J Glob Optim (2014) 59:59–80
DOI 10.1007/s10898-013-0085-7

Path-following gradient-based decomposition algorithms for separable convex optimization

Quoc Tran Dinh · Ion Necoara · Moritz Diehl

Received: 14 October 2012 / Accepted: 13 June 2013 / Published online: 22 June 2013

Abstract A new decomposition optimization algorithm, called *path-following gradient-based decomposition*, is proposed to solve separable convex optimization problems. Unlike path-following Newton methods considered in the literature, this algorithm does not require any smoothness assumption on the objective function. This allows us to handle more general classes of problems arising in many real applications than in the path-following Newton methods. The new algorithm is a combination of three techniques, namely smoothing, Lagrangian decomposition and path-following gradient framework. The algorithm decomposes the original problem into smaller subproblems by using dual decomposition and smoothing via self-concordant barriers, updates the dual variables using a path-following gradient method and allows one to solve the subproblems in parallel. Moreover, compared to augmented Lagrangian approaches, our algorithmic parameters are updated automatically without any tuning strategy. We prove the global convergence of the new algorithm and analyze its convergence rate. Then, we modify the proposed algorithm by applying Nesterov's

Q. Tran Dinh (✉) · M. Diehl
Optimization in Engineering Center (OPTEC) and Department of Electrical Engineering,
Katholieke Universiteit Leuven, Leuven, Belgium
e-mail: quoc.trandinh@epfl.ch

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

Present address
Q. Tran Dinh
Laboratory for Information and Inference Systems (LIONS),
EPFL, Lausanne, Switzerland

I. Necoara
Automatic Control and Systems Engineering Department,
University Politehnica Bucharest, 060042 Bucharest, Romania
e-mail: ion.necoara@acse.pub.ro

Q. Tran Dinh
Department of Mathematics–Mechanics–Informatics,
Vietnam National University, Hanoi, Vietnam

accelerating scheme to get a new variant which has a better convergence rate than the first algorithm. Finally, we present preliminary numerical tests that confirm the theoretical development.

Keywords Path-following gradient method · Dual fast gradient algorithm · Separable convex optimization · Smoothing technique · Self-concordant barrier · Parallel implementation

1 Introduction

Many optimization problems arising in engineering and economics can conveniently be formulated as *Separable Convex Programming Problems* (SepCP). Particularly, optimization problems related to a network $\mathcal{N}(\mathcal{V}, \mathcal{E})$ of N agents, where $\mathcal{V}$ denotes the set of nodes and $\mathcal{E}$ denotes the set of edges in the network, can be cast as separable convex optimization problems. Mathematically, an (SepCP) can be expressed as follows:

$$\phi^* := \left\{ \begin{array}{ll} \max\limits_{x} & \left\{ \phi(x) := \sum\limits_{i=1}^{N} \phi_i(x_i) \right\}, \\ \text{s.t.} & \sum\limits_{i=1}^{N} (A_i x_i - b_i) = 0, \\ & x_i \in X_i, i = 1, \ldots, N, \end{array} \right. \tag{SepCP}$$

where the decision variable $x := (x_1, \ldots, x_N)$ with $x_i \in \mathbb{R}^{n_i}$, the function $\phi_i : \mathbb{R}^{n_i} \to \mathbb{R}$ is concave and the feasible set is described by the set $X := X_1 \times \cdots \times X_N$, with $X_i \in \mathbb{R}^{n_i}$ being nonempty, closed and convex for all $i = 1, \ldots, N$. Let us denote $A := [A_1, \ldots, A_N]$, with $A_i \in \mathbb{R}^{m \times n_i}$ for $i = 1, \ldots, N$, $b := \sum_{i=1}^{N} b_i \in \mathbb{R}^m$ and $n_1 + \cdots + n_N = n$. The constraint $Ax - b = 0$ in (SepCP) is called a *coupling linear constraint*, while $x_i \in X_i$ are referred to as *local constraints* of the i-th component (agent).

Several applications of (SepCP) can be found in the literature such as distributed control, network utility maximization, resource allocation, machine learning and multistage stochastic convex programming [1,2,11,17,21,22]. Problems of moderate size or possessing a sparse structure can be solved by standard optimization methods in a centralized setup. However, in many real applications we meet problems, which may not be solvable by standard optimization approaches or by exploiting problem structures, e.g. nonsmooth separate objective functions, dynamic structure or distributed information. In those situations, decomposition methods can be considered as an appropriate framework to tackle the underlying optimization problem. Particularly, the Lagrangian dual decomposition is one technique widely used to decompose a large-scale separable convex optimization problem into smaller subproblem components, which can simultaneously be solved in a *parallel manner* or in a *closed form*.

Various approaches have been proposed to solve (SepCP) in decomposition frameworks. One class of algorithms is based on Lagrangian relaxation and subgradient-type methods of multipliers [1,5,13]. However, it has been observed that subgradient methods are usually slow and numerically sensitive to the choice of step sizes in practice [14]. The second approach relies on augmented Lagrangian functions, see e.g. [7,8,18]. Many variants were proposed to process the inseparability of the crossproduct terms in the augmented Lagrangian function in different ways. Another research direction is based on alternating direction methods which were studied, for example, in [2]. Alternatively, proximal point-type methods were extended

to the decomposition framework, see, e.g. [3,11]. Other researchers employed interior point methods in the framework of (dual) decomposition such as [9,12,19,22].

