KU LEUVEN

STADIUS
Center for Dynamical Systems, Signal Processing and Data Analytics

Citation/Reference	Quoc Tran Dinh, Carlo Savorgnan, Moritz Diehl (2013), **Combining Lagrangian decomposition and excessive gap smoothing technique for solving large-scale separable convex optimization problems** Comput. Optim. Appl., 55, 75-111
Archived version	Final publisher's version / pdf
Published version	insert link to the published version of your paper http://dx.doi.org/10.1007/s10589-012-9515-6
Journal homepage	
Author contact	quoc.trandinh@epfl.ch
IR	url in Lirias https://lirias.kuleuven.be/handle/123456789/372730

(article begins on next page)

Comput Optim Appl (2013) 55:75–111
DOI 10.1007/s10589-012-9515-6

Combining Lagrangian decomposition and excessive gap smoothing technique for solving large-scale separable convex optimization problems

Quoc Tran Dinh · Carlo Savorgnan · Moritz Diehl

Received: 22 June 2011 / Published online: 18 November 2012
© Springer Science+Business Media New York 2012

Abstract A new algorithm for solving large-scale convex optimization problems with a separable objective function is proposed. The basic idea is to combine three techniques: Lagrangian dual decomposition, excessive gap and smoothing. The main advantage of this algorithm is that it automatically and simultaneously updates the smoothness parameters which significantly improves its performance. The convergence of the algorithm is proved under weak conditions imposed on the original problem. The rate of convergence is $O(\frac{1}{k})$, where k is the iteration counter. In the second part of the paper, the proposed algorithm is coupled with a dual scheme to construct a switching variant in a dual decomposition framework. We discuss implementation issues and make a theoretical comparison. Numerical examples confirm the theoretical results.

Keywords Excessive gap · Smoothing technique · Lagrangian decomposition · Proximal mappings · Large-scale problem · Separable convex optimization · Distributed optimization

Q. Tran Dinh (✉) · C. Savorgnan · M. Diehl
Department of Electrical Engineering (ESAT-SCD) and Optimization in Engineering Center (OPTEC), KU Leuven, Kasteelpark Arenberg 10, 3001 Heverlee-Leuven, Belgium
e-mail: quoc.trandinh@esat.kuleuven.be

C. Savorgnan
e-mail: carlo.savorgnan@esat.kuleuven.be

M. Diehl
e-mail: moritz.diehl@esat.kuleuven.be

Q. Tran Dinh
Vietnam National University, Hanoi, Vietnam

1 Introduction

Large-scale convex optimization problems appear in many areas of science such as graph theory, networks, transportation, distributed model predictive control, distributed estimation and multistage stochastic optimization [16, 20, 30, 35–37, 39]. Solving large-scale optimization problems is still a challenge in many applications [4]. Over the years, thanks to the development of parallel and distributed computer systems, the chances for solving large-scale problems have been increased. However, methods and algorithms for solving this type of problems are limited [1, 4].

Convex minimization problems with a separable objective function form a class of problems which is relevant in many applications. This class of problems is also known as separable convex minimization problems, see, e.g. [1]. Without loss of generality, a separable convex optimization problem can be written in the form of a convex program with separable objective function and coupled linear constraints [1]. In addition, decoupling convex constraints may also be considered. Mathematically, this problem can be formulated in the following form:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \phi(x) := \sum_{i=1}^{M} \phi_i(x_i) \\ &\text{s.t.} \quad x_i \in X_i, \quad i = 1, \ldots, M, \\ &\qquad \sum_{i=1}^{M} A_i x_i = b, \end{aligned} \tag{1}$$

where $\phi_i : \mathbb{R}^{n_i} \to \mathbb{R}$ is convex, $X_i \in \mathbb{R}^{n_i}$ is a nonempty, closed convex set, $A_i \in \mathbb{R}^{m\times n_i}$, $b \in \mathbb{R}^m$ for all $i = 1, \ldots, M$, and $n_1 + n_2 + \cdots + n_M = n$. The last constraint is called *coupling linear constraint*.

In the literature, several solution approaches have been proposed for solving problem (1). For example, (augmented) Lagrangian relaxation and subgradient methods of multipliers [1, 10, 29, 36], Fenchel's dual decomposition [11], alternating direction methods [2, 9, 13, 15], proximal point-type methods [3, 33], spitting methods [7, 8], interior point methods [18, 32, 39], mean value cross decomposition [17] and partial inverse method [31] have been studied among many others. One of the classical approaches for solving (1) is Lagrangian dual decomposition. The main idea of this approach is to solve the dual problem by means of a subgradient method. It has been recognized in practice that subgradient methods are usually slow [25] and numerically sensitive to the choice of step sizes. In the special case of a strongly convex objective function, the dual function is differentiable. Consequently, gradient schemes can be applied to solve the dual problem.

Recently, Nesterov [25] developed smoothing techniques for solving nonsmooth convex optimization problems based on the fast gradient scheme which was introduced in his early work [24]. The fast gradient schemes have been used in numerous applications including image processing, compressed sensing, networks and system identification, see e.g. [9, 12, 28]. Exploiting Nesterov's idea in [26], Necoara and Suykens [22] applied smoothing technique to the dual problem in the framework of

Lagrangian dual decomposition and then used Nesterov's fast gradient scheme to maximize the smoothed function of the dual problem. This resulted in a new variant of dual decomposition algorithms for solving separable convex optimization. The authors proved that the rate of convergence of their algorithm is $O(\frac{1}{k})$ which is much better than $O(\frac{1}{\sqrt{k}})$ in the subgradient methods of multipliers [6, 23], where k is the iteration counter. A main disadvantage of this scheme is that the smoothness parameter requires to be given *a priori*. Moreover, this parameter crucially depends on a given desired accuracy. Since the Lipschitz constant of the gradient of the objective function in the dual problem is inversely proportional to the smoothness parameter, the algorithm usually generates short steps towards a solution of the dual problem although the rate of convergence is $O(\frac{1}{k})$.

To overcome this drawback, in this paper, we propose a new algorithm which combines three techniques: smoothing [26, 27], excessive gap [27] and Lagrangian dual decomposition [1] techniques. Although the convergence rate is still $O(\frac{1}{k})$, the algorithms developed in this paper have some advantages compared to the one in [22]. First, instead of fixing the smoothness parameters, we update them dynamically at every iteration. Second, our algorithm is a primal-dual method which not only gives us a dual approximate solution but also a primal approximate solution of (1). Note that the computational cost of the proposed algorithms remains almost the same as in the proximal-center-based decomposition algorithm proposed in [22, Algorithm 3.2]. (Algorithm 3.2 in [22] requires one to compute an additional dual step.) This algorithm is called *dual decomposition with two primal steps* (Algorithm 1). Alternatively, we apply the switching strategy of [27] to obtain a decomposition algorithm with switching primal-dual steps for solving problem (1). This algorithm differs from the one in [27] at two points. First, the smoothness parameter is dynamically updated with an exact formula. Second, proximal-based mappings are used to handle the nonsmoothness of the objective function. The second point is more significant since, in practice, estimating the Lipschitz constants is not an easy task even if the objective function is differentiable. We notice that the proximal-based mappings proposed in this paper only play a role for handling the nonsmoothness of the objective function. Therefore, the algorithms developed in this paper do not belong to any proximal-point algorithm class considered in the literature. The approach presented in the present paper is different from splitting methods and alternating methods considered in the literature, see, e.g. [2, 7, 13, 15] in the sense that it solves the convex subproblems of each component simultaneously without transforming the original problem to any equivalent form. Moreover, all algorithms are first order methods which can be implemented in a highly parallel and distributed manner.

Contribution The contribution of this paper is the following:

1. We apply the Lagrangian relaxation, smoothing and excessive gap techniques to large-scale separable convex optimization problems which are not necessarily smooth. Note that the excessive gap condition that we use in this paper is different from the one in [27], where not only the duality gap is measured but also the feasibility gap is used in the framework of constrained optimization, see Lemma 3.
2. We propose two algorithms for solving general separable convex optimization problems. The first algorithm is new, while the second one is a new variant of the

first algorithm proposed by Nesterov in [27, Algorithm 1] applied to Lagrangian dual decomposition. Both algorithms allow us to obtain the primal and dual approximate solutions simultaneously. Moreover, all the algorithm parameters are updated automatically without any tuning procedure. A special case of these algorithms, a new method for solving problem (1) with a strongly convex objective function is studied. All the algorithms are highly parallelizable and distributed.

3. The convergence of the algorithms is proved and the convergence rate is estimated. In the two first algorithms, this convergence rate is $O(\frac{1}{k})$ which is much higher than $O(\frac{1}{\sqrt{k}})$ in subgradient methods [6, 23], where k is the iteration counter. In the last algorithm, the convergence rate is $O(\frac{1}{k^2})$.

The rest of the paper is organized as follows. In the next section, we briefly describe the Lagrangian dual decomposition method [1] for separable convex optimization, the smoothing technique via prox-functions as well as excessive gap techniques [27]. We also provide several technical lemmas which will be used in the sequel. Section 3 presents a new algorithm called *decomposition algorithm with two primal steps* and estimates its worst-case complexity. Section 4 is a combination of the two primal steps and the two dual steps schemes which we call *decomposition algorithm with switching primal-dual steps*. Section 5 is an application of the two dual steps scheme (53) to solve problem (1) with a strongly convex objective function. We also discuss the implementation issues of the proposed algorithms and a theoretical comparison of Algorithms 1 and 2 in Sect. 6. Numerical examples are presented in Sect. 7 to examine the performance of the proposed algorithms and to compare different methods.

Notation Throughout the paper, we shall consider the Euclidean space $\mathbb{R}^n$ endowed with an inner product $x^T y$ for $x, y \in \mathbb{R}^n$ and the norm $\|x\| := \sqrt{x^T x}$. The notation $x := (x_1, \ldots, x_M)$ represents a column vector in $\mathbb{R}^n$, where x_i is a subvector in $\mathbb{R}^{n_i}$, $i = 1, \ldots, M$ and $n_1 + \cdots + n_M = n$.

2 Lagrangian dual decomposition and excessive gap smoothing technique

A classical technique to address coupling constraints in optimization is Lagrangian relaxation [1]. However, this technique often leads to a nonsmooth optimization problem in the dual form. To overcome this situation, we combine the Lagrangian dual decomposition and smoothing technique in [26, 27] to obtain a smoothly approximate dual problem.

For simplicity of discussion, we consider problem (1) with $M = 2$. However, the methods presented in the next sections can be directly applied to the case $M > 2$ (see Sect. 6). Problem (1) with $M = 2$ can be rewritten as follows:

$$\phi^* := \begin{cases} \min\limits_{x:=(x_1,x_2)} \phi(x) := \phi_1(x_1) + \phi_2(x_2) \\ \text{s.t.} \quad A_1x_1 + A_2x_2 = b, \\ \qquad x \in X_1 \times X_2 := X, \end{cases} \tag{2}$$

where ϕ_i, X_i and A_i are defined as in (1) for $i = 1, 2$ and $b \in \mathbb{R}^m$. Problem (2) is said to satisfy the Slater constraint qualification condition if $\mathrm{ri}(X) \cap \{x = (x_1, x_2) \mid A_1x_1 + A_2x_2 = b\} \neq \emptyset$, where $\mathrm{ri}(X)$ is the relative interior of the convex set X. Let us denote by X^* the solution set of this problem. We make the following assumption.

Assumption 1 *The solution set X^* is nonempty and either the Slater qualification condition for problem (2) holds or X_i is polyhedral. The function ϕ_i is proper, lower semicontinuous and convex in $\mathbb{R}^n$, $i = 1, 2$.*

Note that the objective function ϕ is not necessarily smooth. For example, $\phi(x) = \|x\|_1 = \sum_{i=1}^n |x_{(i)}|$, which is nonsmooth and separable.

2.1 Decomposition via Lagrangian relaxation

Let us first define the Lagrange function of problem (2) as:

$$\mathcal{L}(x, y) := \phi_1(x_1) + \phi_2(x_2) + y^T(A_1x_1 + A_2x_2 - b), \tag{3}$$

where $y \in \mathbb{R}^m$ is the multiplier associated with the coupling constraint $A_1x_1 + A_2x_2 = b$. Then, the dual problem of (2) can be written as:

$$d^* := \max_{y \in \mathbb{R}^m} d(y), \tag{4}$$

where

$$d(y) := \min_{x \in X} \big\{ \mathcal{L}(x, y) := \phi_1(x_1) + \phi_2(x_2) + y^T(A_1x_1 + A_2x_2 - b) \big\}, \tag{5}$$

is the dual function.

Let $A = [A_1, A_2]$. Due to Assumption 1, *strong duality* holds and we have:

$$d^* = \max_{y \in \mathbb{R}^m} d(y) \overset{\text{strong duality}}{=} \min_{x \in X} \big\{ \phi(x) \mid Ax = b \big\} = \phi^*. \tag{6}$$

Let us denote by Y^* the solution set of the dual problem (4). It is well known that Y^* is bounded due to Assumption 1.

Finally, we note that the dual function d defined by (5) can be computed separately as:

$$d(y) = d_1(y) + d_2(y), \tag{7}$$

where

$$d_i(y) := \min_{x_i \in X_i} \big\{ \phi_i(x_i) + y^T A_i x_i \big\} - \frac{1}{2} b^T y, \quad i = 1, 2. \tag{8}$$

We denote by $x_i^*(y)$ a solution of the minimization problem in (8) ($i = 1, 2$) and $x^*(y) := (x_1^*(y), x_2^*(y))$. The representation (7)–(8) is called a *dual decomposition* of the dual function d. It is obvious that, in general, the dual function d is convex and nonsmooth.

2.2 Smoothing via prox-functions

Let us recall the definition of a proximity function. A function p_X is called a proximity function (prox-function) of a given nonempty, closed and convex set $X \subset \mathbb{R}^{n_x}$ if p_X is continuous, strongly convex with a convexity parameter $\sigma_X > 0$ and $X \subseteq \mathrm{dom}(p_X)$. Let x^c be the prox-center of X which is defined as:

$$x^c = \arg\min_{x \in X} p_X(x). \tag{9}$$

Without loss of generality, we can assume that $p_X(x^c) = 0$. Otherwise, we consider the function $\hat{p}_X(x) := p_X(x) - p_X(x^c)$. Let:

$$D_X := \max_{x \in X} p_X(x) \geq 0. \tag{10}$$

We make the following assumption.

Assumption 2 *Each feasible set X_i is endowed with a prox-function p_i which has a convexity parameter $\sigma_i > 0$. Moreover, $0 \leq D_i := \max_{x_i \in X_i} p_i(x_i) < +\infty$ for $i = 1, 2$.*

Particularly, if X_i is bounded then Assumption 2 is satisfied. Throughout the paper, we assume that Assumptions 1 and 2 are satisfied.

Now, we consider the following functions:

$$d_i(y; \beta_1) := \min_{x_i \in X_i} \left\{ \phi_i(x_i) + y^T A_i x_i + \beta_1 p_i(x_i) \right\} - \frac{1}{2} b^T y, \quad i = 1, 2, \tag{11}$$

$$d(y; \beta_1) := d_1(y; \beta_1) + d_2(y; \beta_1). \tag{12}$$

Here, $\beta_1 > 0$ is a given parameter called *smoothness parameter*. We denote by $x_i^*(y; \beta_1)$ the solution of (11), i.e.:

$$x_i^*(y; \beta_1) := \arg\min_{x_i \in X_i} \left\{ \phi_i(x_i) + y^T A_i x_i + \beta_1 p_i(x_i) - \frac{1}{2} b^T y \right\}, \quad i = 1, 2. \tag{13}$$

Note that we can use different parameters β_1^i for (11) ($i = 1, 2$).

The following lemma shows the main properties of $d(\cdot; \beta_1)$, whose proof can be found, e.g., in [22, 27].

