end{aligned}
\]

(1)

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

Problem (1) also includes some of nonsmooth optimization problems, if the nonsmooth function \( g(x) \) is convex:

\[
\begin{aligned}
\text{minimize} & \quad f(x) + g(x) \\
\text{subject to} & \quad x \in \mathcal{X},
\end{aligned}
\]

(2)

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

\[
\begin{aligned}
\text{minimize} & \quad f(x) + y \\
\text{subject to} & \quad x \in \mathcal{X}, g(x) \leq y.
\end{aligned}
\]

(3)

We do not assume \( f(x) \) is convex, so (1) is in general a nonconvex optimization problem. The focus of this paper is thus on iterative algorithms that find the stationary points of problem (1) efficiently. This includes applications in which (1) does not exhibit a closed-form solution, even when (1) is convex. As example applications consider 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 \( f(x) \) 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]. In portfolio optimization, \( f(x) \) is the expected return of the portfolio (to be maximized) and the \( \mathcal{X} \) characterizes the trading constraints [10]. In sparse signal recovery, \( f(x) + g(x) \) is the loss function characterizing the difference between the true signal and the estimated signal [11, 12].

Commonly used iterative algorithms belong to the class of descent direction methods such as the conditional gradient method and the gradient projection method [13], which often suffer from slow convergence. To speed up the convergence, algorithms based on nonlinear best-response have been widely studied. In particular, they are applicable if the constraint set of (1) has a Cartesian product structure \( \mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_K \):

\[
\begin{aligned}
\text{minimize} & \quad f(x_1, \ldots, x_K) \\
\text{subject to} & \quad x_k \in \mathcal{X}_k, k = 1, \ldots, K.
\end{aligned}
\]

(4)

Distributed and possibly parallel computation is particularly suitable for problem (4) because changing one variable does not affect the feasibility of other variables [14]. A notable distributed algorithm based on the nonlinear best-response is the block coordinate descent (BCD) method (also called Gauss-Seidel method) [13, Sec. 2.7]: in each iteration, only one variable is updated by its best-response, i.e., the vector that minimizes \( f(x) \) with respect to (w.r.t.) that variable only while the remaining variables are fixed, and the variables are updated sequentially. This method and its variants have been successfully adopted to many practical problems [1, 6, 7, 10, 15].

When the number of variables is large, the convergence speed of the BCD method may be slow due to the sequential update. Existing parallel update based on nonlinear best-response seems more desirable, but their convergence conditions are rather restrictive, c.f. the diagonal dominance condition on the objective function \( f(x) \) [14], which are not only difficult to satisfy but also hard to verify. This condition can be avoided if \( f(x) \) is convex and if we introduce stepsize (i.e., memory) when updating the variable, but the stepsize should be inversely proportional to the number of variables, so it tends to be overly conservative in large systems and inevitably slows down the convergence [2, 10, 16].

A recent progress in parallel algorithms has been made in [8, 17, 18], in which it was shown that, for a large class of (possibly nonconvex) optimization problems, the stationary point can be found by solving a sequence of successively refined approximate problems, and convergence to a stationary point is established if, among other conditions, the approxi-
mate function and stepsizes are properly selected. A systematic way was also presented there to construct an approximate problem preserving as much convexity of the original problem as possible to achieve faster convergence than classical conditional gradient method and gradient projection method.

Despite its novelty, the parallel algorithm proposed in [8, 17, 18] suffers from three limitations. Firstly, the approximate function must be strongly convex, and this is usually guaranteed by artificially adding a quadratic regularization term to the original objective function \( f(x) \), which however may destroy the desirable problem structure that could otherwise be exploited, e.g., to obtain computationally efficient closed-form solution of the approximate problems [6]. Secondly, decreasing stepsizes must be used. On the one hand, a slowly decaying stepsize is preferable to make notable progress and to achieve satisfactory convergence speed; on the other hand, theoretical convergence is guaranteed only when the stepsize decays fast enough. In practice, it is a difficult task on its own to find a decay rate that provides a good trade-off between convergence speed and convergence guarantee, and current practices mainly rely on heuristics [17]. Thirdly, the objective function \( f(x) \) has to be coercive and its gradient \( \nabla f(x) \) has to be Lipschitz continuous, so some important classes of optimization problems, for instance, semidefinite programming (SDP), cannot be solved by [8, 17].

The contribution of this paper consists in the development of a novel iterative convex approximation method to solve problem (1). Our work is directly motivated by [8, 9, 15, 17] but it greatly generalizes the results obtained there. In particular, the advantages of the proposed method are the following:

1) The approximate function only needs a weak form of convexity, namely, pseudo-convexity. This iterative method not only includes as special cases many existing methods, for example, [4, 6, 8, 9, 15, 17], but also opens new possibilities to construct approximate problems that are easier to solve. For example, in the MIMO IC, MAC and 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 and they typically yield faster convergence than constant stepsizes [2, 10, 16] and decreasing stepsizes [8, 17]. For example, unity stepsize can be used when the approximate problem structure and they typically yield faster convergence than classical conditional gradient method and gradient projection method.

The rest of the paper is organized as follows. In Section II we introduce some preliminaries on functions with different level of convexity, and iterative methods used to solve an optimization problem. The proposed method is presented in Section III. We show in Section IV the connection between the proposed method and many existing algorithms. In Section V, the proposed method is applied to solve different applications in communication networks (sum rate maximization problem of MIMO BC and IC), signal processing (LASSO) and financial engineering (multi-portfolio optimization). The paper is finally concluded in Section VI.

**Notation:** We use \( x, x \) and \( X \) to denote scalar, vector and matrix, respectively. \( X_{jk} \) is the \((j,k)\)-th element of \( X \); \( x_k \) is the \(k\)-th element of \( x \), and \( x = (x_k)_{k=1}^K \). \( x^{-1} \) is the element-wise inverse of \( x \), i.e., \( x_k^{-1} = 1/x_k \). \( \nabla f(x) \) denotes the gradient of \( f \) at \( x \). \( \nabla f(x) \) consists of the component-wise projection of \( f \) onto \([a, b] \): \( [x]_a^b = \max (\min( [x], b), a) \), and \( [x]^+ = [x]_{[0, \infty]} \). \( x \) is the smallest integer that is larger than or equal to \( x \). \( \text{d}(x) \) is a vector that consists of the diagonal elements of \( X \) and \( \text{diag}(x) \) is a diagonal matrix whose diagonal elements are as same as \( x \). \( \text{I} \) is a vector whose elements are equal to 1.

**II. PRELIMINARIES ON DESCENT DIRECTION METHOD AND CONVEX FUNCTIONS**

In this section, we introduce the concepts that are fundamental in the development of the rest of the paper.

**Stationary point.** A point \( y \) is a stationary point of (1) if

\[
(x - y)^T \nabla f(y) \geq 0, \quad \forall x \in \mathcal{X}. \quad (5)
\]

Condition (5) is the necessary condition of locally (and globally) optimal variables, and stationary points are accepted as solutions when (1) is nonconvex. If (1) is convex, stationary points coincide with (globally) optimal variables and condition (5) is also sufficient for \( y \) to (globally) optimal.

**Descent direction.** A vector \( d \) is a descent direction of the function \( f(x) \) at \( x = x^t \) if

\[
\nabla f(x^t)^T d^t < 0. \quad (6)
\]

If (6) is satisfied, \( f(x) \) can be decreased compared with \( f(x^t) \) when \( x \) is updated from \( x^t \) along \( d^t \), because in the Taylor expansion of \( f(x) \) around \( x = x^t \):

\[
f(x^t + \gamma d^t) = f(x^t) + \gamma \nabla f(x^t)^T d^t + o(\gamma),
\]

the first order term is negative in view of (6) and it would dominate the higher order terms when \( \gamma \) is sufficiently small. More rigorously, if \( d^t \) is a descent direction, there exists a \( \gamma > 0 \) such that [20, 8.2.1]

\[
f(x^t + \gamma d^t) < f(x^t), \forall \gamma \in (0, \gamma^t).
\]
Note that the other way around is not necessarily true, i.e., \( f(x^{t+1}) < f(x^t) \) does not necessarily mean \( x^{t+1} - x^t \) is a descent direction of \( f(x) \) at \( x = x^t \).