Lemma 1 *For any $\beta_1 > 0$, the function $d_i(\cdot; \beta_1)$ defined by (11) is well-defined, concave and continuously differentiable on $\mathbb{R}^m$. The gradient $\nabla_y d_i(y; \beta_1) = A_i x_i^*(y; \beta_1) - \frac{1}{2} b$ is Lipschitz continuous with a Lipschitz constant $L_i^d(\beta_1) = \frac{\|A_i\|^2}{\beta_1 \sigma_i}$ ($i = 1, 2$).*

Consequently, the function $d(\cdot; \beta_1)$ defined by (12) is concave and differentiable. Its gradient is given by $\nabla d_y(y; \beta_1) := Ax^*(y; \beta_1) - b$ which is Lipschitz continuous with a Lipschitz constant $L^d(\beta_1) := \frac{1}{\beta_1} \sum_{i=1}^2 \frac{\|A_i\|^2}{\sigma_i}$. Moreover, it holds that:

$$d(y; \beta_1) - \beta_1(D_1 + D_2) \leq d(y) \leq d(y; \beta_1), \tag{14}$$

and $d(y; \beta_1) \to d(y)$ as $\beta_1 \downarrow 0^+$ for any $y \in \mathbb{R}^m$.

Remark 1 Even without the boundedness of X, if the solution set X^* of (2) is bounded then, in principle, we can bound the feasible set X by a large compact set which contains all the sampling points generated by the algorithms (see Sect. 4 below). However, in the following algorithms we do not use D_i, $i = 1, 2$ (defined by (10)) in any computational step. They only appear in the theoretical complexity estimates.

Next, for a given $\beta_2 > 0$, we define a mapping $\psi(\cdot;\beta_2)$ from X to $\mathbb{R}$ by:

$$\psi(x;\beta_2) := \max_{y\in\mathbb{R}^m}\left\{(Ax-b)^T y - \frac{\beta_2}{2}\|y\|^2\right\}. \tag{15}$$

This function can be considered as a smoothed version of $\psi(x) := \max_{y\in\mathbb{R}^m}\{(Ax - b)^T y\}$ via the prox-function $p(y) := \frac{1}{2}\|y\|^2$. It is easy to show that the unique solution of the maximization problem in (15) is given explicitly as $y^*(x;\beta_2) = \frac{1}{\beta_2}(Ax - b)$ and $\psi(x;\beta_2) = \frac{1}{2\beta_2}\|Ax - b\|^2$. Therefore, $\psi(\cdot;\beta_2)$ is well-defined and differentiable on X. Let:

$$f(x;\beta_2) := \phi(x) + \psi(x;\beta_2) = \phi(x) + \frac{1}{2\beta_2}\|Ax - b\|^2. \tag{16}$$

The next lemma summarizes the properties of $\psi(\cdot;\beta_2)$ and $f(\cdot;\beta_2)$.

Lemma 2 *For any $\beta_2 > 0$, the function $\psi(\cdot;\beta_2)$ defined by (15) is a quadratic function of the form $\psi(x;\beta_2) = \frac{1}{2\beta_2}\|Ax - b\|^2$ on X. Its gradient vector is given by*:

$$\nabla_x\psi(x;\beta_2) = \frac{1}{\beta_2}A^T(Ax - b), \tag{17}$$

which is Lipschitz continuous with a Lipschitz constant $L^{\psi}(\beta_2) := \frac{1}{\beta_2}(\|A_1\|^2 + \|A_2\|^2)$. Moreover, the following estimate holds for all $x, \hat{x} \in X$:

$$\begin{aligned}\psi(x;\beta_2) \le \psi(\hat{x};\beta_2) + \nabla_{x_1}\psi(\hat{x};\beta_2)^T(x_1 - \hat{x}_1) + \nabla_{x_2}\psi(\hat{x};\beta_2)^T(x_2 - \hat{x}_2)\\ + \frac{L_1^{\psi}(\beta_2)}{2}\|x_1 - \hat{x}_1\|^2 + \frac{L_2^{\psi}(\beta_2)}{2}\|x_2 - \hat{x}_2\|^2,\end{aligned} \tag{18}$$

and

$$f(x;\beta_2) - \frac{1}{2\beta_2}\|Ax - b\|^2 = \phi(x) \le f(x;\beta_2), \tag{19}$$

where $L_1^{\psi}(\beta_2) := \frac{2}{\beta_2}\|A_1\|^2$ and $L_2^{\psi}(\beta_2) := \frac{2}{\beta_2}\|A_2\|^2$.

Proof It is sufficient to only prove (18). Since $\psi(x;\beta_2) = \frac{1}{2\beta_2}\|A_1x_1 + A_2x_2 - b\|^2$, we have:

$$\begin{aligned}&\psi(x;\beta_2) - \psi(\hat{x};\beta_2) - \nabla_x\psi(\hat{x};\beta_2)^T(x - \hat{x})\\ &= \frac{1}{2\beta_2}\left\|A_1(x_1 - \hat{x}_1) + A_2(x_2 - \hat{x}_2)\right\|^2\end{aligned}$$

$$\leq \frac{1}{\beta_2}\|A_1\|^2\|x_1 - \hat{x}_1\|^2 + \frac{1}{\beta_2}\|A_2\|^2\|x_2 - \hat{x}_2\|^2. \tag{20}$$

This inequality is indeed (18). The inequality (19) follows directly from (16). □

2.3 Excessive gap technique

Since the duality gap of the primal and dual problems (2)–(4) is measured by $g(x, y) := \phi(x) - d(y)$, if the gap g is equal to zero for some feasible point (x, y) then this point is an optimal solution of (2)–(4). In this section, we apply a technique called *excessive gap* proposed by Nesterov in [27] to the Lagrangian dual decomposition framework. First, we recall the following definition.

Definition 1 We say that a point $(\bar{x}, \bar{y}) \in X \times \mathbb{R}^m$ satisfies the *excessive gap* condition with respect to two smoothness parameters $\beta_1 > 0$ and $\beta_2 > 0$ if:

$$f(\bar{x}; \beta_2) \leq d(\bar{y}; \beta_1), \tag{21}$$

where $f(\cdot; \beta_2)$ and $d(\cdot; \beta_1)$ are defined by (19) and (12), respectively.

The following lemma provides an upper bound estimate for the duality gap and the feasibility gap of problem (2).

Lemma 3 *Suppose that* $(\bar{x}, \bar{y}) \in X \times \mathbb{R}^m$ *satisfies the excessive gap condition* (21). *Then for any* $y^* \in Y^*$, *we have*:

$$\begin{aligned} -\|y^*\|\|A\bar{x} - b\| \leq \phi(\bar{x}) - d(\bar{y}) &\leq \beta_1(D_1 + D_2) - \frac{1}{2\beta_2}\|A\bar{x} - b\|^2 \\ &\leq \beta_1(D_1 + D_2), \end{aligned} \tag{22}$$

and

$$\|A\bar{x} - b\| \leq \beta_2\left\{\|y^*\| + \left[\|y^*\|^2 + \frac{2\beta_1}{\beta_2}(D_1 + D_2)\right]^{1/2}\right\}. \tag{23}$$

Proof Suppose that $\bar{x}$ and $\bar{y}$ satisfy the condition (21). For a given $y^* \in Y^*$, one has:

$$\begin{aligned} d(\bar{y}) \leq d(y^*) &= \min_{x \in X}\left\{\phi(x) + (Ax - b)^T y^*\right\} \\ &\leq \phi(\bar{x}) + (A\bar{x} - b)^T y^* \leq \phi(\bar{x}) + \|A\bar{x} - b\|\|y^*\|, \end{aligned}$$

which implies the first inequality of (22). By using Lemma 1 and (16) we have:

$$\phi(\bar{x}) - d(\bar{y}) \overset{(14)+(19)}{\leq} f(\bar{x}; \beta_2) - d(\bar{y}; \beta_1) + \beta_1(D_1 + D_2) - \frac{1}{2\beta_2}\|A\bar{x} - b\|^2.$$

Now, by substituting the condition (21) into this inequality, we obtain the second inequality of (22). Let $\eta := \|Ax - b\|$. It follows from (22) that $\eta^2 - 2\beta_2\|y^*\|\eta - 2\beta_1\beta_2(D_1 + D_2) \leq 0$. The estimate (23) follows from this inequality after few simple calculations. □

3 New decomposition algorithm

In this section, we derive an iterative decomposition algorithm for solving (2) based on the excessive gap technique. This method is called a *decomposition algorithm with two primal steps*. The aim is to generate a point $(\bar{x}, \bar{y}) \in X \times \mathbb{R}^m$ at each iteration such that this point maintains the excessive gap condition (21) while the algorithm drives the parameters β_1 and β_2 to zero.

3.1 Finding a starting point

As assumed earlier, the function ϕ_i is convex but not necessarily differentiable. Therefore, we can not use the gradient information of these functions. We consider the following mappings ($i = 1, 2$):

$$P_i(\hat{x}; \beta_2) := \arg\min_{x_i \in X_i} \left\{ \phi_i(x_i) + y^*(\hat{x}; \beta_2)^T A_i (x_i - \hat{x}_i) + \frac{L_i^{\psi}(\beta_2)}{2} \|x_i - \hat{x}_i\|^2 \right\}, \tag{24}$$

where $y^*(\hat{x}; \beta_2) := \frac{1}{\beta_2}(A\hat{x} - b)$. Since $L_i^{\psi}(\beta_2)$ defined in Lemma 2 is positive, $P_i(\cdot; \beta_2)$ is well-defined. This mapping is called a *proximal operator* [3]. Let $P(\cdot; \beta_2) = (P_1(\cdot; \beta_2), P_2(\cdot; \beta_2))$.

First, we show in the following lemma that there exists a point $(\bar{x}, \bar{y})$ satisfying the excessive gap condition (21). The proof of this lemma can be found in the Appendix.

Lemma 4 *Suppose that x^c is the prox-center of X. For a given $\beta_2 > 0$, let*:

$$\bar{y} := \beta_2^{-1}\big(Ax^c - b\big) \quad \textit{and} \quad \bar{x} := P\big(x^c; \beta_2\big). \tag{25}$$

If the parameter β_1 is chosen such that:

$$\beta_1 \beta_2 \geq 2 \max_{1 \leq i \leq 2} \left\{ \frac{\|A_i\|^2}{\sigma_i} \right\}, \tag{26}$$

then $(\bar{x}, \bar{y})$ satisfies the excessive gap condition (21).

3.2 Main iteration scheme

Suppose that $(\bar{x}, \bar{y}) \in X \times \mathbb{R}^m$ satisfies the excessive gap condition (21). We generate a new point $(\bar{x}^+, \bar{y}^+) \in X \times \mathbb{R}^m$ by applying the following update scheme:

$$\big(\bar{x}^+, \bar{y}^+\big) := \mathcal{A}_m^p\big(\bar{x}, \bar{y}; \beta_1, \beta_2^+, \tau\big) \quad \Longleftrightarrow \quad \begin{cases} \hat{x} := (1-\tau)\bar{x} + \tau x^*(\bar{y}; \beta_1), \\ \bar{y}^+ := (1-\tau)\bar{y} + \tau y^*(\hat{x}; \beta_2^+), \\ \bar{x}^+ := P(\hat{x}; \beta_2^+), \end{cases} \tag{27}$$

$$\beta_1^+ := (1-\tau)\beta_1 \quad \text{and} \quad \beta_2^+ = (1-\tau)\beta_2, \tag{28}$$

where $P(\cdot; \beta_2^+) = (P_1(\cdot; \beta_2^+), P_2(\cdot; \beta_2^+))$ and $\tau \in (0, 1)$ will be chosen appropriately.

Remark 2 In the scheme (27), the points $x^*(\bar{y};\beta_1) = (x_1^*(\bar{y};\beta_1), x_2^*(\bar{y};\beta_1))$, $\hat{x} = (\hat{x}_1, \hat{x}_2)$ and $\bar{x}^+ = (\bar{x}_1^+, \bar{x}_2^+)$ can be computed *in parallel*. To compute $x^*(\bar{y};\beta_1)$ and $\bar{x}^+$ we need to solve two corresponding convex programs in $\mathbb{R}^{n_1}$ and $\mathbb{R}^{n_2}$, respectively.

The following theorem shows that the scheme (27)–(28) maintains the excessive gap condition (21).

Theorem 1 *Suppose that* $(\bar{x}, \bar{y}) \in X \times \mathbb{R}^m$ *satisfies* (21) *with respect to two values* $\beta_1 > 0$ *and* $\beta_2 > 0$. *Then if the parameter* τ *is chosen such that* $\tau \in (0,1)$ *and*:

$$\beta_1\beta_2 \geq \frac{2\tau^2}{(1-\tau)^2}\max_{1\leq i\leq 2}\left\{\frac{\|A_i\|^2}{\sigma_i}\right\}, \tag{29}$$

then the new point $(\bar{x}^+, \bar{y}^+)$ *generated by the scheme* (27)–(28) *is in* $X \times \mathbb{R}^m$ *and maintains the excessive gap condition* (21) *with respect to two new values* β_1^+ *and* β_2^+.

Proof The last line of (27) shows that $\bar{x}^+ \in X$. Let us denote by $\hat{y} := y^*(\hat{x};\beta_2^+)$. Then, by using the definition of $d(\cdot;\beta_1)$, the second line of (27) and $\beta_1^+ = (1-\tau)\beta_1$, we have:

$$\begin{aligned} d\big(\bar{y}^+;\beta_1^+\big) &= \min_{x\in X}\big\{\phi(x) + (Ax-b)^T\bar{y}^+ + \beta_1^+\big[p_1(x_1) + p_2(x_2)\big]\big\} \\ &\overset{\text{line 2 (27)}}{=} \min_{x\in X}\big\{\phi(x) + (1-\tau)(Ax-b)^T\bar{y} + \tau(Ax-b)^T\hat{y} \\ &\quad + (1-\tau)\beta_1\big[p_1(x_1) + p_2(x_2)\big]\big\} \\ &= \min_{x\in X}\big\{(1-\tau)\big[\phi(x) + (Ax-b)^T\bar{y} + \beta_1\big[p_1(x_1) + p_2(x_2)\big]\big]_{[1]} \\ &\quad + \tau\big[\phi(x) + (Ax-b)^T\hat{y}\big]_{[2]}\big\}. \end{aligned} \tag{30}$$

Now, we estimate the first term $[\cdot]_1$ in the last line of (30). Since $\beta_2^+ = (1-\tau)\beta_2$, one has:

$$\psi(\bar{x};\beta_2) = \frac{1}{2\beta_2}\|A\bar{x}-b\|^2 = (1-\tau)\frac{1}{2\beta_2^+}\|A\bar{x}-b\|^2 = (1-\tau)\psi\big(\bar{x};\beta_2^+\big). \tag{31}$$

Moreover, if we denote by $x^1 := x^*(\bar{y};\beta_1)$ then, by the strong convexity of p_1 and p_2, (31) and $f(\bar{x};\beta_2) \leq d(\bar{y};\beta_1)$, we have:

$$\begin{aligned} [\cdot]_1 &= \phi(x) + (Ax-b)^T\bar{y} + \beta_1\big[p_1(x_1) + p_2(x_2)\big] \\ &\geq \min_{x\in X}\big\{\phi(x) + (Ax-b)^T\bar{y} + \beta_1\big[p_1(x_1) + p_2(x_2)\big]\big\} \\ &\quad + \frac{1}{2}\beta_1\big[\sigma_1\|x_1 - x_1^1\|^2 + \sigma_2\|x_2 - x_2^1\|^2\big] \\ &= d(\bar{y};\beta_1) + \frac{1}{2}\beta_1\big[\sigma_1\|x_1 - x_1^1\|^2 + \sigma_2\|x_2 - x_2^1\|^2\big] \end{aligned}$$

$$
\begin{aligned}
&\overset{(21)}{\geq} && f(\bar{x};\beta_2)+\frac{1}{2}\beta_1\left[\sigma_1\left\|x_1-x_1^1\right\|^2+\sigma_2\left\|x_2-x_2^1\right\|^2\right]\\
&\overset{\text{def. } f(\cdot;\beta_2)}{=} && \phi(\bar{x})+\psi(\bar{x};\beta_2)+\frac{1}{2}\beta_1\left[\sigma_1\left\|x_1-x_1^1\right\|^2+\sigma_2\left\|x_2-x_2^1\right\|^2\right]\\
&\overset{(31)}{=} && \phi(\bar{x})+\psi\left(\bar{x};\beta_2^+\right)+\frac{1}{2}\beta_1\left[\sigma_1\left\|x_1-x_1^1\right\|^2+\sigma_2\left\|x_2-x_2^1\right\|^2\right]\\
& && -\tau\psi\left(x;\beta_2^+\right)\\
&\overset{(20)}{=} && \phi(\bar{x})+\psi\left(\hat{x};\beta_2^+\right)+\nabla_x\psi\left(\hat{x};\beta_2^+\right)^T(\bar{x}-\hat{x})+\frac{1}{2}\beta_1\left[\sigma_1\left\|x_1-x_1^1\right\|^2\right.\\
& && \left.+\sigma_2\left\|x_2-x_2^1\right\|^2\right]\\
& && +\frac{1}{2\beta_2^+}\left\|A(\bar{x}-\hat{x})\right\|^2-\tau\psi\left(\bar{x};\beta_2^+\right). \qquad (32)
\end{aligned}
$$

For the second term $[\cdot]_2$ in the last line of (30), we use the fact that $\hat{y}=\frac{1}{\beta_2^+}(A\hat{x}-b)$ and $\nabla_x\psi(\hat{x};\beta_2)=A^T\hat{y}$ to obtain:

$$
\begin{aligned}
[\cdot]_2 \quad &:= && \phi(x)+(Ax-b)^T\hat{y}=\phi(x)+\hat{y}^TA(x-\hat{x})+(A\hat{x}-b)^T\hat{y}\\
&\overset{\text{def. } \hat{y}+(17)}{=} && \phi(x)+\nabla_x\psi\left(\hat{x};\beta_2^+\right)^T(x-\hat{x})+\frac{1}{\beta_2^+}\|A\hat{x}-b\|^2\\
&\overset{\text{def. } \hat{\psi}}{=} && \phi(x)+\psi\left(\hat{x};\beta_2^+\right)+\nabla_x\psi\left(\hat{x};\beta_2^+\right)^T(x-\hat{x})+\psi\left(\hat{x};\beta_2^+\right). \qquad (33)
\end{aligned}
$$

Substituting (32) and (33) into (30) and noting that $(1-\tau)(\bar{x}-\hat{x})+\tau(x-\hat{x})=\tau(x-x^1)$ due to the first line of (27), we obtain:

$$
\begin{aligned}
d\left(\bar{y}^+;\beta_1^+\right) \quad &= && \min_{x\in X}\left\{(1-\tau)[\cdot]_1+\tau[\cdot]_2\right\}\\
&\overset{(32)+(33)}{\geq} && \min_{x\in X}\left\{(1-\tau)\left[\phi(\bar{x})+\psi\left(\hat{x};\beta_2^+\right)+\nabla_x\psi\left(\hat{x};\beta_2^+\right)^T(\bar{x}-\hat{x})\right.\right.\\
& && \left.+\frac{1}{2}\beta_1\left[\sigma_1\left\|x_1-x_1^1\right\|^2+\sigma_2\left\|x_2-x_2^1\right\|^2\right]\right]\\
& && \left.+\tau\left[\phi(x)+\psi\left(\hat{x};\beta_2^+\right)+\nabla_x\psi\left(\hat{x};\beta_2^+\right)^T(x-\hat{x})\right]\right\}\\
& && -\tau(1-\tau)\psi\left(\bar{x};\beta_2^+\right)+\frac{(1-\tau)}{2\beta_2^+}\left\|A(\bar{x}-\hat{x})\right\|^2+\tau\psi\left(\hat{x};\beta_2^+\right)\\
&= && \min_{x\in X}\left\{(1-\tau)\phi(\bar{x})+\tau\phi(x)+\psi\left(\hat{x};\beta_2^+\right)\right.\\
& && +\nabla_x\psi\left(\hat{x};\beta_2^+\right)^T\left[(1-\tau)(\bar{x}-\hat{x})+\tau(x-\hat{x})\right]\\
& && \left.+\frac{1}{2}(1-\tau)\beta_1\left[\sigma_1\left\|x_1-x_1^1\right\|^2+\sigma_2\left\|x_2-x_2^1\right\|^2\right]\right\}+\mathbf{T}_1
\end{aligned}
$$

$$\overset{\phi\text{-convex}}{\geq} \min_{x\in X}\Big\{\phi\big((1-\tau)\bar{x}+\tau x\big)+\psi\big(\hat{x};\beta_2^+\big)+\tau\nabla_x\psi\big(\hat{x};\beta_2^+\big)^T\big(x-x^1\big) + \frac{1}{2}(1-\tau)\beta_1\big[\sigma_1\|x_1-x_1^1\|^2+\sigma_2\|x_2-x_2^1\|^2\big]\Big\}+\mathbf{T}_1, \tag{34}$$

where $\mathbf{T}_1 := \frac{(1-\tau)}{2\beta_2^+}\|A(\bar{x}-\hat{x})\|^2+\tau\psi(\hat{x};\beta_2^+)-\tau(1-\tau)\psi(\bar{x};\beta_2^+)$. Next, we note that the condition (29) is equivalent to:

$$(1-\tau)\beta_1\sigma_i \geq \frac{2\tau^2}{(1-\tau)\beta_2}\|A_i\|^2 \geq L_i^{\psi}\big(\beta_2^+\big)\tau^2, \quad i=1,2. \tag{35}$$

Moreover, if we denote by $u := \bar{x}+\tau(x-\bar{x})$ then:

$$u-\hat{x} = \bar{x}+\tau(x-\bar{x})-\hat{x} = \bar{x}+\tau(x-\bar{x})-(1-\tau)\bar{x}-\tau x^1 = \tau\big(x-x^1\big). \tag{36}$$

Now, by using Lemma 2, the condition (35) and (36), the estimation (34) becomes:

$$\begin{aligned}
d\big(\bar{y}^+;\beta_1^+\big)-\mathbf{T}_1 &\overset{(36)}{\geq} \min_{u:=\bar{x}+\tau(x-\bar{x})\in\bar{x}+\tau(X-\bar{x})}\Big\{\phi(u)+\psi\big(\hat{x};\beta_2^+\big) \\
&\quad +\nabla_x\psi(\hat{x};\beta_2)^T(u-\hat{x}) \\
&\quad +\frac{\beta_1(1-\tau)\sigma_1}{2\tau^2}\|u_1-\hat{x}_1\|^2+\frac{\beta_1(1-\tau)\sigma_2}{2\tau^2}\|u_2-\hat{x}_2\|^2\Big\} \\
&\overset{\bar{x}+\tau(X-\bar{x})\subseteq X}{\geq} \min_{u\in X}\Big\{\psi\big(\hat{x};\beta_2^+\big)+\phi(u)+\nabla_x\psi\big(\hat{x};\beta_2^+\big)^T(u-\hat{x}) \\
&\quad +\frac{\beta_1(1-\tau)\sigma_1}{2\tau^2}\|u_1-\hat{x}_1\|^2+\frac{\beta_1(1-\tau)\sigma_2}{2\tau^2}\|u_2-\hat{x}_2\|^2\Big\} \\
&\overset{(35)}{\geq} \min_{u\in X}\Big\{\phi(u)+\psi\big(\hat{x};\beta_2^+\big)+\nabla_x\psi\big(\hat{x};\beta_2^+\big)^T(u-\hat{x}) \\
&\quad +\frac{L_1^{\psi}(\beta_2^+)}{2}\|u_1-\hat{x}_1\|^2+\frac{L_2^{\psi}(\beta_2^+)}{2}\|u_2-\hat{x}_2\|^2\Big\} \\
&\overset{\text{line 3 (27)}}{=} \phi\big(\bar{x}^+\big)+\psi\big(\hat{x};\beta_2^+\big)+\nabla_x\psi\big(\hat{x};\beta_2^+\big)^T\big(\bar{x}^+-\hat{x}\big) \\
&\quad +\frac{L_1^{\psi}(\beta_2^+)}{2}\big\|\bar{x}_1^+-\hat{x}_1\big\|^2+\frac{L_2^{\psi}(\beta_2^+)}{2}\big\|\bar{x}_2^+-\hat{x}_2\big\|^2 \\
&\overset{(18)}{\geq} \phi\big(\bar{x}^+\big)+\psi\big(\bar{x}^+;\beta_2^+\big) = f\big(\bar{x}^+;\beta_2^+\big).
\end{aligned} \tag{37}$$

To complete the proof, we show that $\mathbf{T}_1 \geq 0$. Indeed, let us define $\hat{u} := A\hat{x}-b$ and $\bar{u} := A\bar{x}-b$, then $\hat{u}-\bar{u} = A(\hat{x}-\bar{x})$. We have:

$$\mathbf{T}_1 \overset{\text{def. } \psi(\cdot;\beta_2)}{=} \frac{\tau}{2\beta_2^+}\|A\hat{x}-b\|^2-\frac{\tau(1-\tau)}{2\beta_2^+}\|A\bar{x}-b\|^2+\frac{(1-\tau)}{2\beta_2^+}\big\|A(\hat{x}-\bar{x})\big\|^2$$

$$
\begin{aligned}
&= \frac{1}{2\beta_2^+}\left[\tau\|\hat{u}\|^2 - \tau(1-\tau)\|\bar{u}\|^2 + (1-\tau)\|\hat{u}-\bar{u}\|^2\right] \\
&= \frac{1}{2\beta_2^+}\left[\tau\|\hat{u}\|^2 - \tau(1-\tau)\|\bar{u}\|^2 + (1-\tau)\|\hat{u}\|^2\right. \\
&\quad \left. + (1-\tau)\|\bar{u}\|^2 - 2(1-\tau)\hat{u}^T\bar{u}\right] \\
&= \frac{1}{2\beta_2^+}\left[\|\hat{u}\|^2 + (1-\tau)^2\|\bar{u}\|^2 - 2(1-\tau)\hat{u}^T\bar{u}\right] \\
&= \frac{1}{2\beta_2^+}\left\|\hat{u} - (1-\tau)\bar{u}\right\|^2 \geq 0. \qquad (38)
\end{aligned}
$$

Substituting (38) into (37) we obtain the inequality $d(\bar{y}^+;\beta_1^+) \geq f(\bar{x}^+;\beta_2^+)$. □

Remark 3 If ϕ_i is convex and differentiable such that its gradient is Lipschitz continuous with a Lipschitz constant $L^{\phi_i} \geq 0$ for some $i = 1, 2$, then instead of using the proximal mapping $P_i(\cdot;\beta_2)$ in (27) we can use the following mapping:

$$
\begin{aligned}
G_i\left(\hat{x};\beta_2^+\right) := \arg\min_{x_i \in X_i} \Big\{ & \nabla\phi_i(\hat{x}_i)^T(x_i - \hat{x}_i) \\
& + y^*(\hat{x};\beta_2)^T A_i(x_i - \hat{x}_i) + \frac{\hat{L}_i^{\psi}(\beta_2^+)}{2}\|x_i - \hat{x}_i\|^2 \Big\}, \qquad (39)
\end{aligned}
$$

where $\hat{L}_i^{\psi}(\beta_2^+) := L^{\phi_i} + \frac{2\|A_i\|^2}{\beta_2^+}$. Indeed, let us prove the condition $d(\bar{y}^+;\beta_1^+) \geq f(\hat{\bar{x}}^+;\beta_2^+)$, where $G(x;\beta_2) := (G_1(x_1;\beta_2), G_2(x_2;\beta_2))$ and $\hat{\bar{x}}^+ := G(\hat{x};\beta_2^+)$. First, by using the convexity of ϕ_i and the Lipschitz continuity of its gradient, we have:

$$
\begin{aligned}
&\phi_i(\hat{x}_i) + \nabla\phi_i(\hat{x}_i)^T(u_i - \hat{x}_i) \leq \phi_i(u_i) \\
&\leq \phi_i(\hat{x}_i) + \nabla\phi_i(\hat{x}_i)^T(u_i - \hat{x}_i) + \frac{L^{\phi_i}}{2}\|u_i - \hat{x}_i\|^2. \qquad (40)
\end{aligned}
$$

Next, by summing up the second inequality for $i = 1, 2$ and adding to (18) we have:

$$
\begin{aligned}
\phi(u) + \psi\left(u;\beta_2^+\right) \leq \phi(\hat{x}) + \psi\left(\hat{x};\beta_2^+\right) + \left[\nabla\phi(\hat{x}) + \nabla_x\psi\left(\hat{x};\beta_2^+\right)\right]^T(u - \hat{x}) \\
+ \frac{\hat{L}_1^{\psi}(\beta_2^+)}{2}\|u_1 - \hat{x}_1\|^2 + \frac{\hat{L}_2^{\psi}(\beta_2^+)}{2}\|u_2 - \hat{x}_2\|^2. \qquad (41)
\end{aligned}
$$

Finally, from the second inequality of (37) we have:

$$
\begin{aligned}
d\left(\bar{y}^+;\beta_1^+\right) - \mathbf{T}_1 \overset{(35)}{\geq} & \min_{u\in X}\Big\{\phi(u) + \psi\left(\hat{x};\beta_2^+\right) + \nabla_x\psi\left(\hat{x};\beta_2^+\right)^T(u - \hat{x}) \\
& + \frac{(1-\tau)\beta_1\sigma_1}{2\tau^2}\|u_1 - \hat{x}_1\|^2 \\
& + \frac{(1-\tau)\beta_1\sigma_2}{2\tau^2}\|u_2 - \hat{x}_2\|^2\Big\}
\end{aligned}
$$

$$
\begin{aligned}
&\overset{\phi\text{-convex}+(41)}{\geq} \min_{u\in X}\Big\{\phi(\hat{x}) + \nabla\phi(\hat{x})^T(u-\hat{x}) + \psi\big(\hat{x};\beta_2^+\big)\\
&\qquad + \nabla_x\psi\big(\hat{x};\beta_2^+\big)^T(u-\hat{x})\\
&\qquad + \frac{\hat{L}_1^{\psi}(\beta_2^+)}{2}\|u_1-\hat{x}_1\|^2 + \frac{\hat{L}_2^{\psi}(\beta_2^+)}{2}\|u_2-\hat{x}_2\|^2\Big\}\\
&\overset{(39)}{=} \phi(\hat{x}) + \psi\big(\hat{x};\beta_2^+\big)\\
&\qquad + \big[\nabla\phi(\hat{x}) + \nabla_x\psi\big(\hat{x};\beta_2^+\big)\big]^T\big(\hat{\hat{x}}^+ - \hat{x}\big)\\
&\qquad + \frac{\hat{L}_1^{\psi}(\beta_2^+)}{2}\big\|\hat{\hat{x}}_1^+ - \hat{x}_1\big\|^2 + \frac{\hat{L}_2^{\psi}(\beta_2^+)}{2}\big\|\hat{\hat{x}}_2^+ - \hat{x}_2\big\|^2\\
&\overset{(41)}{\geq} \phi\big(\hat{\hat{x}}^+\big) + \psi\big(\hat{\hat{x}}^+;\beta_2^+\big) = f\big(\hat{\hat{x}}^+;\beta_2^+\big).
\end{aligned}
$$

In this case, the conclusion of Theorem 1 is still valid for the substitution $\hat{\hat{x}}^+ := G(\hat{x};\beta_2^+)$ provided that:

$$
\frac{(1-\tau)}{\tau^2}\beta_1\sigma_i \geq L^{\phi_i} + \frac{2\|A_i\|^2}{(1-\tau)\beta_2}, \quad i = 1, 2. \tag{42}
$$

If X_i is polytopic then problem (39) becomes a convex quadratic program.