**Quasi-convex function.** A function \( h(x) \) is quasi-convex if
\[
h((1 - \alpha)x + \alpha y) \leq \max \{h(x), h(y)\}, \quad \forall x, y \in \mathcal{X}.
\]
It is strictly quasi-convex if the above inequality is satisfied with strict inequality whenever \( x \neq y \). A locally optimal solution \( y \) of a quasi-convex function \( h(x) \) over a convex set \( \mathcal{X} \) is also globally optimal, i.e.,
\[
h(x) \geq h(y), \quad \forall x \in \mathcal{X}.
\]

**Pseudo-convex function.** A function \( h(x) \) is pseudo-convex if [21]
\[
\nabla h(x)^T (y - x) \geq 0 \implies h(y) \geq h(x), \quad \forall x, y \in \mathcal{X}.
\]

Another equivalent definition of pseudo-convex functions turns out to be more useful in our context:
\[
h(y) < h(x) \implies \nabla h(x)^T (y - x) < 0.
\]
In other words, \( h(y) < h(x) \) implies \( y - x \) is a descent direction of \( h(x) \). A pseudo-convex function is also quasi-convex [21, Th. 9.3.5].

**Convex function.** A function \( h(x) \) is convex if
\[
h(y) \geq h(x) + \nabla h(x)^T (y - x), \quad \forall x, y \in \mathcal{X}.
\]
It is strictly convex if the above inequality is satisfied with strict inequality whenever \( x \neq y \). It is easy to see that a convex function is pseudo-convex.

**Strongly convex functions.** A function \( h(x) \) is strongly convex with constant \( \alpha \) if
\[
h(y) \geq h(x) + \nabla h(x)^T (y - x) + \frac{\alpha}{2} \|x - y\|^2, \quad \forall x, y \in \mathcal{X},
\]
for some positive constant \( \alpha \). The relationship of functions with different degree of convexity is summarized in Fig. 1.

### III. The Proposed Iterative Convex Approximation Method

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

At iteration \( t \), given the point \( x = x^t \), \( \tilde{f}(x; x^t) \) is the approximate function of \( f(x) \) around the point \( x = x^t \). Then the approximate problem is
\[
\begin{aligned}
\text{minimize} & \quad \tilde{f}(x; x^t) \\
\text{subject to} & \quad x \in \mathcal{X},
\end{aligned}
\]
and its optimal variable and solution set is denoted as \( \mathbb{B}x^t \) and \( \mathcal{S}(x^t) \), respectively:
\[
\mathbb{B}x^t \in \mathcal{S}(x^t) \triangleq \left\{ x^* \in \mathcal{X} : \tilde{f}(x^*; x^t) = \min_{x \in \mathcal{X}} \tilde{f}(x; x^t) \right\}.
\]

We assume that the approximate function \( \tilde{f}(x; y) \) satisfies the following technical conditions:
(A1) The approximate function \( \tilde{f}(x; y) \) is pseudo-convex in \( x \in \mathcal{X} \) for any given \( y \in \mathcal{X} \);
(A2) The approximate function \( \tilde{f}(x; y) \) is continuously differentiable in \( x \in \mathcal{X} \) for any \( y \in \mathcal{X} \) and continuous in \( y \in \mathcal{X} \) for any \( x \in \mathcal{X} \), and \( \nabla_{x} \tilde{f}(x; x) = \nabla_{x} f(x) \);

Based on (9), we define the mapping \( \mathbb{B}x \) that will be used to generate the sequence of points in the proposed algorithm:
\[
\mathcal{X} \ni x \mapsto \mathbb{B}x \in \mathcal{X}.
\]

We shall exploit the following properties of the mapping \( \mathbb{B}x \).

**Proposition 1.** Given the mapping \( \mathbb{B}x \) defined in (10), the following properties hold:
(i) A vector \( x \) is a stationary point of (1) if and only if \( x \in \mathcal{S}(x) \);
(ii) If \( x \) is not a stationary point of (1), then \( \mathbb{B}x - x \) is a descent direction of \( f(x) \):
\[
\nabla f(x)^T (\mathbb{B}x - x) < 0.
\]

**Proof:** See Appendix A.

If \( x^t \) is not a stationary point, \( \mathbb{B}x^t - x^t \) is a descent direction by Proposition 1. The vector update \( x^{t+1} \) in the \((t + 1)\)-th iteration is thus defined as:
\[
x^{t+1} = x^t + \gamma^t (\mathbb{B}x^t - x^t),
\]
where \( \gamma^t \in (0, 1] \) is an appropriate stepsize that can be determined by either the exact line search (also known as minimization rule) or the successive line search (also known as Armijo rule).

**Exact line search.** The stepsize is selected such that the function \( f(x) \) is decreased to the largest extent along the descent direction \( \mathbb{B}x^t - x^t \):
\[
\gamma^t \in \arg \min_{0 \leq \gamma^t \leq 1} f(x^t + \gamma(\mathbb{B}x^t - x^t)).
\]

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

If \( f(x) \) in (1) is convex and \( \gamma \) nulls the gradient of \( f(x^t + \gamma(\mathbb{B}x^t - x^t)) \), i.e., \( \nabla_{\gamma} f(x^t + \gamma(\mathbb{B}x^t - x^t)) = 0 \), then \( \gamma^t \) in (13) is simply the projection of \( \gamma^* \) onto the interval \([0, 1] \):
\[
\gamma^t = [\gamma^*]_{0}^{1} = \begin{cases}
1, & \text{if } \| \nabla_{\gamma} f(x^t + \gamma(\mathbb{B}x^t - x^t)) \|_{\gamma = 1} \geq 0, \\
0, & \text{if } \| \nabla_{\gamma} f(x^t + \gamma(\mathbb{B}x^t - x^t)) \|_{\gamma = 0} \leq 0,
\end{cases}
\]
\( \gamma^* \), otherwise.

---

This definition is consistent with the one used in [20] but different from the one used in [21].
If $0 \leq \gamma^t = \gamma^* \leq 1$, the constrained optimization problem in (13) is essentially unconstrained. In some applications it is possible to compute $\gamma^*$ analytically, e.g., if $f(x)$ is convex quadratic as in multi-portfolio optimization problem (Section V-C) and LASSO (Section V-D). Otherwise, for general convex functions, $\gamma^*$ can be found efficiently by bisection method. Since $f(x^t + \gamma^*(\mathbb{B}x^t - x^t))$ is convex in $\gamma$, it follows that $\nabla f(x^t + \gamma^*(\mathbb{B}x^t - x^t)) < 0$ if $\gamma < \gamma^*$ and $\nabla f(x^t + \gamma^*(\mathbb{B}x^t - x^t)) > 0$ if $\gamma > \gamma^*$. Given an interval $[\gamma_{\text{low}}, \gamma_{\text{up}}]$ containing $\gamma^*$ (the initial value of $\gamma_{\text{low}}$ and $\gamma_{\text{up}}$ is 0 and 1, respectively), set $\gamma_{\text{mid}} = (\gamma_{\text{low}} + \gamma_{\text{up}})/2$ and refine $\gamma_{\text{low}}$ and $\gamma_{\text{up}}$ according to the following rule:

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

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 $f(x)$ (e.g., convexity) can be exploited to efficiently compute $\gamma^t$ 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 stepsizes $\gamma^t$ is set to be $\gamma^t = \beta^m$, where $m_t$ is the smallest nonnegative integer $m$ satisfying the following inequality:

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

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

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

**Theorem 2.** Consider the sequence $\{x^t\}$ generated by Algorithm 1. Assume Assumptions (A1)-(A2) as well as the following assumptions are satisfied:

(A3) The solution set $\mathcal{S}(x^1)$ is nonempty for $t = 1, 2, \ldots$;

(A4) Given any convergent subsequence $\{x^t\}_{t \in \mathcal{T}}$ where $\mathcal{T} \subseteq \{1, 2, \ldots\}$, the sequence $\{\mathbb{B}x^t\}_{t \in \mathcal{T}}$ is bounded.

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

**Proof:** See Appendix B.

If the approximate function $\tilde{f}(x; x^t)$ has some other properties, it is possible to determine the stepsize in a more straightforward way. For example, if $\tilde{f}(x; x^t)$ is a global upper bound of $f(x)$ that is exact at $x = x^t$:

(A5) $f(x; x^t) \geq f(x)$ and $\tilde{f}(x; x^t) = f(x^t)$,

then we can always set $\gamma^t = 1$ and (12) reduces to $x^{t+1} = \mathbb{B}x^t$. To see this, we first remark that $\gamma^t = 1$ is an optimal variable of the following problem:

$$1 \in \arg\min_{0 \leq \gamma \leq 1} \tilde{f}(x^t + \gamma(\mathbb{B}x^t - x^t); x^t),$$

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

$$\begin{align*}
\tilde{f}(\mathbb{B}x^t; x^t) &\leq \tilde{f}(x^t + \beta^m(\mathbb{B}x^t - x^t); x^t) \\
&\leq \tilde{f}(x^t) + \alpha \beta^m \nabla \tilde{f}(x^t; x^t)^T(\mathbb{B}x^t - x^t) \\
&= f(x^t) + \alpha \beta^m \nabla f(x^t)^T(\mathbb{B}x^t - x^t),
\end{align*}$$

where Assumptions (A2) and (A5) are used in the last equality. Invoking Assumption (A5) again, we immediately get

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

In other words, applying the exact line search on $\tilde{f}(x^t + \gamma(\mathbb{B}x^t - x^t); x^t)$ as in (15) is equivalent to applying the successive line search on $f(x^t + \gamma(\mathbb{B}x^t - x^t))$ as in (17). Thus the conditions of Theorem 2 are satisfied and Algorithm 1 converges under a constant unity stepsize.

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

**On the uniqueness of $\mathbb{B}x^t$.** In general, $\mathbb{B}x^t$ may not be unique. However, if $f(x; x^t)$ is hemivariate, i.e., it is not constant on any line segment of $X$ (or equivalently, there exists no distinct points $x_1$ and $x_2$ such that $f(\alpha x_1 + (1-\alpha)x_2; x^t) = f(x_1; x^t)$ for all $0 \leq \alpha < 1$), then $\mathbb{B}x^t$ is unique. This condition is automatically satisfied if $f(x; x^t)$ is strictly or strongly convex.

**On the pseudo-convexity of the approximate function $\tilde{f}(x; x^t)$.** Assumption (A1) is tight in the sense that if Assumption (A1) is not satisfied, Proposition 1 does not hold any more. Consider the following simple example:

$$f(x) = x^3, \quad -1 \leq x \leq 1,$$

and $x^t = 0$. It is easy to verify that if

$$\tilde{f}(x; x^t) = x^3,$$

which is quasi-convex but not pseudo-convex, all assumptions except (A1) are satisfied. It is easy to see that $\mathbb{B}x^t = -1$, however

$$(\mathbb{B}x^t - x^t)^T \nabla f(x^t) = (-1 - 0) \cdot 0 = 0,$$

and thus $\mathbb{B}x^t - x^t$ is not a descent direction and Proposition 1 does not hold any more.

**On the approximate problem (9).** The only requirement on the approximate function $\tilde{f}(x; x^t)$ is that it is pseudo-convex, and to the best of our knowledge, this is the weakest condition in the literature. As a result, it enables us to construct new approximate functions that can often be optimized more easily or even in closed-form, and results in a significant reduction of the computational cost if the approximate problem must otherwise be optimized by iterative algorithms as in standard solvers [22]. Assumption (A2) is a standard condition in successive convex approximation techniques and can be satisfied by many approximation functions, cf. Section IV. Sufficient conditions for Assumptions (A3)-(A4) are that either the feasible set $X$ in (8) is bounded or the approximate function in (8) is strongly convex [23]. We will show how...
these assumptions are satisfied by the applications considered in Section V.

A. Nonsmooth optimization problems

Suppose \( f(x; x^t) \) is an approximate function of \( f(x) \) around \( x = x^t \) satisfying Assumptions (A1)-(A2). Then the approximate problem for problem (2) is

\[
\begin{align*}
\text{minimize} & \quad f(x; x^t) + y \\
\text{subject to} & \quad x \in \mathcal{X}, \; g(x) \leq y,
\end{align*}
\]

(18)

and we denote its optimal \( x \) and \( y \) as \( Bx^t \) and \( y^*(x^t) \). Then the stepsize \( \gamma^t \) based on the exact line search (13) is

\[
\gamma^t \in \arg \min_{0 \leq \gamma \leq 1} \left\{ f(x^t + \gamma(Bx^t - x^t)) + y^* + \gamma(y^*(x^t) - y^*) \right\},
\]

(19)

where \( y^* = g(x^t) \). Besides, it is easy to infer from (18) that \( g^*(x^t) = g(Bx^t) \). The notations in (18)-(19) can thus be simplified by removing the auxiliary variable \( y \): (18) and (19) is respectively equivalent to

\[
Bx^t = \arg \min_{x \in \mathcal{X}} \left\{ f(x; x^t) + g(x) \right\}
\]

(20)

and

\[
\gamma^t \in \arg \min_{0 \leq \gamma \leq 1} \left\{ f(x^t + \gamma(Bx^t - x^t)) + y^* + (g(Bx^t) - g(x^t)) \right\}.
\]

(21)

The convergence of Algorithm 1 with \( Bx^t \) defined in (20) and \( \gamma^t \) defined in (21) readily follows from Theorem 2.

IV. SPECIAL CASES OF THE PROPOSED METHOD

In this section, we interpret some existing methods in the context of Algorithm 1 and show that how they represent special cases of the proposed algorithm.

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

\[
f(x; x^t) = \nabla f(x^t)^T (x - x^t).
\]

(23)

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

Gradient projection method. In this method, \( Bx^t \) is given by [13, Sec. 2.3]

\[
Bx^t = [x^t - s^t \nabla f(x^t)]_{\mathcal{X}},
\]

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

\[
f(x; x^t) = \nabla f(x^t)^T (x - x^t) + \frac{1}{2s^t} \| x - x^t \|_2^2,
\]

which is the first-order approximation of \( f(x) \) plus a quadratic regularizing term introduced for numerical stability [14]. Then the stepsize is selected by either the exact line search or successive line search.

Jacobi algorithm. If \( f(x) \) is componentwise convex, the approximate function can be defined as

\[
f(x; x^t) = \sum_{k=1}^{K} f(x_k, x^t_{-k}) + \frac{\tau_k}{2} \| x_k - x^t_k \|_2^2,
\]

(24)

where \( \tau_k \geq 0 \) for \( k = 1, \ldots, K \). The \( k \)-th component function in (24) is obtained from the original function \( f(x) \) by fixing all variables except \( x_k \), i.e., \( x_{-k} = x^t_{-k} \), and further adding an (optional) quadratic regularization term. The resulting approximate problem is given by

\[
\begin{align*}
\text{minimize} & \quad \sum_{k=1}^{K} f(x_k, x^t_{-k}) \\
\text{subject to} & \quad x \in \mathcal{X}.
\end{align*}
\]

(25)

If the constraint set \( \mathcal{X} \) has a Cartesian product structure (4), problem (25) can be decomposed into a set of smaller subproblems, one for each variable, which are then solved in parallel and commonly known as the Jacobi update:

\[
B_k x^t \in \arg \min_{x_k \in \mathcal{X}_k} f(x_k, x^t_{-k}), \; k = 1, \ldots, K,
\]

(26)

where \( B_k x^t \) can be interpreted as variable \( x_k \)'s best-response to other variables \( x_{-k} = (x_j)_{j \neq k} \) when \( x_{-k} = x^t_{-k} \).

To guarantee the convergence, one of the conditions proposed in [8] is that \( \tau_k > 0 \) for all \( k \) in (24). However, this may destroy the convenient structure that could otherwise be exploited. For example, in the case \( \tau_k = 0 \), (25) may exhibit a closed-form solution. In the proposed method, we can set \( \tau_k = 0 \) and \( f(x) \) only needs to be componentwise pseudo-convex. To the best of our knowledge, this is the weakest convergence condition on Jacobi algorithms in the literature.