3.3 The step size update rule

Next step we show how to update the parameter τ such that the conditions (26) and (29) hold. From the update rule (28) we have $\beta_1^+\beta_2^+ = (1-\tau)^2\beta_1\beta_2$. Suppose that β_1 and β_2 satisfy the condition (29), i.e.:

$$
\beta_1\beta_2 \geq \frac{\tau^2}{(1-\tau)^2}\bar{L}^2, \quad \text{where } \bar{L} := \left[2\max_{1\leq i\leq 2}\left\{\frac{\|A_i\|^2}{\sigma_i}\right\}\right]^{1/2}.
$$

If we substitute β_1 and β_2 by β_1^+ and β_2^+, respectively, in this inequality then we have $\beta_1^+\beta_2^+ \geq \frac{\tau_+^2}{(1-\tau_+)^2}\bar{L}^2$. However, since $\beta_1^+\beta_2^+ = (1-\tau)^2\beta_1\beta_2$, it implies $\beta_1\beta_2 \geq \frac{\tau_+^2}{(1-\tau)^2(1-\tau_+)^2}\bar{L}^2$. Therefore, if $\frac{\tau^2}{(1-\tau)^2} \geq \frac{\tau_+^2}{(1-\tau)^2(1-\tau_+)^2}$ then β_1^+ and β_2^+ satisfy (29). This condition leads to $\tau \geq \frac{\tau_+}{1-\tau_+}$. Since $\tau, \tau_+ \in (0,1)$, the last inequality implies:

$$
0 < \tau_+ \leq \frac{\tau}{\tau+1} < 1. \tag{43}
$$

Hence, (27)–(28) are well-defined. At the first iteration $k = 0$, both conditions (26) and (29) need to be satisfied. This leads to $0 < \tau_0 \leq 0.5$.

Now, we define a rule to update the step size parameter τ.

Lemma 5 *Suppose that* τ_0 *is arbitrarily chosen in* $(0, \frac{1}{2}]$. *Then the sequence* $\{\tau_k\}_{k\geq 0}$ *generated by*:

$$\tau_{k+1} := \frac{\tau_k}{\tau_k + 1} \tag{44}$$

satisfies the following equality:

$$\tau_k = \frac{\tau_0}{1 + \tau_0 k}, \quad \forall k \geq 0. \tag{45}$$

Moreover, the sequence $\{\beta_k\}_{k\geq 0}$ *generated by* $\beta_{k+1} = (1 - \tau_k)\beta_k$ *for fixed* $\beta_0 > 0$ *satisfies*:

$$\beta_k = \frac{\beta_0}{\tau_0 k + 1}, \quad \forall k \geq 0. \tag{46}$$

Proof If we denote by $t := \frac{1}{\tau}$ and consider the function $\xi(t) := t + 1$ then the sequence $\{t_k\}_{k\geq 0}$ generated by the rule $t_{k+1} := \xi(t_k) = t_k + 1$ satisfies $t_k = t_0 + k$ for all $k \geq 0$. Hence $\tau_k = \frac{1}{t_k} = \frac{1}{t_0+k} = \frac{\tau_0}{\tau_0 k+1}$ for $k \geq 0$. To prove (46), we observe that $\beta_{k+1} = \beta_0 \prod_{i=0}^{k}(1 - \tau_i)$. Hence, by substituting (45) into the last equality and carrying out some simple calculations, we get (46). □

Remark 4 Since $\tau_0 \in (0, 0.5]$, from Lemma 5 we see that with $\tau_0 := 0.5$ the right-hand side estimate of (46) is minimized. In this case, the update rule of τ_k is simplified to $\tau_k := \frac{1}{k+2}$ for $k \geq 0$.

3.4 The algorithm and its worst case complexity

Now, we combine the results of Lemma 4, Theorem 1 and Lemma 5 in order to build the following algorithm.

Algorithm 1 (Decomposition algorithm with two primal steps)
Initialization: Perform the following steps:

1. Set $\tau_0 := 0.5$. Choose $\beta_1^0 > 0$ and $\beta_2^0 > 0$ as follows:

$$\beta_1^0 = \beta_2^0 := \left[2 \max_{1\leq i\leq 2}\left\{\frac{\|A_i\|^2}{\sigma_i}\right\}\right]^{1/2}.$$

2. Compute $\bar{x}^0$ and $\bar{y}^0$ from (25) as:

$$\bar{y}^0 := \left(\beta_2^0\right)^{-1}\left(Ax^c - b\right) \quad \text{and} \quad \bar{x}^0 := P\left(x^c; \beta_2^0\right).$$

Iteration: For $k = 0, 1, \ldots,$ perform the following steps:

1. If a given stopping criterion is satisfied then terminate.
2. Update the smoothness parameter $\beta_2^{k+1} := (1 - \tau_k)\beta_2^k$.

3. Compute $\bar{x}_i^{k+1}$ *in parallel* for $i = 1, 2$ and $\bar{y}^{k+1}$ by the scheme (27):

$$\big(\bar{x}^{k+1}, \bar{y}^{k+1}\big) := \mathcal{A}_m^p\big(\bar{x}^k, \bar{y}^k; \beta_1^k, \beta_2^{k+1}, \tau_k\big).$$

4. Update the smoothness parameter: $\beta_1^{k+1} := (1 - \tau_k)\beta_1^k$.
5. Update the step size parameter τ_k by: $\tau_{k+1} := \frac{1}{k+3}$.

End.

As mentioned in Remark 2, there are two steps in the scheme $\mathcal{A}_m^p$ of Algorithm 1 that can be parallelized. The first step is finding $x^*(\bar{y}^k; \beta_1)$ and the second one is computing $\bar{x}^{k+1}$. In general, both steps require solving two convex programming subproblems in parallel. The stopping criterion of Algorithm 1 will be discussed in Sect. 6.

Note that the dual solution set Y^* is compact due to Assumption 1 and the conclusions of Lemma 3 hold for any $y^* \in Y^*$, we can define the following quantities:

$$\begin{aligned} &D_X := D_1 + D_2, \qquad D_{Y^*} := \min_{y^* \in Y^*} \|y^*\| \quad \text{and} \\ &D_Y := \min_{y^* \in Y^*} \Big[\|y^*\| + \sqrt{\|y^*\|^2 + 2(D_1 + D_2)}\Big]. \end{aligned} \tag{47}$$

The next theorem provides the worst-case complexity estimate for Algorithm 1.

Theorem 2 *Let* $\{(\bar{x}^k, \bar{y}^k)\}$ *be a sequence generated by Algorithm* 1. *Then the following duality and feasibility gap estimates hold*:

$$-D_{Y^*}\|A\bar{x}^k - b\| \le \phi\big(\bar{x}^k\big) - d\big(\bar{y}^k\big) \le \frac{2\bar{L}D_X}{k+2}, \tag{48}$$

and

$$\|A\bar{x}^k - b\| \le \frac{2\bar{L}D_Y}{k+2}, \tag{49}$$

where

$$\bar{L} := \left[2\max_{1 \le i \le 2}\left\{\frac{\|A_i\|^2}{\sigma_i}\right\}\right]^{1/2}$$

and D_X, D_{Y^*} *and* D_Y *are defined in* (47).

Proof By the choice of $\beta_1^0 = \beta_2^0 = \bar{L}$ and Steps 1 in the initialization phase of Algorithm 1 we see that $\beta_1^k = \beta_2^k$ for all $k \ge 0$. Moreover, since $\tau_0 = 0.5$, by Lemma 5, we have $\beta_1^k = \beta_2^k = \frac{\beta_0}{\tau_0 k+1} = \frac{\bar{L}}{0.5k+1}$. Now, by applying Lemma 3 with β_1 and β_2 equal to β_1^k and β_2^k respectively, we obtain the estimates (48) and (49). □

Remark 5 The worst case complexity of Algorithm 1 is $O(\frac{1}{\varepsilon})$. However, the constants in the estimations (48) and (49) also depend on the choices of β_1^0 and β_2^0, which

satisfy the condition (26). The values of β_1^0 and β_2^0 will affect the accuracy of the duality and feasibility gaps.

4 Switching decomposition algorithm

In this section, we apply a switching strategy to obtain a new variant of the first algorithm proposed in [27, Algorithm 1] for solving problem (2). This scheme alternately switches between a two primal steps scheme and a two dual steps scheme depending on the iteration counter k being even or odd.

4.1 The gradient mapping of the smoothed dual function

Since the smoothed dual function $d(\cdot;\beta_1)$ is Lipschitz continuously differentiable on $\mathbb{R}^m$ (see Lemma 1). We define the following mapping:

$$G(\hat{y};\beta_1) := \arg\max_{y\in\mathbb{R}^m}\left\{\nabla_y d(\hat{y};\beta_1)^T(y-\hat{y}) - \frac{L^d(\beta_1)}{2}\|y-\hat{y}\|^2\right\}, \tag{50}$$

where $L^d(\beta_1) := \frac{\|A_1\|^2}{\beta_1\sigma_1} + \frac{\|A_2\|^2}{\beta_1\sigma_2}$ and $\nabla_y d(\hat{y};\beta_1) = A_1x_1^*(\hat{y};\beta_1) + A_2x_2^*(\hat{y};\beta_1) - b$. This problem can explicitly be solved to get the unique solution:

$$G(\hat{y};\beta_1) = \hat{y} + L^d(\beta_1)^{-1}\big[Ax^*(\hat{y};\beta_1) - b\big]. \tag{51}$$

4.2 A decomposition scheme with switching primal-dual steps

First, we adapt the scheme (27)–(28) in the switching primal-dual framework. Suppose that the pair $(\bar{x},\bar{y}) \in X\times\mathbb{R}^m$ satisfies the excessive gap condition (21). The two primal steps scheme computes $(\bar{x}^+,\bar{y}^+)$ as follows:

$$\big(\bar{x}^+,\bar{y}^+\big) := \mathcal{A}^p(\bar{x},\bar{y};\beta_1,\beta_2,\tau) \quad\Longleftrightarrow\quad \begin{cases}\hat{x} := (1-\tau)\bar{x} + \tau x^*(\bar{y};\beta_1),\\ \bar{y}^+ := (1-\tau)\bar{y} + \tau y^*(\hat{x};\beta_2),\\ \bar{x}^+ := P(\hat{x};\beta_2),\end{cases} \tag{52}$$

and then updates $\beta_1^+ := (1-\tau)\beta_1$, where $\tau\in(0,1)$ and $P(\cdot;\beta_2)$ is defined in (24). The difference between schemes $\mathcal{A}_m^p$ and $\mathcal{A}^p$ is that the parameter β_2 is fixed in $\mathcal{A}^p$.

Symmetrically, the two dual steps scheme computes $(\bar{x}^+,\bar{y}^+)$ as:

$$\big(\bar{x}^+,\bar{y}^+\big) := \mathcal{A}^d(\bar{x},\bar{y};\beta_1,\beta_2,\tau) \quad\Longleftrightarrow\quad \begin{cases}\hat{y} := (1-\tau)\bar{y} + \tau y^*(\bar{x};\beta_2),\\ \bar{x}^+ := (1-\tau)\bar{x} + \tau x^*(\hat{y};\beta_1),\\ \bar{y}^+ := G(\hat{y};\beta_1),\end{cases} \tag{53}$$

where $\tau\in(0,1)$. The parameter β_1 is kept unchanged, while β_2 is updated by $\beta_2^+ := (1-\tau)\beta_2$.

The following result shows that $(\bar{x}^+,\bar{y}^+)$ generated either by $\mathcal{A}^p$ or by $\mathcal{A}^d$ maintains the excessive gap condition (21).

Lemma 6 *Suppose that* $(\bar{x}, \bar{y}) \in X \times \mathbb{R}^m$ *and satisfies* (21) *with respect to two values* β_1 *and* β_2. *Then if the parameter* τ *is chosen such that* $\tau \in (0, 1)$ *and*:

$$\beta_1\beta_2 \geq \frac{2\tau^2}{1-\tau}\max_{1\leq i\leq 2}\left\{\frac{\|A_i\|^2}{\sigma_i}\right\}, \tag{54}$$

then the new point $(\bar{x}^+, \bar{y}^+)$ *generated either by the scheme* $\mathcal{A}^p$ *or by* $\mathcal{A}^d$ *is in* $X \times \mathbb{R}^m$ *and maintains the excessive gap condition* (21) *with respect to either two new values* β_1^+ *and* β_2 *or* β_1 *and* β_2^+.

The proof of this lemma is quite similar to [27, Theorem 4.2.] that we omit here.

Remark 6 Given $\beta_1 > 0$, we can choose $\beta_2 > 0$ such that the condition (26) holds. We compute a point $(\bar{x}^0, \bar{y}^0)$ as:

$$\bar{x}^0 := x^*\big(y^c; \beta_1\big) \quad \text{and} \quad \bar{y}^0 := G\big(y^c; \beta_1\big) = L^d(\beta_1)^{-1}(A\bar{x} - b). \tag{55}$$

Then, similar to (25), the point $(\bar{x}^0, \bar{y}^0)$ satisfies (21). Therefore, we can use this point as a starting point for Algorithm 2 below.

In Algorithm 2 below we apply either the scheme $\mathcal{A}^p$ or $\mathcal{A}^d$ by using the following rule:

Rule A *If the iteration counter* k *is even then apply* $\mathcal{A}^p$. *Otherwise,* $\mathcal{A}^d$ *is used.*

Now, we provide an update rule to generate a sequence $\{\tau_k\}$ such that the condition (54) holds. Let $\bar{L}^2 := 2\max_{1\leq i\leq 2}\{\frac{\|A_i\|^2}{\sigma_i}\}$. Suppose that at the iteration k the condition (54) holds, i.e.:

$$\beta_1^k\beta_2^k \geq \frac{\tau_k^2}{1-\tau_k}\bar{L}^2. \tag{56}$$

Note that, at the iteration $k+1$, we either update β_1^k or β_2^k. Thus we have $\beta_1^{k+1}\beta_2^{k+1} = (1-\tau_k)\beta_1^k\beta_2^k$. However, since the condition (56) holds, we have $(1-\tau_k)\beta_1^k\beta_2^k \geq \tau_k^2\bar{L}^2$. Now, we suppose that the condition (54) is satisfied with β_1^{k+1} and β_2^{k+1}, i.e.:

$$\beta_1^{k+1}\beta_2^{k+1} \geq \frac{\tau_{k+1}^2}{1-\tau_{k+1}}\bar{L}^2.$$

This condition holds if $\tau_k^2\bar{L}^2 \geq \frac{\tau_{k+1}^2}{1-\tau_{k+1}}\bar{L}^2$, which leads to $\tau_{k+1}^2 + \tau_k^2\tau_{k+1} - \tau_k^2 \leq 0$. Since $\tau_k, \tau_{k+1} \in (0, 1)$, we obtain $0 < \tau_{k+1} \leq \frac{\tau_k}{2}[\sqrt{\tau_k^2+4} - \tau_k] < \tau_k$. The tightest rule for updating τ_k is:

$$\tau_{k+1} := \frac{\tau_k}{2}\Big[\sqrt{\tau_k^2+4} - \tau_k\Big], \tag{57}$$

for all $k \geq 0$ and $\tau_0 \in (0, 1)$ given. Associated with $\{\tau_k\}$, we generate two sequences $\{\beta_1^k\}$ and $\{\beta_2^k\}$ as:

$$\beta_1^{k+1} := \begin{cases} (1-\tau_k)\beta_1^k & \text{if } k \text{ is even} \\ \beta_1^k & \text{otherwise,} \end{cases} \quad \text{and} \quad \beta_2^{k+1} := \begin{cases} \beta_2^k & \text{if } k \text{ is even} \\ (1-\tau_k)\beta_2^k & \text{otherwise,} \end{cases} \tag{58}$$

where $\beta_1^0 = \beta_2^0 > 0$ are fixed.