DC programming. If the objective function in (1) is the difference of two convex functions \( f_1(x) \) and \( f_2(x) \):

\[
f(x) = f_1(x) - f_2(x),
\]
the following approximate function can be used:
\[
\hat{f}(x; x^t) = f_1(x) - (f_2(x^t) + \nabla f_2(x^t)^T(x - x^t)).
\]
Since \( f_2(x) \) is convex and \( f_2(x) \geq f_2(x^t) + \nabla f_2(x^t)^T(x - x^t) \), Assumption (A5) is satisfied and a constant unity stepsize can be chosen. This will generally yield faster convergence than [8] which uses decreasing stepsizes.

A. Successive upper bound minimization and block coordinate descent algorithm

In this subsection, we show that if \( f(x) \) is componentwise pseudo-convex, the successive upper-bound minimization (SUM) method proposed in [15] and the BCD are essentially a descent direction method combined with the successive and exact line search, respectively.

Define \( z^t_k = (x_1^{t+1}, \ldots, x_{k-1}^{t+1}, x_{k+1}^{t+1}, \ldots, x_K^{t+1}) \) (so \( z^t_K = x^{t+1} \)) where \( k = \text{mod} (t, K) + 1 \) and
\[
B_k z_{k-1} = \arg \min_{x_k} \hat{f}_k(x_k; z_{k-1}),
\]
where \( \hat{f}_k(x_k; z_{k-1}^t) \) is an approximate function of \( f(x) \) around \( x = z_{k-1}^t \) which satisfies the following assumptions for any \( k = 1, \ldots, K \):

(B1) \( \hat{f}_k(x_k; z_{k-1}^t) \) is pseudo-convex and has a unique minimum for any given \( z_{k-1}^t \);

(B2) \( \hat{f}_k(x_k; z_{k-1}^t) \) is continuously differentiable for any given \( z_{k-1}^t \) and \( \nabla \hat{f}_k(x_k; z_{k-1}^t) = \nabla f(z_{k-1}^t) \);

(B3) \( \hat{f}_k(x_k; z_{k-1}^t) \geq f(x_1^{t+1}, \ldots, x_{k-1}^{t+1}, x_k, x_{k+1}^{t+1}, \ldots, x_K^{t+1}) \) with equality achieved when \( x_k = \hat{x}_k^t \).

According to Proposition 1 and Assumption (B1)-(B2) and assuming without loss of generality (w.l.o.g.) \( x_k^t \neq B_k z_{k-1}^t \), we can show by the same line of arguments used in (16)-(17) that the following update
\[
x_k^{t+1} = B_k z_{k-1}^t = x_k^t + \gamma_k^t (B_k z_{k-1}^t - x_k^t),
\]
where \( z_{k-1}^t \) is a stationary point (proved later in Appendix C). The approximate function can be used:
\[
\hat{f}(x; x^t) = f_1(x) - (f_2(x^t) + \nabla f_2(x^t)^T(x - x^t)).
\]
If Assumptions (B1)-(B3) are satisfied and the lower set \( L(x^0) = \{x : f(x) \leq f(x^0)\} \) is bounded, then any limit point of the sequence \( \{x^t\} \) generated by (28) is a stationary point of (4).

**Proof:** See Appendix C.

It is easy to see that \( \hat{f}_k(x_k; z_{k-1}^t) = f_k(x_1^{t+1}, \ldots, x_{k-1}^{t+1}, x_k, x_{k+1}^{t+1}, \ldots, x_K^{t+1}) \) is a special case of the approximate function in the sense that the inequality in Assumption (B3) is always satisfied with equality. In this case, (28) simply reduces to
\[
x_k^{t+1} = \arg \min_{x_k} f(x_1^{t+1}, \ldots, x_{k-1}^{t+1}, x_k, x_{k+1}^{t+1}, \ldots, x_K^{t+1}),
\]
and it is just the variable update rule in the BCD method. Since no approximation is used in (29), \( \gamma_k^t = 1 \) is the stepsize obtained when the exact line search is directly applied to \( f(x) \) along the direction \( x_k^{t+1} - x_k^t \), otherwise the optimality and uniqueness of \( x_k^{t+1} \) would be contradicted.

The variable \( x_k^{t+1} \) in (29) is unique if the function \( f(x_1^{t+1}, x_k, x_{k+1}^{t+1}, \ldots, x_K^{t+1}) \) is hemivariate. If this assumption does not hold, an additional quadratic regularization can be included so that the approximate function is hemivariate [26]:
\[
x_k^{t+1} = \arg \min_{x_k} \left\{ f(x_1^{t+1}, \ldots, x_{k-1}^{t+1}, x_k, x_{k+1}^{t+1}, \ldots, x_K^{t+1}) + \frac{\tau_k}{2} \|x_k - \hat{x}_k^t\|^2 \right\},
\]
where \( \tau_k \) is a given positive constant. The approximate function in (30) is easily seen to satisfy Assumptions (B1)-(B3), so the conclusion of Proposition 3 holds.

V. EXAMPLE APPLICATIONS

A. MIMO Broadcast Channel Capacity Computation

In MIMO BC, assume \( H_k \) is the channel from the base station to user \( k \), \( Q_k \) is the transmit covariance matrix of the signal from the base station to user \( k \), and \( R > 0 \) is the covariance matrix of the additive Gaussian noise at the base station. Then the sum capacity of MIMO BC is [27]
\[
\max \log |R + \sum_{k=1}^K H_k Q_k H_k^H| - \log |R|
\]
subject to \( Q_k \geq 0, k = 1, \ldots, K, \sum_{k=1}^K \text{tr}(Q_k) \leq P, \) (31)
where \( P \) is the power budget at the base station.

Although problem (31) is convex, its solution cannot be expressed in closed-form and can only be solved iteratively. To apply Algorithm 1, we invoke (24)-(25) and the approximate problem at \( t \)-th iteration is
\[
\max \sum_{k=1}^K \log |R_k(Q_k^{t,k}) + H_k Q_k H_k^H|
\]
subject to \( Q_k \geq 0, k = 1, \ldots, K, \sum_{k=1}^K \text{tr}(Q_k) \leq P, \) (32)
where \( R_k(Q_k^{t,k}) \triangleq R + \sum_{j \neq k} H_j Q_j H_j^H \). The approximate function is concave in \( Q \) and differentiable in both \( Q \) and \( Q' \), so Assumptions (A1)-(A2) are satisfied. Since the constraint
set in (32) is compact, the approximate problem (32) always has a solution and Assumptions (A3)-(A4) are satisfied.

Problem (32) is convex and the constraint set has a nonempty interior, so strong duality holds and (32) can be solved from the dual domain by relaxing the sum-power constraint into the Lagrangian [28]:

$$
\mathbb{B}Q^t = \arg \max_{(Q_k \succeq 0)_{k=1}^K} \left\{ \sum_{k=1}^K \log \left( R_k(Q_{-k}^t) + H_k Q_k H_k^H \right) - \lambda^* \left( \sum_{k=1}^K \text{tr}(Q_k) - P \right) \right\},
$$

where $\mathbb{B}Q^t = (\mathbb{B}_k Q^t)_{k=1}^K$ and $\lambda^*$ is the optimal Lagrange multiplier that satisfies the following conditions: $\lambda^* \geq 0$, $\sum_{k=1}^K \text{tr}(\mathbb{B}_k Q^t) - P \leq 0$, $\lambda^* \left( \sum_{k=1}^K \text{tr}(\mathbb{B}_k Q^t) - P \right) = 0$, and can be found efficiently by bisection.

The problem in (33) is uncoupled among different variables $Q_k$ 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: $\mathbb{B}Q^t = (\mathbb{B}_k Q^t)_{k=1}^K$ and

$$
\mathbb{B}_k Q^t = \arg \max_{Q_k \succeq 0} \left\{ \log \left( R_k(Q_{-k}^t) + H_k Q_k H_k^H \right) - \lambda^* \text{tr}(Q_k) \right\},
$$

and $\mathbb{B}_k Q^t$ has a closed-form expression based on waterfilling solution [2]. Thus problem (32) has a closed-form solution up to a Lagrange multiplier that can be found efficiently by bisection. After the update (descent) direction $\mathbb{B}Q^t - Q^t$ is found, the base station can implement the exact line search to determine the stepsize using the bisection method described in Section III.