Lemma 7 *Let* $\{\tau_k\}$, $\{\beta_1^k\}$ *and* $\{\beta_2^k\}$ *be three sequences generated by* (57) *and* (58), *respectively. Then*:

$$\frac{(1-\tau_0)\beta_1^0}{2\tau_0 k + 1} < \beta_1^k < \frac{2\beta_1^0\sqrt{1-\tau_0}}{\tau_0 k}, \quad \text{and} \quad \frac{\beta_2^0\sqrt{1-\tau_0}}{2\tau_0 k + 1} < \beta_2^k < \frac{2\beta_2^0}{\tau_0 k}, \tag{59}$$

for all $k \geq 1$, *provided that* $\beta_1^0 = \beta_2^0 > 0$.

The proof of this lemma can be found in the Appendix.

Remark 7 We can see that the function $\eta_k(\tau_0) := \frac{4\bar{\beta}\sqrt{1-\tau_0}}{\tau_0(k+\tau_0)}$ on the right-hand side of (59) is decreasing in $(0, 1)$ for $k \geq 1$. Therefore, we can choose τ_0 as large as possible to minimize $\eta_k(\cdot)$ in $(0, 1)$. By using the conditions (26) and (54), we can derive the optimal value of τ_0 as $\tau_0 := \frac{\sqrt{5}-1}{2} \approx 0.618$.

4.3 The algorithm and its worst-case complexity

Now, we combine the results of Remark 6, Lemmas 6 and 7 to build the following algorithm.

Algorithm 2 (Decomposition algorithm with switching primal-dual steps)
Initialization: Perform the following steps:

1. Choose $\tau_0 := 0.5(\sqrt{5} - 1)$ and set $\beta_1^0 = \beta_2^0 := [2\max_{1 \leq i \leq 2}\{\frac{\|A_i\|^2}{\sigma_i}\}]^{1/2}$.
2. Compute $\bar{x}^0$ and $\bar{y}^0$ as:

$$\bar{x}^0 := x^*\big(y^c; \beta_1^0\big), \quad \text{and} \quad \bar{y}^0 := L^d\big(\beta_1^0\big)^{-1}\big(A\bar{x}^0 - b\big).$$

Iteration: For $k = 0, 1, \ldots,$ perform the following steps:

1. If a given stopping criterion is satisfied then terminate.
2. If k is even then:
   2a. Compute $(\bar{x}^{k+1}, \bar{y}^{k+1})$ as:

$$\big(\bar{x}^{k+1}, \bar{y}^{k+1}\big) := \mathcal{A}^p\big(\bar{x}^k, \bar{y}^k; \beta_1^k, \beta_2^k, \tau_k\big).$$

   2b. Update the smoothness parameter β_1^k as $\beta_1^{k+1} := (1 - \tau_k)\beta_1^k$.

3. Otherwise, i.e. if k is odd then:
   3a. Compute $(\bar{x}^{k+1}, \bar{y}^{k+1})$ as:

$$\left(\bar{x}^{k+1}, \bar{y}^{k+1}\right) := \mathcal{A}^d\left(\bar{x}^k, \bar{y}^k; \beta_1^k, \beta_2^k, \tau_k\right).$$

   3b. Update the smoothness parameter β_2^k as $\beta_2^{k+1} := (1-\tau_k)\beta_2^k$.
4. Update the step size parameter τ_k as: $\tau_{k+1} := \frac{\tau_k}{2}[(\tau_k^2+4)^{1/2} - \tau_k]$.

End.

The main steps of Algorithm 2 are Steps 2a and 2b, which require us to compute either the two primal steps scheme $\mathcal{A}^p$ or the two dual steps scheme $\mathcal{A}^d$. In $\mathcal{A}^p$, we need to solve two convex subproblem pairs in parallel, while in $\mathcal{A}^d$, it only requires to solve one convex subproblem pair in parallel. The following theorem shows the convergence of this algorithm.

Theorem 3 *Let the sequence $\{(\bar{x}^k, \bar{y}^k)\}_{k\geq 0}$ be generated by Algorithm 2. Then the duality and feasibility gaps satisfy*:

$$-D_{Y^*}\left\|A\bar{x}^{k+1} - b\right\| \leq \phi\left(\bar{x}^{k+1}\right) - d\left(\bar{y}^{k+1}\right) \leq \frac{2\bar{L}D_X}{k+1}, \tag{60}$$

and

$$\left\|A\bar{x}^{k+1} - b\right\| \leq \frac{(\sqrt{5}+1)\bar{L}D_Y}{k+1}, \tag{61}$$

where

$$\bar{L} := \left[2\max_{1\leq i\leq 2}\left\{\frac{\|A_i\|^2}{\sigma_i}\right\}\right]^{1/2}$$

and D_X, D_{Y^*} and D_Y are defined in (47).

Proof The conclusion of this theorem follows directly from Lemmas 3 and 5, the condition $\tau_0 = \frac{\sqrt{5}-1}{2}$, $\beta_1^0 = \beta_2^0 = \bar{L}$ and the fact that $\beta_1^k \leq \beta_2^k$. □

Remark 8 Note that the worst-case complexity of Algorithm 2 is still $O(\frac{1}{\varepsilon})$. However, the constant factor in the complexity estimate (48) is as same as the one in (60), while the constant factor in (49) is smaller than the one in (61). As we discuss in Sect. 6 below, the rate of decrease of τ_k in Algorithm 2 is smaller than two times of τ_k in Algorithm 1. Consequently, the sequences $\{\beta_1^k\}$ and $\{\beta_2^k\}$ generated by Algorithm 1 approach zero faster than the ones generated by Algorithm 2.

Remark 9 Note that the role of the schemes $\mathcal{A}^p$ and $\mathcal{A}^d$ in Algorithm 2 can be exchanged. Therefore, Algorithm 2 can be modified as follows to obtain a symmetric variant:

1. At Step 2 of the initialization phase, we use (25) to compute $\bar{x}^0$ and $\bar{y}^0$ instead of (55).

2. At Steps 2a, we apply $\mathcal{A}^p$ if the iteration counter k is odd. Otherwise, we use $\mathcal{A}^d$ at Step 3a.
3. At Steps 2b, β_2^k is updated if k is odd. Otherwise, β_1^k is updated at Step 3b.

5 Application to strongly convex programming problems

If ϕ_i $(i = 1, 2)$ in (2) is strongly convex then the convergence rate of the dual scheme (53) can be accelerated up to $O(\frac{1}{k^2})$.

Suppose that ϕ_i is strongly convex with a convexity parameter $\sigma_i > 0$ $(i = 1, 2)$. Then the function d defined by (5) is well-defined, concave and differentiable. Moreover, its gradient is given by:

$$\nabla d(y) = A_1 x_1^*(y) + A_2 x_2^*(y) - b, \tag{62}$$

which is Lipschitz continuous with a Lipschitz constant $L^d := \frac{\|A_1\|^2}{\sigma_1} + \frac{\|A_2\|^2}{\sigma_2}$. The excessive gap condition (21) in this case becomes:

$$f(\bar{x}; \beta_2) \leq d(\bar{y}), \tag{63}$$

for given $\bar{x} \in X$, $\bar{y} \in \mathbb{R}^m$ and $\beta_2 > 0$. From Lemma 3 we conclude that if the point $(\bar{x}, \bar{y})$ satisfies (63) then, for a given $y^* \in Y^*$, the following estimates hold:

$$-2\beta_2 \left\| y^* \right\|^2 \leq \phi(\bar{x}) - d(\bar{y}) \leq 0, \tag{64}$$

and

$$\|A\bar{x} - b\| \leq 2\beta_2 \left\| y^* \right\|. \tag{65}$$

We now adapt the scheme (53) to this special case. Suppose $(\bar{x}, \bar{y}) \in X \times \mathbb{R}^m$ satisfies (63), we generate a new pair $(\bar{x}^+, \bar{y}^+)$ as:

$$\left(\bar{x}^+, \bar{y}^+\right) := \mathcal{A}_s^d(\bar{x}, \bar{y}; \beta_2, \tau) \quad \Longleftrightarrow \quad \begin{cases} \hat{y} := (1-\tau)\bar{y} + \tau y^*(\bar{x}; \beta_2), \\ \bar{x}^+ := (1-\tau)\bar{x} + \tau x^*(\hat{y}), \\ \bar{y}^+ = \hat{y} + \frac{1}{L^d}(Ax^*(\hat{y}) - b), \end{cases} \tag{66}$$

where $y^*(\bar{x}; \beta_2) = \frac{1}{\beta_2}(A\bar{x} - b)$, and $x^*(y) := (x_1^*(y), x_2^*(y))$ is the solution of the minimization problem in (5). The parameter β_2 is updated by $\beta_2^+ := (1-\tau)\beta_2$ and $\tau \in (0, 1)$ will appropriately be chosen.

The following lemma shows that $(\bar{x}^+, \bar{y}^+)$ generated by (66) satisfies (63) whose proof can be found in [27].

Lemma 8 *Suppose that the point $(\bar{x}, \bar{y}) \in X \times \mathbb{R}^m$ satisfies the excessive gap condition (63) with the value $\beta_2 > 0$. Then if the parameter τ is chosen such that $\tau \in (0, 1)$ and:*

$$\beta_2 \geq \frac{\tau^2 L^d}{1-\tau}, \tag{67}$$

then the new point $(\bar{x}^+, \bar{y}^+)$ *computed by* (66) *is in* $X \times \mathbb{R}^m$ *and also satisfies* (63) *with a new parameter value* β_2^+.

Now, let us derive the rule to update the parameter τ. Suppose that β_2 satisfies (67). Since $\beta_2^+ = (1-\tau)\beta_2$, the condition (67) holds for β_2^+ if $\tau^2 \geq \frac{\tau_+^2}{1-\tau_+}$. Therefore, similar to Algorithm 2, we update the parameter τ by using the rule (44). The conclusion of Lemma 7 still holds for this case.

Before presenting the algorithm, it is necessary to find a starting point $(\bar{x}^0, \bar{y}^0)$ which satisfies (63). Let $\beta_2 := L^d$. We compute $(\bar{x}^0, \bar{y}^0)$ as:

$$\bar{x}^0 := x^*\big(y^c\big) \quad \text{and} \quad \bar{y}^0 := \frac{1}{L^d}\big(A\bar{x}^0 - b\big). \tag{68}$$

It follows from Lemma 7.4 [27] that $(\bar{x}^0, \bar{y}^0)$ satisfies the excessive gap condition (63).

Finally, the decomposition algorithm for solving the strongly convex programming problem of the form (2) is described in detail as follows:

Algorithm 3 (Decomposition algorithm for strongly convex objective function)
Initialization: Perform the following steps:

1. Choose $\tau_0 := 0.5(\sqrt{5}-1)$. Set $\beta_2^0 := \frac{\|A_1\|^2}{\sigma_1} + \frac{\|A_2\|^2}{\sigma_2}$.
2. Compute $\bar{x}^0$ and $\bar{y}^0$ as:

$$\bar{x}^0 := x^*\big(y^c\big) \quad \text{and} \quad \bar{y}^0 := \frac{1}{L^d}\big(A\bar{x}^0 - b\big).$$

Iteration: For $k = 0, 1, \ldots,$ perform the following steps:

1. If a given stopping criterion is satisfied then terminate.
2. Compute $(\bar{x}^{k+1}, \bar{y}^{k+1})$ using scheme (66):

$$\big(\bar{x}^{k+1}, \bar{y}^{k+1}\big) := \mathcal{A}_s^d\big(\bar{x}^k, \bar{y}^k; \beta_2^k, \tau_k\big).$$

3. Update the smoothness parameter as: $\beta_2^{k+1} := (1-\tau_k)\beta_2^k$.
4. Update the step size parameter τ_k as: $\tau_{k+1} := \frac{\tau_k}{2}[(\tau_k^2+4)^{1/2} - \tau_k]$.

End.

The convergence of Algorithm 3 is stated as in Theorem 4 below.

Theorem 4 *Let* $\{(\bar{x}^k, \bar{y}^k)\}_{k\geq 0}$ *be a sequence generated by Algorithm* 3. *Then the following duality and feasibility gaps are satisfied*:

$$-\frac{4L^d D_{Y^*}^2}{(k+2)^2} \leq \phi\big(\bar{x}^k\big) - d\big(\bar{y}^k\big) \leq 0, \tag{69}$$

and

$$\big\|A\bar{x}^k - b\big\| \leq \frac{4L^d D_{Y^*}}{(k+2)^2}, \tag{70}$$

where $L^{\phi} := \frac{\|A_1\|^2}{\sigma_1} + \frac{\|A_2\|^2}{\sigma_2}$ *and* D_{Y^*} *is defined in* (47).

Proof From the update rule of τ^k, we have $(1 - \tau_{k+1}) = \frac{\tau_{k+1}^2}{\tau_k^2}$. Moreover, since $\beta_2^{k+1} = (1 - \tau_k)\beta_2^k$, it implies that

$$\beta_2^{k+1} = \beta_2^0 \prod_{i=0}^{k}(1 - \tau_i) = \frac{\beta_2^0(1 - \tau_0)}{\tau_0^2}\tau_k^2.$$

By using the inequalities (78) and $\beta_2^0 = L^d$, we have $\beta_2^{k+1} < \frac{4L^d(1-\tau_0)}{(\tau_0 k+2)^2}$. With $\tau_0 = 0.5(\sqrt{5} - 1)$, one has $\beta_2^k < \frac{4L^d}{(k+2)^2}$. By substituting this inequality into (64) and (65), we obtain (69) and (70), respectively. □

Theorem 4 shows that the worst-case complexity of Algorithm 3 is $O(\frac{4L^d D_{Y^*}^2}{\sqrt{\varepsilon}})$. Moreover, at each iteration of this algorithm, only two convex problems need to be solved *in parallel*.

6 Discussion on implementation and theoretical comparison

In order to apply Algorithm 1, 2 or 3 to solve the problem (1), we need to choose a prox-function for each feasible set X_i for $i = 1, \ldots, M$. The simplest prox-function is $p_i(x_i) := \frac{1}{2}\|x_i - x_i^c\|^2$, for a given $x_i^c \in X_i$. However, in some applications, we can choose an appropriate prox-function such that it captures the structure of the feasible set X_i. In (24), we have used the Euclidean distance to construct the proximal terms. In principle, we can use a generalized Bregman distance instead of the Euclidean distance, see [26] for more details.

6.1 Extension to a multi-component separable objective function

The algorithms developed in the previous sections can be directly applied to solve problem (1) in the case $M > 2$. First, we provide the following formulas to compute the parameters of Algorithms 1–3.

1. The constant $\bar{L}$ in Theorems 2 and 3 is replaced by
$$\bar{L}_M = \left[M \max_{1 \le i \le M}\left\{\frac{\|A_i\|^2}{\sigma_i}\right\}\right]^{1/2}.$$
2. The initial values of β_1^0 and β_2^0 in Algorithms 2 and 3 are $\beta_1^0 = \beta_2^0 = \bar{L}_M$.
3. The Lipschitz constant $L_i^{\psi}(\beta_2)$ in Lemma 2 is $L_i^{\psi}(\beta_2) = \beta_2^{-1} M\|A_i\|^2$ $(i = 1, \ldots, M)$.
4. The Lipschitz constant $L^d(\beta_1)$ in Lemma 1 is
$$L^d(\beta_1) := \frac{1}{\beta_1}\sum_{i=1}^{M}\frac{\|A_i\|^2}{\sigma_i}.$$

5. The Lipschitz constant L^d in Algorithm 3 is

$$L^d := \sum_{i=1}^{M} \frac{\|A_i\|^2}{\sigma_i}.$$

Note that these constants depend on the number of components M and the structure of matrix A_i $(i = 1, \ldots, M)$.