We remark that the proposed approximate problem (32) holds for column-rank deficient channel matrices $H_k$ as well. In that case, problem (32) is convex but not strongly convex. If the approximate function in [8] is used, an additional quadratic regularization term must be included into (32) (and thus (34)) to make the approximate problem strongly convex, cf. (24), but the resulting approximate problem no longer has a closed-form solution and more computational resources are required.

**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). We assume white Gaussian noise $\mathbf{R} = \sigma^2 \mathbf{I}$ with $\sigma^2 = 1$, and $P = 10$ dB. The simulation results are averaged over 20 instances.

We apply Algorithm 1 where the stepsize is calculated by the exact line search, and compare it with the iterative algorithm proposed in [2, 16], which uses the same approximate problem (32) but 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 [2] requires many more iterations to converge. This is because the exact line search can always give the largest increase in objective value while fixed stepsize tends to be very overly conservative. Employing the exact line search adds complexity as compared to the simple choice of a fixed stepsize, but since the objective function of (31) is convex, the additional cost to perform the exact line search based on the bisection method is affordable. More specifically, it takes 0.0023 seconds to solve (32) and 0.0018 seconds to perform the exact line search (the software/hardware environment will be specified later). Therefore, the overall CPU time (time per iteration×number of iterations) is still dramatically decreased due to the notable reduction in the number of iterations. Besides, in contrast to [2], increasing the number of users $K$ does not slow down the convergence, so the proposed algorithm is scalable in large scale networks.

We also compare the proposed algorithm with the iterative algorithm [18], which uses the approximate problem (32) but with an additional quadratic regularization term, cf. (24), 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 and it controls how fast the stepsize decreases. We can see from Fig. 3 that the convergence behavior of [18] is quite sensitive to the decreasing rate $d$. The choice $d = 0.01$ performs well when the number of transmit and receive antenna is (5,4), but it is no longer a good choice when the number of transmit and receive antenna is (10,8). A good decreasing rate $d$ is usually dependent on problem parameters and no general rule is equally good 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 (32), while the approximate problem proposed in [18] does not have a closed-form solution and can only be solved iteratively. Specifically, it takes CVX (version 2.0 [29]) 21.1785 seconds (based on the dual approach (34) where λ∗ is found by bisection). Therefore, the overall complexity per iteration of the proposed algorithm is much lower than that of [18].

B. MIMO Interference Channel Sum-Rate Maximization

In MIMO IC, suppose \( H_{jk} \) is the channel from the transmitter of user \( k \) to the receiver of user \( j \) and \( R_{nk} \) is the covariance matrix of additive Gaussian noise at the receiver of user \( k \). The sum rate maximization problem is

\[
\begin{align*}
\text{maximize} & \quad \sum_{k=1}^{K} \log |I + H_{kk}Q_kH_{kk}^H R_k(Q)^{-1}| \\
\text{subject to} & \quad Q_k \succeq 0, \text{tr}(Q_k) \leq P, \quad k = 1, \ldots, K,
\end{align*}
\]

where \( R_k(Q) \triangleq R_{kk} + \sum_{j \neq k} H_{kj}Q_jH_{kj}^H \) and it is the covariance matrix of noise plus interference at the receiver of user \( k \).

Employing the SUM method, the approximate problem at iteration \( t \) is:

\[
\begin{align*}
\mathbb{E}_k Q^t & = \arg \max_{Q_k \succeq 0, \text{tr}(Q_k) \leq P} \left\{ \log |I + H_{kk}Q_kH_{kk}^H R_k(Q^t)^{-1}| + \langle \Pi_k(Q^t), Q_k - Q_k^{t-1} \rangle \right\}, \\
\text{where } k & = \text{mod}(t - 1, K) + 1, \quad \langle X, Y \rangle \triangleq \text{tr}(X^HY) \quad \text{and} \quad \Pi_k(Q) \triangleq \sum_{j \neq k} \nabla Q_k \log |I + H_{jj}Q_jH_{jj}^H R_j(Q)^{-1}| \\
& = -\sum_{j \neq k} H_{kj}^H R_j(Q^t)H_{kj}, \\
\hat{R}_j(Q) & \triangleq R_j(Q)^{-1} - \langle R_j(Q) + H_{jj}Q_jH_{jj}^H \rangle^{-1}.
\end{align*}
\]

The optimization problem in (36a) is convex and \( \mathbb{E}_k Q^t \) has a closed-form expression based on generalized water-filling solution [6, 30]. It was shown in [6] that Assumptions (B2) and (B3) is satisfied by the approximation function in (36a), so the variables are simply updated as follows:

\[
Q_k^{t+1} = \begin{cases} 
\mathbb{E}_k Q^t, & \text{if } k = \text{mod}(t - 1, K) + 1, \\
Q_k^t, & \text{otherwise}.
\end{cases}
\]

In other words, the users sequentially update their transmit covariance matrices and at any iteration, only one user updates the transmit covariance matrix. More details on the implementation issues can be found in [6].

Update (36) was firstly proposed in [6] but the convergence was left as an open issue there. In addition to Assumptions (B2)-(B3), Assumption (B1) is satisfied because \( \mathbb{E}_k Q^t \) in (36a) is unique [30, Lemma 2], and \( L(Q^t) \) is bounded because the constraint set of (35) is bounded, so the conditions of Proposition 3 are satisfied, and thus every limit point of the sequence \( \{Q^t\}_t \) generated by (36) is a stationary point of (35).

C. Multi-Portfolio Optimization Problem

Consider the multi-portfolio optimization problem studied in [10]. Given a total number of \( N \) assets and assume the return of the \( n \)-th asset over a single-period investment horizon is modeled as a random variable denoted by \( r_n \).

Let \( \mu = (\mu_n)_{n=1}^N \) be the vector of expected returns where \( \mu_n = \mathbb{E}[r_n] \), and \( \mathbf{R} = (R_{mn})_{m,n} \) be the covariance matrix where \( R_{mn} = \mathbb{E}[(r_m - \mu_m)(r_n - \mu_n)] \). Since a portfolio manager is usually managing multiple accounts simultaneously, we let \( x_k = (x_{k,n})_{n=1}^N \) be the vector of weights for account \( k \) defining the proportion of wealth, and different accounts influence each other negatively through the so-called market impact cost. Suppose the market impact price function is \( \Omega(\sum_{k=1}^{K} x_k) \) where \( \Omega > 0 \) is given, the return of account \( k \) is given by the following function [10]:

\[
f_k(x_k, x_{-k}) = \mu^T x_k - \frac{1}{2} \rho_k x_k^T \mathbf{R} x_k - \frac{1}{2} x_k^T \Omega(\sum_{j=1}^{K} x_j),
\]

where \( \rho_k > 0 \) and it is the so-called risk-aversion parameter.

The goal of the portfolio manager is to maximize the total welfare of all accounts he manages, i.e.,

\[
\max_{\{x_k\}_{k=1}^{K}} \sum_{k=1}^{K} f_k(x_k, x_{-k})
\]

subject to \( x_k \in X_k \), \( k = 1, \ldots, K \),

\[
(37)
\]

where \( X_k \) is the (closed and convex) constraint set characterized by the trading restrictions such as budget constraint, diversification constraint and no-shortselling constraint.

Problem (37) is convex [10] and can be solved by standard interior point methods. However, the variable dimension is \( KN \), which is quite large when either \( K \) or \( N \) is large. In this case, standard interior point methods may be computationally inefficient and we instead apply Algorithm 1. The approximate problem for account \( k \) is (cf. (26))

\[
\mathbb{B}_k x^t = \arg \max_{x_k \in X_k} U(x_k, x_{-k}^t), \quad k = 1, \ldots, K,
\]

(38)

where \( U(x) \) is the objective function in (37):

\[
\begin{align*}
U(x) & \triangleq \sum_{k=1}^{K} f_k(x_k, x_{-k}) = \mu^T x - \frac{1}{2} x^T M x, \\
\bar{\mu} & \triangleq (\mu_k)_{k=1}^K, \quad M \triangleq \text{diag}(\rho) \otimes \mathbf{R} + J_K \otimes \Omega, \quad \text{and } J_K \text{ is a } K \times K \text{ matrix with all entries equal to 1. The approximate problems for different users are solved in parallel, each of which has a variable dimension of } N \text{ and is thus much easier to solve than the original problem (37).}
\end{align*}
\]

Since \( U(x) \) is convex quadratic, the exact line search has a closed-form expression (detailed steps are omitted):

\[
\gamma^t = \arg \max_{0 \leq \gamma \leq 1} U(x^t + \gamma(\mathbb{B}x^t - x^t))
\]

\[
\quad = \left[-\left(\frac{\mu - \mathbb{M}x^t}{\mathbb{B}x^t - x^t}\right)^T (\mathbb{B}x^t - x^t) \right]^{-1}.
\]

(39)

The exact line search (39) typically yields much faster convergence than fixed stepsize \( \gamma^t = 1/K \) used in [10].

The verification of Assumptions (A1)-(A3) is straightforward and we verify Assumption (A4) only. Since the optimization problem in (38) is strongly convex [10], it follows from [23] that \( \mathbb{B}x \) defined in (38) is a continuous function in the sense that if \( \lim_{\tau \to \infty} x^t = y \), then \( \lim_{\tau \to \infty} \mathbb{B}x^t = \mathbb{B}y \). Assumption (A4) is thus verified.

Simulations. The parameters are set as follows. The number of assets is \( N = 100 \) and the number of users is \( K = 10 \). The
The LASSO Problem

LASSO is an important and widely studied problem in sparse signal recovery [11, 12, 31, 32]:

$$\min_x \frac{1}{2} \|Ax - b\|_2^2 + \mu \|x\|_1,$$

(40)

where $A \in \mathbb{R}^{N \times K}$ (with $N \ll K$), $b \in \mathbb{R}^{K \times 1}$ and $\mu > 0$ are given parameters. Problem (40) has the same form of (2) with the following decomposition:

$$f(x) \triangleq \frac{1}{2} \|Ax - b\|_2^2, \quad \text{and} \quad g(x) \triangleq \mu \|x\|_1.$$  

(41)

Problem (40) is convex, but the objective function is nondifferentiable and cannot be minimized analytically. To apply Algorithm 1, scalar decomposition is adopted, i.e., $x = (x_k)_{k=1}^K$. Recalling (20) and (24), the approximate problem is

$$\mathbb{B}x^t = \arg \min_x \{ \sum_{k=1}^K f(x_k, x_{-k}^t) + g(x) \}.$$  

(42)

Note that $g(x)$ can be decomposed among different components of $x$, i.e., $g(x) = \sum_{k=1}^K g(x_k)$, so the vector problem (42) reduces to $K$ independent scalar problems that can be solved in parallel, and the $k$-th scalar problem is

$$\mathbb{B}_k x^t = \arg \min_{x_k} \{ f(x_k, x_{-k}^t) + g(x_k) \} = \frac{g(A^T A)^{-1} S_k (r_k(x^t))}{d_k(A^T A)^{-1} S_k (r_k(x^t))},$$

(43)

where $S_k(b) = [b - a]^+ - [-b - a]^+$ is the so-called soft-thresholding operator [32] and

$$r(x) \triangleq d(A^T A) \circ x - A^T (Ax - b),$$

(44)

or more compactly:

$$\mathbb{B}x^t = \left( \mathbb{B}_k x^t \right)_{k=1}^K = d(A^T A)^{-1} \circ S_{\mu} (r(x^t)).$$

(45)

The stepsize based on the proposed exact line search (21) is

$$\gamma^t = \arg \min_{0 \leq \gamma \leq 1} \{ f(x^t + \gamma (\mathbb{B}x^t - x^t)) + \gamma (g(\mathbb{B}x^t) - g(x^t)) \} = \arg \min_{0 \leq \gamma \leq 1} \left\{ \frac{1}{2} \|A(x^t + \gamma(\mathbb{B}x^t - x^t)) - b\|_2^2 \right\} + \gamma \mu \|\mathbb{B}x^t\|_1 - \|x^t\|_1 \right\}$$

$$= \left[ - (Ax^t - b)^T A (\mathbb{B}x^t - x^t) + \mu (\|\mathbb{B}x^t\|_1 - \|x^t\|_1) \right]_{0}.$$  

(45)

The exact line search is a convex quadratic optimization problem with a scalar variable and a bound constraint, so it exhibits a closed-form solution (45). Therefore, both the update direction and stepsize can be calculated in closed-form. We name the proposed update (44)-(45) as Soft-Thresholding with Exact Line search Algorithm (STELA).

The proposed update (44)-(45) 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 (44). This is in the same spirit of [17] and the convergence speed is generally faster than BCD [33] or gradient-based update [34]. Secondly, the proposed exact line search (45) 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 than [17] where decreasing stepsizes are used.

**Computational complexity.** The computational overhead associated with the proposed exact line search (45) can significantly be reduced if (45) is carefully implemented as outlined in the following. The most complex operation in (45) is the matrix-vector multiplication, namely, $Ax^t - b$ in the numerator and $A(\mathbb{B}x^t - x^t)$ in the denominator. On one hand, $Ax^t - b$ is already available from $r(x^t)$, which anyway must be computed in order to compute the best-response in (44). On the other hand, the matrix-vector multiplication $A(\mathbb{B}x^t - x^t)$ is also required for the computation of $Ax^{t+1} - b$ in the following as this update can alternatively be computed as:

$$Ax^{t+1} - b = A(x^t + \gamma^t (\mathbb{B}x^t - x^t)) - b = (Ax^t - b) + \gamma^t A(\mathbb{B}x^t - x^t),$$

(46)

where only vector addition is involved. As a result, the stepsize (45) does not incur any additional matrix-vector multiplications, but only affordable vector-vector multiplications.

Figure 4. Multi-portfolio optimization: utility versus the number of iterations
Figure 5. Operation flow and signaling exchange between local processor \( p \) and the central processor.

**Signaling exchange.** When \( A \) is too large to be stored and processed by a centralized architecture, 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 \( A \) as

\[
A = [A_1, A_2, \ldots, A_P],
\]

where \( A_p \in \mathbb{R}^{N \times K_p} \) and \( \sum_{p=1}^{P} K_p = K. \) Matrix \( A_p \) is stored and processed in local processor \( p, \) and the following computations can be decomposed among the local processors:

\[
\begin{align*}
Ax &= \sum_{p=1}^{P} A_p x_p, \quad (47a) \\
A^T(Ax - b) &= (A^T P (Ax - b))_{p=1}^{P}, \quad (47b) \\
d(A^T A) &= (d(A^T P A))_{p=1}^{P}. \quad (47c)
\end{align*}
\]

where \( x_p \in \mathbb{R}^{K_p}. \) The decomposition in (47) enables us to discuss the signaling exchange between local processor \( p \) and the central processor involved in (44) and (45).

The signaling exchange is summarized in Fig. 5. Figurally, the central processor sends \( Ax^t - b \) to each local processor \( p \) (S1.1), and each local processor first computes \( A^T_p (Ax^t - b) \) and then sends it back to the central processor (S1.2), which forms \( A^T (Ax^t - b) \) (S1.3) as in (47b) and calculates \( r(x^t) \) as in (43) (S1.4) and then \( b x^t \) as in (44) (S1.5). Then the central processor sends \( b x^t - x^t_p \) to each processor \( p \) (S2.1), and each processor first computes \( A_p (b x^t - x^t_p) \) and then sends it back to the central processor (S2.2), which forms \( A (b x^t - x^t) \) (S2.3) as in (47a), calculates \( \gamma^t \) as in (45) (S2.4), and updates \( x^{t+1} \) (S3.1) and \( Ax^{t+1} - b \) (S3.2) according to (46). From Fig. 5 we can see clearly that the exact line search (45) does not incur any additional signaling compared with that of predetermined stepsizes (e.g., constant stepsize and decreasing stepsize), because the signaling exchange in S2.1-S2.2 has to be carried out anyway to compute \( Ax^{t+1} - b \) in S3.2, cf. (46).