Next, we rewrite the smoothed dual function $d(y; \beta_1)$ defined by (12) for the case $M > 2$ as follows:

$$d(y; \beta_1) = \sum_{i=1}^{M} d_i(y; \beta_1),$$

where the function values $d_i(y; \beta_1)$ can be computed in parallel as:

$$d_i(y; \beta_1) = -M^{-1} b_i^T y + \min_{x_i \in X_i} \left\{\phi_i(x_i) + y^T A_i x_i + \beta_1 p_i(x_i)\right\}, \quad \forall i = 1, \ldots, M.$$

The quantities $\hat{y}$ and $y^+ := G(\hat{y}; \beta_1)$ defined in (52) and (53) can respectively be expressed as:

$$\hat{y} := (1 - \tau)\bar{y} + (1 - \tau) \sum_{i=1}^{M} \frac{1}{\beta_2} \left(A_i \bar{x}_i - \frac{1}{M} b \right), \quad \text{and}$$

$$y^+ := \hat{y} + \sum_{i=1}^{M} \left[\frac{1}{L^d(\beta_1)} \left(A_i x_i^*(\hat{y}; \beta_1) - \frac{1}{M} b \right) \right].$$

These formulas show that each component of $\hat{y}$ and y^+ can be computed by only using the local information and its neighborhood information. Therefore, both algorithms are highly distributed.

Finally, we note that if there exists a component ϕ_i of the objective function ϕ which is Lipschitz continuously differentiable then the mapping $G_i(\hat{x}; \beta_2)$ defined by (39) corresponding to the primal convex subproblem of this component can be used instead of the proximity mapping $P_i(\hat{x}; \beta_2)$ defined by (24). This modification can reduce the computational cost of the algorithms. The sequence $\{\tau_k\}_{k \geq 0}$ generated by the rule (44) still maintains the condition (42) in Remark 3.

6.2 Theoretical comparison

Firstly, we compare Algorithms 1 and 2. From Lemma 3 and the proof of Theorems 2 and 3 we see that the rate of convergence of both algorithms is as same as of β_1^k and β_2^k. At each iteration, Algorithm 1 updates simultaneously β_1^k and β_2^k by using the same value of τ_k, while Algorithm 2 updates only one parameter. Therefore, to update both parameters β_1^k and β_2^k, Algorithm 2 needs two iterations. We analyze the update rule of τ_k in Algorithms 1 and 2 to compare the rate of convergence of both algorithms.

Let us define

$$\xi_1(\tau) := \frac{\tau}{\tau+1} \quad \text{and} \quad \xi_2(\tau) := \frac{\tau}{2}\Big[\sqrt{\tau^2+4}-\tau\Big].$$

The function ξ_2 can be rewritten as $\xi_2(\tau) = \frac{\tau}{\sqrt{(\tau/2)^2+1}+\tau/2}$. Therefore, we can easily show that:

$$\xi_1(\tau) < \xi_2(\tau) < 2\xi_1(\tau).$$

If we denote by $\{\tau_k^{\mathrm{A}_1}\}_{k\geq 0}$ and $\{\tau_k^{\mathrm{A}_2}\}_{k\geq 0}$ the two sequences generated by Algorithms 1 and 2, respectively then we have $\tau_k^{\mathrm{A}_1} < \tau_k^{\mathrm{A}_2} < 2\tau_k^{\mathrm{A}_1}$ for all k provided that $2\tau_0^{\mathrm{A}_1} \geq \tau_0^{\mathrm{A}_2}$. Since Algorithm 1 updates β_1^k and β_2^k simultaneously while Algorithm 2 updates each of them at each iteration. If we choose $\tau_0^{\mathrm{A}_1} = 0.5$ and $\tau_0^{\mathrm{A}_2} = 0.5(\sqrt{5}-1)$ in Algorithms 1 and 2, respectively, then, by directly computing the values of $\tau_k^{\mathrm{A}_1}$ and $\tau_k^{\mathrm{A}_2}$, we can see that $2\tau_k^{\mathrm{A}_1} > \tau_k^{\mathrm{A}_2}$ for all $k \geq 0$. Consequently, the sequences $\{\beta_1^k\}$ and $\{\beta_2^k\}$ in Algorithm 1 converge to zero faster than in Algorithm 2. In other words, Algorithm 1 is faster than Algorithm 2.

Now, we compare Algorithm 1, Algorithm 2 and Algorithm 3.2. in [22] (see also [35]). Note that the smoothness parameter β_1 is fixed in [22, Algorithm 3.2]. Moreover, this parameter is proportional to the given desired accuracy ε, i.e. $\beta_1 := \frac{\varepsilon}{D_X}$, which is often very small. Thus, the Lipschitz constant $L^d(\beta_1)$ is very large. Consequently, [22, Algorithm 3.2] makes a slow progress toward a solution at the very early iterations. In Algorithms 1 and 2, the parameters β_1 and β_2 are dynamically updated starting from given values. Besides, the cost per iteration of [22, Algorithm 3.2] is more expensive than Algorithms 1 and 2 since it requires to solve two convex subproblem pairs in parallel and two dual steps.

6.3 Stopping criterion

In order to terminate the above algorithms, we can use the smooth function $d(\cdot;\beta_1)$ to measure the stopping criterion. It is clear that if β_1 is small, $d(\cdot;\beta_1)$ is an approximation of the dual function d due to Lemma 1. Therefore, we can estimate the duality gap $\phi(x) - d(y)$ by $\phi(x) - d(y;\beta_1)$ and use this quantity in the stopping criterion. More precisely, we terminate the algorithms if:

$$\texttt{rpfgap} := \frac{\|A\bar{x}^k - b\|}{\max\{r^0, 1\}} \leq \varepsilon_{\mathrm{feas}}, \tag{71}$$

and either the approximate duality gap satisfies:

$$\big|f(\bar{x}^k;\beta_2^k) - d\big(\bar{y}^k;\beta_1^k\big)\big| \leq \varepsilon_{\mathrm{fun}} \max\big\{1.0, \big|d\big(\bar{y}^k;\beta_1^k\big)\big|, \big|f\big(\bar{x}^k;\beta_2^k\big)\big|\big\}, \tag{72}$$

or the value $\phi(\bar{x}^k)$ does not significantly change in $j_{\max}$ successive iterations, i.e.:

$$\frac{|\phi(\bar{x}^k) - \phi(\bar{x}^{k-j})|}{\max\{1.0, |\phi(\bar{x}^k)|\}} \leq \varepsilon_{\mathrm{obj}} \quad \text{for } j = 1, \ldots, j_{\max}, \tag{73}$$

where $r^0 := \|A\bar{x}^0 - b\|$ and $\varepsilon_{\mathrm{feas}}$, $\varepsilon_{\mathrm{fun}}$ and $\varepsilon_{\mathrm{obj}}$ are given tolerances.

7 Numerical tests

In this section, we verify the performance of the proposed algorithms by applying them to solve three numerical examples. We test Algorithms 1 and 2 for the two first examples and Algorithm 3 for the last example. We also compare our methods with the exact proximal-based decomposition method (`EPBDM`) in [3], the nonmonotone proximal-center based decomposition method (`PCBDM`) [22, 35] and the parallel variant of the alternating direction method of multipliers (`ADMM`) presented in [19].

7.1 Implementation details

All the algorithms have been implemented in C++ running on a 16 cores Intel ®Xeon 2.7 GHz workstation with 12 GB of RAM. In order to solve general convex primal subproblems, we either implemented a primal-dual interior point method using Mehrotra's predictor-corrector scheme [21] which we call `pcPDIpAlg` or used the `IpOpt` solver [38]. All the algorithms have been parallelized by using `OpenMP`.

We chose quadratic prox-functions $p_{X_i}(x_i) := \frac{1}{2}\|x_i - x_i^c\|_2^2$ in our algorithms and `PCBDM`, where $x_i^c \in \mathbb{R}^{n_i}$ is the center point of X_i for $i = 1, \ldots, M$. We initialized the `pcPDIpAlg` and `IpOpt` solvers at the values given by the previous iteration. The accuracy levels in both solvers were fixed at 10^{-8}. The parameter β_1 in the primal subproblems of `PCBDM` was fixed at $\beta_1 := \frac{\varepsilon_{\text{fun}} \max\{1.0, |\phi(\bar{x}^0)|\}}{D_X}$.

For `ADMM`, we considered three variants with respect to three strategies of updating the penalty parameter ρ_k. In the first and the second variants which we named `ADMM-v1` and `ADMM-v2`, we used the tuning rule in [14, Strategy 3, p. 352] with $\rho_0 = 1$ and $\rho_0 = 10^3$, respectively. In the third variant, named `ADMM-v3`, the penalty parameter was fixed at $\rho_k = 10^3$ for all iterations.

For `EPBDM`, we used an exact variant of [3, Algorithm 1], where we chose the proximity parameter as follows. First, we chose $\varepsilon_c := 0.5\min\{\frac{1}{3}, \frac{1}{2\|A\|+1}\}$ and then set $\underline{\beta}_1 := [\min\{\frac{1-\varepsilon_c}{2}, \frac{1-\varepsilon_c}{2\|A\|}\}]^{-1}$ and $\bar{\beta}_1 := \varepsilon_c^{-1}$. Finally, we selected $\beta_1 = 0.5(\underline{\beta}_1 + \bar{\beta}_1)$.

We terminated Algorithms 1, 2 and 3 if the condition (71) and either (72) or (73) were satisfied. We terminated the three last algorithms if (71) and (73) were satisfied. Here the tolerances were set to $\varepsilon_{\text{feas}} = \varepsilon_{\text{fun}} = \varepsilon_{\text{obj}} = 10^{-3}$ and $j_{\max} = 5$ in all algorithms. We also terminated all the algorithms if they reached the maximum number of iterations `maxiter`. We claimed that the problem could not be solved if either any primal subproblem could not be solved or the maximum number of iterations was reached in our implementation.

We benchmarked all algorithms with performance profiles [5]. Recall that a performance profile is built based on a set $\mathcal{S}$ of n_s algorithms (solvers) and a collection $\mathcal{P}$ of n_p problems. Suppose that we build a profile based on computational time. We denote by $T_{p,s} := \textit{computational time required to solve problem } p \textit{ by solver } s$. We compare the performance of algorithm s on problem p with the best performance of any algorithm on this problem; that is we compute the performance ratio $r_{p,s} := \frac{T_{p,s}}{\min\{T_{p,\hat{s}} | \hat{s} \in \mathcal{S}\}}$. Now, let $\tilde{\rho}_s(\tilde{\tau}) := \frac{1}{n_p}\text{size}\{p \in \mathcal{P} \mid r_{p,s} \le \tilde{\tau}\}$ for $\tilde{\tau} \in \mathbb{R}_+$. The function $\tilde{\rho}_s : \mathbb{R} \to [0, 1]$ is the probability for solver s that a performance ratio is within a factor $\tilde{\tau}$ of the best possible ratio. We use the term "performance profile" for the distribution function

$\tilde{\rho}_s$ of a performance metric. We plotted the performance profiles in log-scale, i.e. $\rho_s(\tau) := \frac{1}{n_p}\text{size}\{p \in \mathcal{P} \mid \log_2(r_{p,s}) \leq \tau := \log_2 \tilde{\tau}\}$.

7.2 Numerical experiments

We considered three numerical examples. The first example is a separable convex quadratic programming problem. The second one is a nonlinear smooth separable convex programming problem, while the last problem is a DSL dynamic spectrum management problem.

7.2.1 Separable convex quadratic programming

Let us consider the following separable convex quadratic programming (QP) problem:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} & \left\{ \phi(x) := \sum_{i=1}^{M} \frac{1}{2} x_i^T Q_i x_i + q_i^T x_i \right\} \\ \text{s.t.} \quad & \sum_{i=1}^{M} A_i x_i = b, \\ & x_i \succeq 0, \quad i = 1, \ldots, M, \end{aligned} \tag{74}$$

where Q_i is symmetric, positive semidefinite for $i = 1, \ldots, M$. We tested the above algorithms on a collection of QP problems of the form (74) and compared them via performance profiles.

Problem generation The data of the test collection was generated as follows:

- Matrix $Q_i := R_i R_i^T$, where R_i is an $n_i \times r_i$ random matrix in $[l_Q, u_Q]$ with $r_i := \lfloor n_i/2 \rfloor$.
- Matrix A_i was generated randomly in $[l_A, u_A]$.
- Vector $q_i := -Q_i x_i^0$, where x_i^0 is a given feasible point in $(0, r_{x_0})$ and vector $b := \sum_{i=1}^{M} A_i x_i^0$.
- The density of both matrices A_i and R_i is γ_A.

Note that the problems generated as above are always feasible. Moreover, they are not strongly convex. The test collection consisted of $n_p = n_{p_1} + n_{p_2} + n_{p_3}$ problems with different sizes and the sizes were generated randomly as follows:

- *Class* 1: $n_{p_1} = 20$ problems with $20 < M < 100$, $50 < m < 500$, $5 < n_i < 100$ and $\gamma_A = 0.5$.
- *Class* 2: $n_{p_2} = 20$ problems with $100 < M < 1000$, $100 < m < 600$, $10 < n_i < 50$ and $\gamma_A = 0.1$.
- *Class* 3: $n_{p_3} = 10$ or 20 problems with $1000 < M < 2000$, $500 < m < 1000$, $100 < n_i < 200$ and $\gamma_A = 0.05$.

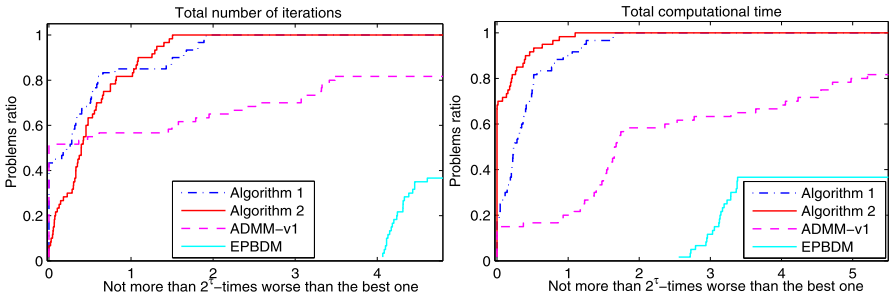

Fig. 1 Performance profiles in $\log_2$ scale for Scenario I: *left*—number of iterations, *right*—CPU time. The solver for the primal subproblems: `pcPDIpAlg`

Scenarios We considered two different scenarios:

Scenario I: In this scenario, we aimed to test Algorithms 1 and 2, `ADMM-v1` and `EPBDM`, where we generated the values of Q relatively small to see an affect of matrix A to the performance of these algorithms. More precisely, we chose $[l_Q, u_Q] = [-0.1, 0.1]$, $[l_A, u_A] = [-1, 1]$ and $r_{x_0} = 2$. We tested on a collection of $n_p = 60$ problems with $n_{p_1} = n_{p_2} = n_{p_3} = 20$.

Scenario II: The second scenario aimed to test the `ADMM` algorithms (with three different ways of updating the penalty parameter) on a collection of $n_p = 50$ problems, where $n_{p_1} = n_{p_2} = 20$ and $n_{p_3} = 10$. We considered three different strategies for updating the penalty parameter ρ_k and chose $[l_Q, u_Q] = [-1, 1]$, $[l_A, u_A] = [-5, 5]$ and $r_{x_0} = 5$.

Results In the first scenario, the size of the problems satisfied $23 \le M \le 1969$, $95 \le m \le 986$ and $1111 \le n \le 293430$. In Fig. 1, the performance profiles of the four algorithms are plotted with respect to the number of iterations and the total of computational time.

From these performance profiles, we observe that Algorithms 1 and 2 converge for all problems. `ADMM-v1` was successful in solving 49/60 (81.67 %) problems while `EPBDM` could only solve 22/60 (36.67 %) problems. `ADMM-v1` and `EPBDM` failed to solve some problems because the maximum number of iterations was reached or the primal subproblems could not be solved. `ADMM-v1` is the best one in terms of number of iterations. It could solve up to 31/60 (51.67 %) problems with the best performance. Algorithm 1 solved 25/60 (41.67 %) problems with the best performance, while this ratio is only 5/60 (8.33 %) in Algorithm 2. If we compare the computational time then Algorithm 2 is the best one. It could solve up to 41/60 (68.33 %) problems with the best performance. Algorithm 1 and `ADMM-v1` solved 10/60 (16.67 %) and 9/60 (15 %) problems with the best performance, respectively.