Remark 4. Updates (44) and (45) 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 observation remains same: the proposed exact line search (45) does not incur additional signaling exchange compared with predetermined stepsizes.

To illustrate the advantage of the proposed successive line search (22), we customize it to the LASSO problem (40). After some straightforward calculations, it is possible to show that \( m_t \) can also be calculated according to a closed-form expression, namely

\[
m_t = \log \left( \frac{\gamma^t (A^T b)^T (A^T b)}{\gamma^t (A^T A)^T (A^T A)} \right).
\]

Since the signaling needs to be exchanged only once, it outperforms state-of-the-art techniques (e.g. [24, Rule A'], [17, Remark 4] and [25, Algorithm 2.1]) in which the whole nonsensive objective function in (40) must be repeatedly evaluated (for different values of \( m \)) and compared with a certain benchmark until \( m_t \) is found.

**Simulations.** We first compare in Fig. 6 the proposed algorithm STELA with FLEXA [17] in terms of error \( e(x^t) \):

\[
e(x^t) \triangleq \| \nabla f(x^t) - [\nabla f(x^t) - x^\star]^{\mu_1} \|_2.
\]

Note that \( x^\star \) is a solution of (40) if and only if \( e(x^\star) = 0 \) [25]. FLEXA is implemented as same as [17] except that all elements are updated in parallel and the selective update scheme is not employed, because it cannot overcome the bottleneck of the decreasing stepsize. We also remark that the stepsize rule for FLEXA is

\[
\gamma^{t+1} = \gamma^t \left( 1 - \min(1, 10^{-4}/(e(x^t))d^t) \right),
\]

where \( d \) is the decreasing rate and \( \gamma^0 = 0.9. \) The code and data generating the figure can be downloaded online [35].

Note that the error \( e(x^t) \) plotted in Fig. 6 needs not monotonically decrease (but \( f(x^t) + g(x^t) \) does) 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. Similar behavior is also observed for $d = 10^{-3}$, until the stepsize becomes small enough. When the stepsize is quickly decreasing ($d = 10^{-1}$), although improvement is made in all iterations, the asymptotic convergence speed is slow because the stepsize is too small to make notable improvement. For this example, $d = 10^{-2}$ works well, but the value of a good decreasing rate is parameters dependent (e.g., $A$, $b$ and $\mu$) and no general rule works well for all choices of parameters. By comparison, the proposed algorithm STELA is fast to converge and robust w.r.t. the choice of parameters.

We also compare in Fig. 7 the proposed algorithm STELA with other competitive algorithms in literature: FISTA [32], ADMM [12], GreedyBCD [36] and SpaRSA [37]. We simulated GreedyBCD out of [36] because it is the one that has guaranteed convergence. The dimension of $A$ is 2000 $\times$ 4000 (the left column of Fig. 7) and 5000 $\times$ 10000 (the right column). It is generated first by Matlab command randn and then each row is normalized to unit norm. The density of $x_{\text{true}}$ is 0.1 (the upper row of Fig. 7), 0.2 (the middle row) and 0.4 (the lower row), and $b$ is generated by $b = Ax_{\text{true}} + e$ while $e$ is a Gaussian white vector with variance $10^{-4}$. The regularization gain $\mu$ is set as $\mu = 0.1 \frac{\|A^Tb\|_\infty}{\mu}$, which allows $x_{\text{true}}$ to be recovered to a high accuracy [37].

The simulations are run 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 [35]. The comparison is in terms of CPU time either a given error bound $e(x^t) \leq 10^{-6}$ is reached or the maximum number of iterations, namely, 2000, is reached. The running time consists of both the initialization stage (represented by a flat curve) and the formal stage. For example, in the proposed algorithm STELA, $d(A^TA)$ is computed in the initialization stage. The simulation results are averaged over 20 instances.

We see from Fig. 7 that the proposed algorithm STELA converges faster than the others. Some more comments follow.

- The proposed algorithm STELA is not sensitive to the density of the true signal $x_{\text{true}}$. When the density is increased from 0.1 (left column) to 0.2 (middle column) and then to 0.4 (right column), the CPU time is increased negligibly.
- The proposed algorithm STELA is relatively scalable w.r.t. the problem dimension. When the dimension of $A$ is increased from $2000 \times 4000$ (the left column) to $5000 \times 10000$ (the right column), the CPU time is only marginally increased.
- The initialization stage of ADMM is time consuming because of some expensive matrix operations, e.g., $AA^T$, $(I + \frac{1}{c}AA^T)^{-1}$ and $A^T(I + \frac{1}{c}AA^T)^{-1}A$ ($c$ is a given positive constant). More details can be found in [12, Sec. 6.4]. Furthermore, the CPU time of initialization stage of ADMM is increased dramatically when the dimension of $A$ is increased from $2000 \times 4000$ to $5000 \times 10000$.
- SpaRSA works better when the density of $x_{\text{true}}$ is smaller, e.g., 0.1, than when it is large, e.g., 0.2 and 0.4.
- The asymptotic convergence speed of GreedyBCD is slow, because only one variable is updated in each iteration.

**VI. Concluding Remarks**

In this paper, we have proposed a novel iterative algorithm based on convex approximation. The only requirement on the approximate function is that it is pseudo-convex. On one hand, the relaxation on the assumptions of 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 if it is violated, the algorithm may not converge. On 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 the algorithm can be further accelerated. The advantages and benefits of the proposed algorithm have been demonstrated using several applications in communication networks, signal processing and financial engineering, and they are also numerically consolidated.

**APPENDIX A**

**Proof of Proposition 1**

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

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

Using Assumption (A2), we get

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

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

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

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

**Figure 7.** Time versus error of different algorithms for LASSO. In the left and right column, the dimension of $A$ is 2000 $\times$ 4000 and 5000 $\times$ 10000, respectively. In the higher, middle and lower column, the density of $x_{\text{true}}$ is 0.1, 0.2 and 0.4.
Secondly, suppose \( y \in S(y) \). We readily get
\[
\nabla f(y)^T(x - y) = \nabla \hat{f}(y; y)^T(x - y) \geq 0, \quad \forall x \in X, \quad (49)
\]
where the equality and inequality comes from Assumption (A2) and minimum principle, respectively, so \( y \) is a stationary point of (1).

ii) From the definition of \( \mathbb{B}x^t \), it is either
\[
\hat{f}(\mathbb{B}x^t; x^t) = \hat{f}(x^t; x^t), \quad (50a)
\]
or
\[
\hat{f}(\mathbb{B}x^t; x^t) < \hat{f}(x^t; x^t), \quad (50b)
\]
If (50a) is true, then \( x^t \in S(x^t) \) and, as we have just shown, it is a stationary point of (1). So only (50b) can be true. We have from the pseudo-convexity of \( \hat{f}(x; x^t) \) (cf. Assumption (A1)) and (50b) that \( \mathbb{B}x^t \neq x^t \) and
\[
\nabla \hat{f}(x^t; x^t)^T(\mathbb{B}x^t - x^t) = \nabla f(x^t)^T(\mathbb{B}x^t - x^t) < 0, \quad (51)
\]
where the equality comes from Assumption (A2).

**APPENDIX B**

**PROOF OF THEOREM 2**

*Proof:* Since \( \mathbb{B}x^t \) is the optimal variable of (8), it satisfies the first-order optimality condition:
\[
\nabla \hat{f}(\mathbb{B}x^t; x^t)^T(x - \mathbb{B}x^t) \geq 0, \quad \forall x \in X. \quad (52)
\]

If (50a) is true, then \( x^t \in S(x^t) \) and it is a stationary point of (1) according to Proposition 1 i). Besides, it follows from (49) (with \( x = \mathbb{B}x^t \) and \( y = x^t \)) that \( \nabla f(x^t)^T(\mathbb{B}x^t - x^t) \geq 0 \). Note that equality is actually achieved, i.e.,
\[
\nabla f(x^t)^T(\mathbb{B}x^t - x^t) = 0
\]
because otherwise \( \mathbb{B}x^t - x^t \) would be an ascent direction of \( f(x; x^t) \) at \( x = x^t \) and the definition of \( \mathbb{B}x^t \) would be contradicted. Then from the definition of the successive line search, we can readily infer
\[
f(x^{t+1}) \leq f(x^t). \quad (53)
\]
It is easy to see (53) holds for the exact line search as well.