Since the performance of Algorithms 1 and 2 and `ADMM` were comparable in the first scenario, we considered the second scenario where we tested Algorithms 1 and 2, `ADMM-v1`, `ADMM-v2` and `ADMM-v3` on a collection of $n_p = 50$ problems. The performance profiles of these algorithms are shown in Fig. 2. From these performance profiles we can observe the following. Algorithms 1 and 2, `ADMM-v1` and `ADMM-v2` were successful in solving all problems, while `ADMM-v3` could only solve 24/50

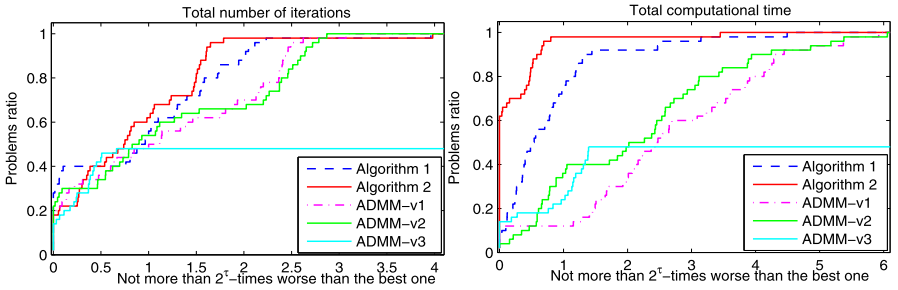

Fig. 2 Performance profiles in $\log_2$ scale for Scenario II: *left*—number of iterations, *right*—CPU time. The solver for the primal subproblems: `IpOpt`

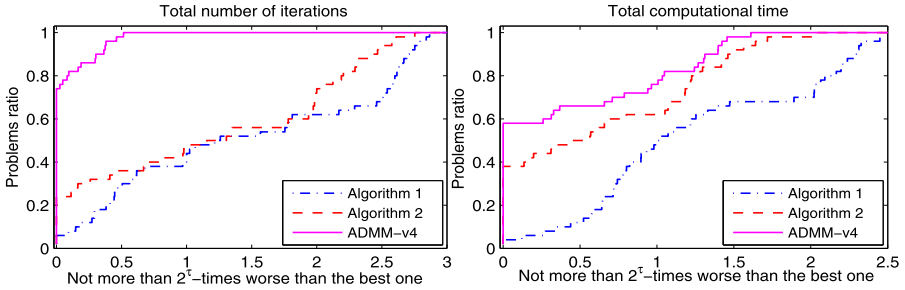

Fig. 3 Performance profiles in $\log_2$ scale for Scenario II by using `Cplex`: *left*—number of iterations, *right*—computational time

(48 %) problems. In terms of the number of iterations, Algorithm 1 solved 14/50 (28 %) problems with the best performance. This ratio in Algorithm 2, `ADMM-v1`, `ADMM-v2` and `ADMM-v2` was 9/50 (18 %), 9/50 (18 %), 11/50 (22 %) and 7/50 (14 %), respectively. In terms of the total of computational time, Algorithm 2 could solve 31/50 (62 %) problems with the best performance, while this quantity was 4/50 (8 %), 6/50 (12 %), 2/50 (4 %) and 7/50 (14 %) in Algorithm 1, `ADMM-v1`, `ADMM-v2` and `ADMM-v2`, respectively.

In order to see the effect of the penalty parameter ρ to the performance of `ADMM`, we tuned this parameter and tested Scenario II using `Cplex` as an optimization solver for solving the primal subproblems. `ADMM` was applied to solve (74) with different fixed values of the penalty parameter in $\{0.5, 1, 2.5, 5, 10, 50, 100, 250, 500, 1000\}$ and then recored the best result in terms of number of iterations corresponding to those values. We denote this `ADMM` variant by `ADMM-v4`. The performance profiles of Algorithms 1 and 2 and `ADMM-v4` are shown in Fig. 3. for a collection of $n_p = 50$ problems. Here, the size of the problems satisfied $23 \leq M \leq 1992$, $95 \leq m \leq 991$ and $1111 \leq n \leq 297818$. We can see from this figure that `ADMM-v4` is the best in terms of number of iterations as well as computational time. This test shows that the performance of `ADMM` crucially depends on the choice of the penalty parameter. In contrast to this, Algorithms 1 and 2 do not need tuning their parameters while still providing good performance when comparing the total computational time. There-

fore, the algorithms proposed in this paper are especially well suited when tuning is too costly because we are not in the situation where a set of problems with slightly different data needs to be solved. For example, when the solution of a single problem instance needs to be found tuning is completely superfluous.

7.2.2 Nonlinear smooth separable convex programming

We consider the nonlinear smooth separable convex programming problem:

$$\begin{cases} \min\limits_{x_i \in \mathbb{R}^{n_i}} \left\{ \phi(x) := \sum\limits_{i=1}^{M} \frac{1}{2}\left(x_i - x_i^0\right) Q_i \left(x_i - x_i^0\right) - w_i \ln\left(1 + b_i^T x_i\right) \right\}, \\ \text{s.t.} \quad \sum\limits_{i=1}^{M} A_i x_i = b, \\ \qquad x_i \succeq 0, \quad i = 1, \ldots, M. \end{cases} \tag{75}$$

Here, Q_i is a positive semidefinite matrix and x_0^i is given vector, $i = 1, \ldots, M$.

Problem generation In this example, we generated a collection of n_p test problems based on the following steps:

- Matrix Q_i is diagonal and was generated randomly in $[l_Q, u_Q]$.
- Matrix A_i was generated randomly in $[l_A, u_A]$ with the density γ_A.
- Vectors b_i and w_i were generated randomly in $[l_b, u_b]$ and $[0, 1]$, respectively, such that $w_i \geq 0$ and $\sum_{i=1}^{M} w_i = 1$.
- Vector $b := \sum_{i=1}^{M} A_i x_i^0$ for a given x_i^0 in $[0, r_{x_0}]$.

The size of the problems was generated randomly based on the following rules:

- *Class* 1: n_{p_1} problems with $20 < M < 50$, $50 < m < 100$, $10 < n_i < 50$ and $\gamma_A = 1.0$.
- *Class* 2: n_{p_2} problems with $50 < M < 250$, $100 < m < 200$, $20 < n_i < 50$ and $\gamma_A = 0.5$.
- *Class* 3: n_{p_3} problems with $250 < M < 1000$, $100 < m < 500$, $50 < n_i < 100$ and $\gamma_A = 0.1$.
- *Class* 4: n_{p_4} problems with $1000 < M < 5000$, $500 < m < 1000$, $50 < n_i < 100$ and $\gamma_A = 0.05$.
- *Class* 5: n_{p_5} problems with $5000 < M < 10000$, $500 < m < 1000$, $50 < n_i < 100$ and $\gamma_A = 0.01$.

Scenarios We observed that `PCBDM` and `EPBDM` were slow in this example compared to the others, and `ADMM-v1` was the best one of the three ADMM variants. We only tested Algorithms 1 and 2 and `ADMM-v1` on this example. We considered two different scenarios as in the previous example:

Scenario I: $[l_Q, u_Q] \equiv [0.0, 0.0]$ (i.e. without quadratic term), $[l_b, u_b] \equiv [0, 100]$, $[l_A, u_A] \equiv [-1, 1]$ and $r_{x_0} = 10$.
Scenario II: $[l_Q, u_Q] \equiv [-0.01, 0.01]$, $[l_b, u_b] \equiv [0, 100]$, $[l_A, u_A] \equiv [-1, 1]$ and $r_{x_0} = 1$.

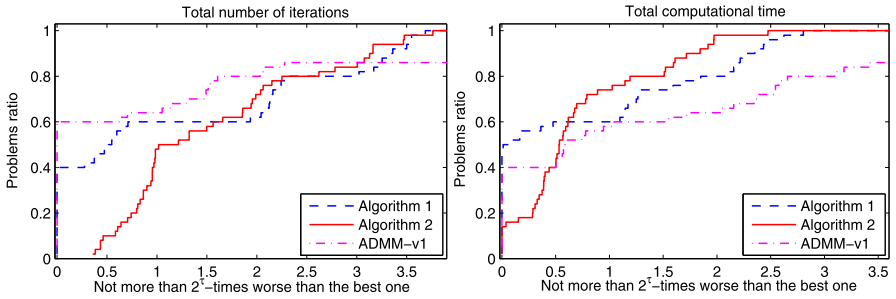

Fig. 4 Performance profiles in $\log_2$ scale for `Scenario I`: *left*—number of iterations, *right*—CPU time. The solver for the primal subproblems: `IpOpt`

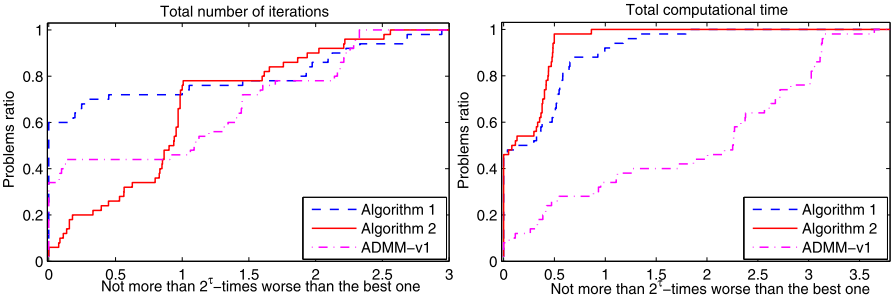

Fig. 5 Performance profiles on `Scenario II` in $\log_2$ scale: *left*—number of iterations, *right*—CPU time. The solver for the primal subproblems: `IpOpt`

Results For *Scenario I*, we tested Algorithms 1 and 2 and `ADMM-v1` on a collection of $n_p = 50$ problems. Here $n_{p_i} = 10$ for $i = 1, \ldots, 5$. The size of the problems is in $20 \leq M \leq 9396$, $50 \leq m \leq 971$ and $695 \leq n \leq 698361$. The performance profiles of these algorithms are plotted in Fig. 4.

The results on this collection shows that Algorithms 1 and 2 solved all the problems, while `ADMM-v1` could solve 43/50 (86 %) problems of the collection. `ADMM-v1` is the best one in terms of number of iterations. It could solve up to 30/50 (60 %) problems with the best performance, while this number was 20/50 (40 %) in Algorithm 1. However, Algorithm 1 is faster than `ADMM-v1` in this test, it could solve 23/50 (46 %) compared to 20/50 (40 %) in `ADMM-v1`. This is due to the fact that the cost per iteration in `ADMM-v1` was higher than in Algorithm 1. Algorithm 2 could only solve 7/50 (14 %) problems with the best performance, but performed very well on average.

For *Scenario II*, we see that the size of the problems is in $20 \leq M \leq 9870$, $50 \leq m \leq 971$ and $695 \leq n \leq 734947$. The performance profiles of three algorithms are plotted in Fig. 5. The results on this collection shows that Algorithm 1 is the best one in terms of number of iterations. It could solve up to 30/50 (60 %) problems with the best performance, while `ADMM-v1` solved 17/50 (34 %) problems with the best performance. However, Algorithm 2 is the best one in terms of computational time. It could solve 23/50 (46 %) problems with the best performance. Algorithm 1 had

the same quantity while ADMM-v1 could solve 4/50 (8 %) problems with the best performance.

7.2.3 DSL dynamic spectrum management problem

Finally, we applied Algorithm 3 to solve a separable convex programming problem arising in DSL dynamic spectrum management. This problem is a convex relaxation of the original DSL dynamic spectrum management formulation considered in [34]. The objective function of this problem is given by:

$$\phi(x) := \sum_{i=1}^{M} \phi_i(x_i), \quad \text{where}$$
$$\phi_i(x_i) := a_i^T x_i - \sum_{j=1}^{n_i} c_i^j \ln\left(\sum_{l=1}^{n_i} h_i^{jl} x_i^l + g_i^l\right), \quad i = 1, \ldots, M. \tag{76}$$

Here, $a_i \in \mathbb{R}^{n_i}$, c_i, $g_i \in \mathbb{R}^{n_i}_+$ and $H_i := (h_i^{jk}) \in \mathbb{R}^{n_i \times n_i}_+$ $(i = 1, \ldots, M)$. As described in [35] the variable x_i refers to a transmit power spectral density, $n_i = N$ for all $i = 1, \ldots, M$ is the number of users, M is the number of frequency tones which is usually large and ϕ_i is a convex approximation of a desired BER function,[1] the coding gain and noise margin. A detailed model and parameter descriptions of this problem can be found in [34, 35].

Since the function ϕ is convex (but not strongly convex), we added a regularization term $\frac{\beta_1}{2}\|x - x^c\|^2$ to the objective of the original problem, where $\beta_1 > 0$ is relatively small and x^c is the prox-center of X. The objective function $\tilde{\phi}(x) := \phi(x) + \frac{\beta_1}{2}\|x - x^c\|^2$ of the resulting problem is strongly convex with a convexity parameter β_1. Moreover, we have $|\tilde{\phi}(x) - \phi(x)| \le \beta_1 D_X$ for all $x \in X$, where D_X is defined by (47). Therefore, if we apply Algorithm 3 to find a vector $\bar{x}^k$ as an ε approximate solution of the resulting problem then $\bar{x}^k$ is also an $\varepsilon + \beta_1 D_X$ approximate solution of the original problem. In our problem, D_X is proportional to 10^{-6} and the magnitudes of the objective function are proportional to 10^3. In order to get the relative accuracy $O(10^{-3})$ we chose β_1 between $[10^5, 10^6]$. The resulting problem is indeed in the form of (1) with a strongly convex objective function.

We tested Algorithm 3 for solving the above resulting problem with different 9 scenarios and compared the results with ADMM-v3, PCBDM and EPBDM. The parameters of the problems were selected as in [34, 35]. In this example, we observed that ADMM-v3 was the most suitable of the three ADMM variants. We also note that the problem possesses coupling inequality constraints. In ADMM we added a slack variables x_{M+1} to transform it into a problem with equality coupling constraints.

The numerical results of the four algorithms are reported in Table 1 for the different 9 scenarios. Here, Iter and Cpu_time are the number of iterations and the CPU time in seconds, respectively; Obj_val and Rel_fgap are the objective value and the relative feasibility gap, respectively. As we can see from Table 1, Algorithm 3

[1] **B**it **E**rror **R**ate function.

Table 1 The performance information and result of Example 7.2.3

Scenarios		P13	P23	P33	P43	P53	P63	P73	P83	P93
$[M, N]$		[477, 12]	[477, 12]	[477, 12]	[477, 12]	[1147, 6]	[224, 7]	[224, 7]	[224, 7]	[1147, 12]
Number of variables		5724	5724	5724	5724	6882	1568	1568	448	13764
Iter	Algorithm 3	**36**	**31**	**35**	**30**	**179**	**33**	**20**	**25**	**20**
	ADMM-v3	284	239	299	173	806	47	122	131	1377
	PCBDM	309	335	337	276	3103	186	135	420	529
	EPBDM	326	409	370	347	1242	166	106	438	114
Cpu_time [s]	Algorithm 3	0.294	0.216	0.253	0.248	1.559	0.089	0.044	0.025	0.481
	ADMM-v3	1.968	1.608	1.968	1.243	6.769	0.098	0.241	0.113	25.120
	PCBDM	1.957	2.186	2.211	1.896	24.479	0.363	0.267	0.350	8.847
	EPBDM	2.084	2.705	2.734	2.226	11.893	0.324	0.241	0.352	2.047
Obj_val	Algorithm 3	3365.508	3787.826	3340.921	3384.034	3397.298	873.976	1039.686	445.135	1565.422
	ADMM-v3	3560.602	3784.410	3329.525	3383.697	3353.029	875.427	1037.106	446.438	1566.989
	PCBDM	3567.030	3787.604	3341.863	3384.884	3399.864	876.572	1040.370	447.245	1567.579
	EPBDM	3566.794	3787.303	3341.517	3384.103	3399.040	876.631	1040.364	447.227	1567.543
Rel_fgap $\times 10^{-4}$	Algorithm 3	6.15	9.60	7.52	7.12	9.42	0.16	5.11	5.14	0.15
	ADMM-v3	9.43	8.65	7.76	9.84	6.53	5.06	4.84	8.04	9.75
	PCBDM	9.54	9.62	9.64	9.74	9.96	8.99	8.80	9.74	9.84
	EPBDM	9.61	9.68	9.64	9.67	9.96	9.19	8.79	9.72	9.84

shows the best performance both in terms of number of iterations and computational time in all the scenarios of this example.