If (50b) is true, \( x^t \) is not a stationary point and \( \mathbb{B}x^t - x^t \) is a strict descent direction of \( f(x) \) at \( x = x^t \) according to Proposition 1 ii): \( f(x) \) is strictly decreased compared with \( f(x^t) \) if \( x \) is updated at \( x^t \) along the direction \( \mathbb{B}x^t - x^t \). From the definition of the successive line search, there always exists a \( \gamma^t \) such that \( 0 < \gamma^t \leq 1 \) and
\[
f(x^{t+1}) = f(x^t + \gamma^t(\mathbb{B}x^t - x^t)) < f(x^t). \quad (54)
\]
This strict decreasing property also holds for the exact line search because it is the stepsize that gives the largest decrease, which is certainly larger than the successive line search.

We know from (53) and (54) that \( \{f(x^t)\} \) is a monotonically decreasing sequence, so it converges, but not to a local maximum. Besides, for any two (possibly different) convergent subsequences \( \{x^t\}_{t \in T_1} \) and \( \{x^t\}_{t \in T_2} \), the following holds:
\[
\lim_{t \to \infty} f(x^t) = \lim_{T_1 \ni t \to \infty} f(x^t) = \lim_{T_2 \ni t \to \infty} f(x^t).
\]

Since \( f(x) \) is a continuous function, we infer from the preceding equation that
\[
f \left( \lim_{T_1 \ni t \to \infty} x^t \right) = f \left( \lim_{T_2 \ni t \to \infty} x^t \right). \quad (55)
\]
Now consider any convergent subsequence \( \{x^t\}_{t \in T} \) with limit point \( y \), i.e., \( \lim_{T \ni t \to \infty} x^t = y \). To show that \( y \) is a stationary point, we first assume the contrary: \( y \) is not a stationary point. Since \( \hat{f}(x; x^t) \) is continuous in both \( x \) and \( x^t \) by Assumption (A2) and \( \{\mathbb{B}x^t\}_{t \in T} \) is bounded by Assumption (A4), it follows from [23, Th. 1] that there exists a sequence \( \{\mathbb{B}x^t\}_{t \in T_s} \) with \( T_s \subseteq T \) such that it converges and \( \lim_{T \ni t \to \infty} \mathbb{B}x^t \in S(y) \). Since both \( f(x) \) and \( \nabla f(x) \) are continuous, applying [23, Th. 1] again implies there is \( T_s' \) such that \( T_s' \subseteq T_s(\subseteq T) \) and \( \{x^{t+1}\}_{t \in T_s'} \) converges to \( y' \) defined as:
\[
y' = y + \rho(\mathbb{B}y - y),
\]
where \( \rho \) is the stepsize when either the exact or successive line search is applied to \( f(y) \) along the direction \( \mathbb{B}y - y \).
Since \( y \) is not a stationary point, it follows from (54) that \( f(y') < f(y) \), but this would contradict (55). Therefore \( y \) is a stationary point, and the proof is completed.

**APPENDIX C**

**PROOF OF PROPOSITION 3**

*Proof:* For any stationary point \( y \) of (4), it satisfies the first-order optimality condition: for any \( k \) and any \( x_k \in X_k \),
\[
\nabla_k f(y_1, \ldots, y_{k-1}, y_k, y_{k+1}, \ldots, y_K)^T(x_k - y_k) \geq 0. \quad (56)
\]
We first show that \( y \) is a stationary point of (4) if and only if \( y_k = \mathbb{B}_k y \) for all \( k = 1, \ldots, K \).

Firstly, suppose \( y_k = \mathbb{B}_k y \) for any \( k \). Since \( \mathbb{B}_k z^{t-1} \) is the optimal variable of (27) and it satisfies the first-order optimality condition:
\[
\nabla_1 \hat{f}(\mathbb{B}_1 y; y)^T(x_1 - \mathbb{B}_1 y) \geq 0. \quad (57)
\]
Substituting \( y_1 = \mathbb{B}_1 y \) into (57) yields
\[
\nabla_1 \hat{f}(y_1; y)^T(x_1 - y_1) = \nabla_1 f(y)^T(x_1 - y_1) \geq 0.
\]

Repeating this argument for \( k = 2, \ldots, K \) sequentially yields the first-order optimality condition of problem (4) in (56) and \( y \) is thus a stationary point of (4).

Secondly, suppose \( y \) is a stationary point of (4). Assume \( k = 1 \). The following relation holds:
\[
\nabla_1 \hat{f}(y_1; y)^T(x_1 - y_1) = \nabla_1 f(y)^T(x_1 - y_1) \geq 0, \quad \forall x_1 \in X_1, \quad (58)
\]
where the equality comes from Assumption (B2) and the inequality comes from the first-order optimality condition (56). Besides, \( \mathbb{B}_1 y \) satisfies the first-order optimality condition of (27):
\[
\nabla_1 \hat{f}(\mathbb{B}_1 y; y)^T(x_1 - \mathbb{B}_1 y) \geq 0, \quad \forall x_1 \in X_1. \quad (59)
\]
Comparing (58) and (59) and recalling the uniqueness of \( \mathbb{B}_1 y \), it must be \( y_1 = \mathbb{B}_1 y \). Repeating this argument for \( k = 2, \ldots, K \) sequentially, we can conclude that \( y_k = \mathbb{B}_k y \) for all \( k = 1, \ldots, K \).
From (28) it is easy to see that for any \( t \): 
\[
 f(x^{t+1}) \geq f(z^{*}_{t}) \geq \cdots \geq f(z^{*}_{K-1}) \geq f(z^{*}_{K}) = f(x^{*+1}). \tag{60}
\]
Since the lower level set \( \mathcal{L}(x^{0}) \) is bounded, the sequence \( \{x^{t}\} \) is bounded. Consider any convergent subsequence \( \{x^{t_{k}}\}_{k \in T} \) whose limit point is denoted as \( y \). It was showed in [13, Prop. 2.7.1] that \( \{x^{t_{k}+1}\}_{k \in T} \) converges to \( y \) as well. Since \( z^{*}_{k} = (x^{t_{k}+1}, x^{t_{k}+2}, \ldots, x^{t_{K}}) \), we can infer the following:
\[
 \lim_{T \to \infty} z^{t_{k}}_{k} = \lim_{T \to \infty} (x^{t_{k}+1}, x^{t_{k}+2}, \ldots, x^{t_{K}}) = y, \tag{61}
\]
where \( 1 \leq k_{t} \leq K \) and it may be different for different \( t \).

To show \( y \) is a stationary point of (4), we first assume the contrary: \( y \) is a non-stationary point. Then for each \( t \in T \), it is always possible to find a \( k_{t} \) with \( 1 \leq k_{t} \leq K \) such that
\[
 \nabla f(z^{t+1}_{k_{t}+1})^{T}(z^{t}_{k_{t}} - z^{*}_{k_{t}-1}) < 0,
\]
and according to (17), the stepsize in the update from \( z^{t+1}_{k_{t}+1} \) to \( z^{t}_{k_{t}} \) is determined by the successive line search. Therefore, the path from \( z^{t}_{k_{t}} \) to \( z^{t+1}_{k_{t}+1} \) consists of a finite number of updates (larger than 1 and smaller than \( 2K - 1 \), (at least) one of which is based on descent direction and the successive line search and the others do not increase the value of the objective function, cf. (60). [13, Prop. 1.2.6] indicates that the line of analysis of Theorem 2 still holds: the limit point of \( \{z^{t}_{k_{t}}\}_{t \in T} \) must be a stationary point of (4), but this contradicts the assumption that \( y \) is a non-stationary point. The proof is thus completed.

**REFERENCES**