8 Conclusions

In this paper, three new algorithms for large-scale separable convex optimization have been proposed. Their convergence has been proved and the worst-case complexity bound has been given. The main advantage of these algorithms is their ability to automatically update the smoothness parameters without any tuning strategy. This allows the algorithms to control the step-size of the search direction at each iteration. Consequently, they generate a larger step at the first iterations instead of remaining fixed for all iterations as in the algorithm proposed in [22]. Although the global convergence rate is still sub-linear, the computational results are remarkable, especially when the number of variables as well as the number of nodes increase. From a theoretical point of view, the algorithms possess a good performance behavior, due to the use of adaptive strategies. Currently, the numerical results are still preliminary, however we believe that the theory presented in this paper is useful and may provide a guidance for practitioners in some classes of problems. Moreover, the steps of the algorithms are rather simple so they can easily be implemented in practice.

Acknowledgements The authors would like to thank Dr. Ion Necoara and Dr. Michel Baes for useful comments on the text and for pointing out some interesting references. Furthermore, the authors are grateful to Dr. Paschalis Tsiaflakis for providing the problem data in the last numerical example.

Research supported by Research Council KUL: CoE EF/05/006 Optimization in Engineering (OPTEC), IOF-SCORES4CHEM, GOA/10/009 (MaNet), GOA /10/11, several PhD/postdoc and fellow grants; Flemish Government: FWO: PhD/postdoc grants, projects G.0452.04, G.0499.04, G.0211.05, G.0226.06, G.0321.06, G.0302.07, G.0320.08, G.0558.08, G.0557.08, G.0588.09, G.0377.09, G.0712.11, research communities (ICCoS, ANMMM, MLDM); IWT: PhD Grants, Belgian Federal Science Policy Office: IUAP P6/04; EU: ERNSI; FP7-HDMPC, FP7-EMBOCON, ERC-HIGHWIND, Contract Research: AMINAL. Other: Helmholtz-viCERP, COMET-ACCM.

Appendix: The proofs of technical lemmas

This appendix provides the proofs of two technical lemmas stated in the previous sections.

Proof of Lemma 4 The proof of this lemma is very similar to Lemma 3 in [27].

Proof Let $\hat{y} := y^*(\hat{x};\beta_2) := \frac{1}{\beta_2}(A\hat{x} - b)$. Then it follows from (18) that:

$$\begin{aligned}\psi(x;\beta_2) &\overset{(18)}{\leq} \psi(\hat{x};\beta_2) + \nabla_{x_1}\psi(\hat{x};\beta_2)^T(x_1 - \hat{x}_1) + \nabla_{x_2}\psi(\hat{x};\beta_2)^T(x_2 - \hat{x}_2)\\ &\quad + \frac{L_1^{\psi}(\beta_2)}{2}\|x_1 - \hat{x}_1\|^2 + \frac{L_2^{\psi}(\beta_2)}{2}\|x_2 - \hat{x}_2\|^2\\ &\overset{\text{def. } \psi(\cdot;\beta_2)}{=} \frac{1}{2\beta_2}\|A\hat{x} - b\|^2 + \hat{y}^T A_1(x_1 - \hat{x}_1) + \hat{y}^T A_2(x_2 - \hat{x}_2)\end{aligned}$$

$$
\begin{aligned}
&+ \frac{L_1^{\psi}(\beta_2)}{2}\|x_1 - \hat{x}_1\|^2 + \frac{L_2^{\psi}(\beta_2)}{2}\|x_2 - \hat{x}_2\|^2 \\
= \quad & \hat{y}^T(Ax - b) - \frac{1}{2\beta_2}\|A\hat{x} - b\|^2 + \frac{L_1^{\psi}(\beta_2)}{2}\|x_1 - \hat{x}_1\|^2 \\
&+ \frac{L_2^{\psi}(\beta_2)}{2}\|x_2 - \hat{x}_2\|^2. \qquad (77)
\end{aligned}
$$

By using the expression $f(x;\beta_2) = \phi(x) + \psi(x;\beta_2)$, the definition of $\bar{x}$, the condition (26) and (77) we have:

$$
\begin{aligned}
f(\bar{x};\beta_2) &\overset{(77)}{\leq} \phi(\bar{x}) + \bar{y}^T A_1\big(\bar{x}_1 - x_1^c\big) + \bar{y}^T A_2\big(\bar{x}_2 - x_2^c\big) \\
&\quad + \frac{L_1^{\psi}(\beta_2)}{2}\big\|\bar{x}_1 - x_1^c\big\|^2 + \frac{L_1^{\psi}(\beta_2)}{2}\big\|\bar{x}_2 - x_2^c\big\|^2 + \frac{1}{2\beta_2}\big\|Ax^c - b\big\|^2 \\
&\overset{(25)}{=} \min_{x\in X}\left\{\phi(x) + \frac{1}{\beta_2}\big\|Ax^c - b\big\|^2 + \bar{y}^T A\big(x - x^c\big)\right. \\
&\quad \left. + \frac{L_1^{\psi}(\beta_2)}{2}\big\|x_1 - x_1^c\big\|^2 + \frac{L_2^{\psi}(\beta_2)}{2}\big\|x_2 - x_2^c\big\|^2\right\} - \frac{1}{2\beta_2}\big\|Ax^c - b\big\|^2 \\
&= \min_{x\in X}\left\{\phi(x) + \bar{y}^T(Ax - b) + \frac{L_1^{\psi}(\beta_2)}{2}\big\|x_1 - x_1^c\big\|^2 + \frac{L_1^{\psi}(\beta_2)}{2}\big\|x_2 - x_2^c\big\|^2\right\} \\
&\quad - \frac{1}{2\beta_2}\big\|Ax^c - b\big\|^2 \\
&\overset{(26)}{\leq} \min_{x\in X}\big\{\phi(x) + \bar{y}^T(Ax - b) + \beta_1\big[p_1(x_1) + p_2(x_2)\big]\big\} - \frac{1}{2\beta_2}\big\|Ax^c - b\big\|^2 \\
&= d(\bar{y};\beta_1) - \frac{1}{2\beta_2}\big\|Ax^c - b\big\|^2 \leq d(\bar{y};\beta_1),
\end{aligned}
$$

which is indeed the condition (21). □

Proof of Lemma 7 Let us define $\xi(t) := \frac{2}{\sqrt{1+4/t^2}+1}$. It is easy to show that ξ is increasing in $(0,1)$. Moreover, $\tau_{k+1} = \xi(\tau_k)$ for all $k \geq 0$. Let us introduce $u := 2/t$. Then, we can show that $\frac{2}{u+2} < \xi(\frac{2}{u}) < \frac{2}{u+1}$. By using this inequalities and the increase of ξ in $(0,1)$, we have:

$$
\frac{\tau_0}{1 + 2\tau_0 k} \equiv \frac{2}{u_0 + 2k} < \tau_k < \frac{2}{u_0 + k} \equiv \frac{2\tau_0}{2 + \tau_0 k}. \qquad (78)
$$

Now, by the update rule (58), at each iteration k, we only either update β_1^k or β_2^k. Hence, it implies that:

$$
\begin{aligned}
\beta_1^k &= (1 - \tau_0)(1 - \tau_2)\cdots(1 - \tau_{2\lfloor k/2\rfloor})\beta_1^0, \\
\beta_2^k &= (1 - \tau_1)(1 - \tau_3)\cdots(1 - \tau_{2\lfloor k/2\rfloor - 1})\beta_2^0,
\end{aligned} \qquad (79)
$$

where $\lfloor x \rfloor$ is the largest integer number which is less than or equal to the positive real number x. On the other hand, since $\tau_{i+1} < \tau_i$ for $i \geq 0$, for any $l \geq 0$, it implies:

$$\begin{aligned}(1-\tau_0)\prod_{i=0}^{2l}(1-\tau_i) &< \big[(1-\tau_0)(1-\tau_2)\cdots(1-\tau_{2l})\big]^2 < \prod_{i=0}^{2l+1}(1-\tau_i), \quad \text{and} \\ \prod_{i=0}^{2l-1}(1-\tau_i) &< \big[(1-\tau_1)(1-\tau_3)\cdots(1-\tau_{2l-1})\big]^2 < (1-\tau_0)^{-1}\prod_{i=0}^{2l}(1-\tau_i).\end{aligned} \tag{80}$$

Note that $\prod_{i=0}^{k}(1-\tau_i) = \frac{(1-\tau_0)}{\tau_0^2}\tau_k^2$, it follows from (79) and (80) for $k \geq 1$ that:

$$\begin{aligned}\frac{(1-\tau_0)\beta_1^0}{\tau_0}\tau_{k+1} < \beta_1^{k+1} < \frac{\beta_1^0\sqrt{1-\tau_0}}{\tau_0}\tau_{k-1}, \quad \text{and} \\ \frac{\beta_2^0\sqrt{1-\tau_0}}{\tau_0}\tau_{k+1} < \beta_2^{k+1} < \frac{\beta_2^0}{\tau_0}\tau_{k-1}.\end{aligned}$$

By combining these inequalities and (78), and noting that $\tau_0 \in (0,1)$, we obtain (59). □

References

1. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and Distributed Computation: Numerical Methods. Prentice Hall, New York (1989)
2. Boyd, S., Parikh, N., Chu, E., Peleato, B.: Distributed optimization and statistics via alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2011)
3. Chen, G., Teboulle, M.: A proximal-based decomposition method for convex minimization problems. Math. Program. **64**, 81–101 (1994)
4. Connejo, A.J., Mínguez, R., Castillo, E., García-Bertrand, R.: Decomposition Techniques in Mathematical Programming: Engineering and Science Applications. Springer, Berlin (2006)
5. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
6. Duchi, J.C., Agarwal, A., Wainwright, M.J.: Dual averaging for distributed optimization: Convergence analysis and network scaling. IEEE Trans. Autom. Control **57**(3), 592–606 (2012)
7. Eckstein, J., Bertsekas, D.: On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators. Math. Program. **55**, 293–318 (1992)
8. Facchinei, F., Pang, J.-S.: Finite-Dimensional Variational Inequalities and Complementarity Problems, vols. 1–2. Springer, Berlin (2003)
9. Goldfarb, D., Ma, S.: Fast multiple splitting algorithms for convex optimization. SIAM J. Optim. **22**(2), 533–556 (2012)
10. Hamdi, A.: Two-level primal-dual proximal decomposition technique to solve large-scale optimization problems. Appl. Math. Comput. **160**, 921–938 (2005)
11. Han, S.P., Lou, G.: A parallel algorithm for a class of convex programs. SIAM J. Control Optim. **26**, 345–355 (1988)
12. Hariharan, L., Pucci, F.D.: Decentralized resource allocation in dynamic networks of agents. SIAM J. Optim. **19**(2), 911–940 (2008)
13. He, B.S., Tao, M., Xu, M.H., Yuan, X.M.: Alternating directions based contraction method for generally separable linearly constrained convex programming problems. Optimization (2011). doi:10.1080/02331934.2011.611885
14. He, B.S., Yang, H., Wang, S.L.: Alternating directions method with self-adaptive penalty parameters for monotone variational inequalities. J. Optim. Theory Appl. **106**, 349–368 (2000)

15. He, B.S., Yuan, X.M.: On the $O(1/n)$ convergence rate of the Douglas–Rachford alternating direction method. SIAM J. Numer. Anal. **50**, 700–709 (2012)
16. Holmberg, K.: Experiments with primal-dual decomposition and subgradient methods for the uncapacitated facility location problem. Optimization **49**(5–6), 495–516 (2001)
17. Holmberg, K., Kiwiel, K.C.: Mean value cross decomposition for nonlinear convex problem. Optim. Methods Softw. **21**(3), 401–417 (2006)
18. Kojima, M., Megiddo, N., Mizuno, S., et al.: Horizontal and vertical decomposition in interior point methods for linear programs. Technical report, Information Sciences, Tokyo Institute of Technology, Tokyo (1993)
19. Lenoir, A., Mahey, P.: Accelerating convergence of a separable augmented Lagrangian algorithm. Technical report, LIMOS/RR-07-14, pp. 1–34 (2007)
20. Love, R.F., Kraemer, S.A.: A dual decomposition method for minimizing transportation costs in multifacility location problems. Transp. Sci. **7**, 297–316 (1973)
21. Mehrotra, S.: On the implementation of a primal-dual interior point method. SIAM J. Optim. **2**(4), 575–601 (1992)
22. Necoara, I., Suykens, J.A.K.: Applications of a smoothing technique to decomposition in convex optimization. IEEE Trans. Autom. Control **53**(11), 2674–2679 (2008)
23. Nedíc, A., Ozdaglar, A.: Distributed subgradient methods for multi-agent optimization. IEEE Trans. Autom. Control **54**, 48–61 (2009)
24. Nesterov, Y.: A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$. Dokl. Akad. Nauk SSSR **269**, 543–547 (1983) (Translated as Soviet Math. Dokl.)
25. Nesterov, Y.: Introductory Lectures on Convex Optimization: A Basic Course. Applied Optimization, vol. 87. Kluwer Academic, Dordrecht (2004)
26. Nesterov, Y.: Excessive gap technique in nonsmooth convex minimization. SIAM J. Optim. **16**(1), 235–249 (2005)
27. Nesterov, Y.: Smooth minimization of non-smooth functions. Math. Program. **103**(1), 127–152 (2005)
28. Neveen, G., Jochen, K.: Faster and simpler algorithms for multicommodity flow and other fractional packing problems. SIAM J. Comput. **37**(2), 630–652 (2007)
29. Ruszczyński, A.: On convergence of an augmented Lagrangian decomposition method for sparse convex optimization. Math. Oper. Res. **20**, 634–656 (1995)
30. Samar, S., Boyd, S., Gorinevsky, D.: Distributed estimation via dual decomposition. In: Proceedings European Control Conference (ECC), Kos, Greece, pp. 1511–1516 (2007)
31. Spingarn, J.E.: Applications of the method of partial inverses to convex programming: decomposition. Math. Program. Ser. A **32**, 199–223 (1985)
32. Tran-Dinh, Q., Necoara, I., Savorgnan, C., Diehl, M.: An inexact perturbed path-following method for Lagrangian decomposition in large-scale separable convex optimization. Int. Report 12-181, ESAT-SISTA, KU Leuven, Belgium (2012). SIAM J. Optim., accepted
33. Tseng, P.: Alternating projection-proximal methods for convex programming and variational inequalities. SIAM J. Optim. **7**(4), 951–965 (1997)
34. Tsiaflakis, P., Diehl, M., Moonen, M.: Distributed spectrum management algorithms for multi-user DSL networks. IEEE Trans. Signal Process. **56**(10), 4825–4843 (2008)
35. Tsiaflakis, P., Necoara, I., Suykens, J.A.K., Moonen, M.: Improved dual decomposition based optimization for DSL dynamic spectrum management. IEEE Trans. Signal Process. **58**(4), 2230–2245 (2010)
36. Vania, D.S.E.: Finding approximate solutions for large scale linear programs. Ph.D. Thesis, No. 18188, ETH, Zurich (2009)
37. Venkat, A.N.: Distributed model predictive control: theory and applications. Ph.D. Thesis, University of Wisconsin-Madison (2006)
38. Wächter, A., Biegler, L.T.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)
39. Zhao, G.: A Lagrangian dual method with self-concordant barriers for multistage stochastic convex programming. Math. Program. **102**, 1–24 (2005)