In this paper, we follow the same line of the dual decomposition framework but in a different way. First, we smooth the dual function by using self-concordant barriers as in [11,19]. With an appropriate choice of the smoothness parameter, we show that the dual function of the smoothed problem is an approximation of the original dual function. Then, we develop a new path-following gradient decomposition method for solving the smoothed dual problem. By strong duality, we can also recover an approximate solution for the original problem. Compared to the previous related methods mentioned above, the new approach has the following advantages. Firstly, since the feasible set of the problem only depends on the parameter of its self-concordant barrier, this allows us to avoid a dependence on the diameter of the feasible set as in prox-function smoothing techniques [11,20]. Secondly, the proposed method is a gradient-type scheme which allows us to handle more general classes of problems than in path-following Newton-type methods [12,19,22], in particular, those with nonsmoothness of the objective function. Thirdly, by smoothing via self-concordant barrier functions, instead of solving the primal subproblems as general convex programs as in [3,7,11,20] we can treat them by using their optimality condition. Nevertheless, solving this condition is equivalent to solving a nonlinear equation or a generalized equation system. Finally, by convergence analysis, we provide an automatical update rule for all the algorithmic parameters.

Contribution The contribution of the paper can be summarized as follows:

(a) We propose using a smoothing technique via barrier function to smooth the dual function of (SepCP) as in [9,12,22]. However, we provide a new estimate for the dual function, see Lemma 1.
(b) We propose a new path-following gradient-based decomposition algorithm, Algorithm 1, to solve (SepCP). This algorithm allows one to solve the primal subproblems formed from the components of (SepCP) in parallel. Moreover, all the algorithmic parameters are updated automatically without using any tuning strategy.
(c) We prove the convergence of the algorithm and estimate its local convergence rate.
(d) Then, we modify the algorithm by applying Nesterov's accelerating scheme for solving the dual to obtain a new variant, Algorithm 2, which possesses a better convergence rate than the first algorithm. More precisely, this convergence rate is $\mathcal{O}(1/\varepsilon)$, where ε is a given accuracy.

Let us emphasize the following points. The new estimate of the dual function considered in this paper is different from the one in [19] which does not depend on the diameter of the feasible set of the dual problem. The worst-case complexity of the second algorithm is $\mathcal{O}(1/\varepsilon)$ which is much higher than in subgradient-type methods of multipliers [1,5,13]. We note that this convergence rate is optimal in the sense of Nesterov's optimal schemes [6,14] applying to dual decomposition frameworks. Both algorithms developed in this paper can be implemented in a *parallel manner*.

Outline The rest of this paper is organized as follows. In the next section, we recall the Lagrangian dual decomposition framework in convex optimization. Section 3 considers a smoothing technique via self-concordant barriers and provides an estimate for the dual function. The new algorithms and their convergence analysis are presented in Sects. 4 and 5. Preliminarily numerical results are shown in the last section to verify our theoretical results.

Notation and terminology Throughout the paper, we work on the Euclidean space $\mathbb{R}^n$ endowed with an inner product $x^T y$ for $x, y \in \mathbb{R}^n$. The Euclidean norm is $\|x\|_2 := \sqrt{x^T x}$ which associates with the given inner product. For a proper, lower semicontinuous convex function f, $\partial f(x)$ denotes the subdifferential of f at x. If f is concave, then we also use $\partial f(x)$ for its super-differential at x. For any $x \in \mathrm{dom}(f)$ such that $\nabla^2 f(x)$ is positive definite, the local norm of a vector u with respect to f at x is defined as $\|u\|_x := \left[u^T \nabla^2 f(x) u\right]^{1/2}$ and its dual norm is $\|u\|_x^* := \max\left\{u^T v \mid \|v\|_x \le 1\right\} = \left[u^T \nabla^2 f(x)^{-1} u\right]^{1/2}$. It is obvious that $u^T v \le \|u\|_x \|v\|_x^*$. The notation $\mathbb{R}_+$ and $\mathbb{R}_{++}$ define the sets of nonnegative and positive real numbers, respectively. The function $\omega : \mathbb{R}_+ \to \mathbb{R}$ is defined by $\omega(t) := t - \ln(1+t)$ and its dual function $\omega_* : [0, 1) \to \mathbb{R}$ is $\omega_*(t) := -t - \ln(1-t)$.

2 Lagrangian dual decomposition in convex optimization

Let $\mathscr{L}(x, y) := \phi(x) + y^T(Ax - b)$ be the partial Lagrangian function associated with the coupling constraint $Ax - b = 0$ of (SepCP). The dual problem of (SepCP) is written as

$$g^* := \min_{y \in \mathbb{R}^m} g(y), \tag{1}$$

where g is the dual function defined by

$$g(y) := \max_{x \in X} \mathscr{L}(x, y) = \max_{x \in X} \left\{\phi(x) + y^T(Ax - b)\right\}. \tag{2}$$

Due to the separability of ϕ, the dual function g can be computed *in parallel* as

$$g(y) = \sum_{i=1}^{N} g_i(y), \quad g_i(y) := \max_{x_i \in X_i} \left\{\phi_i(x_i) + y^T(A_i x_i - b_i)\right\}, \quad i = 1, \ldots, N. \tag{3}$$

Throughout this paper, we require the following fundamental assumptions:

Assumption A.1 The following assumptions hold, see [18]:

(a) The solution set X^* of (SepCP) is nonempty.
(b) Either X is polyhedral or the following Slater qualification condition holds

$$\mathrm{ri}(X) \cap \{x \mid Ax - b = 0\} \neq \emptyset, \tag{4}$$

where $\mathrm{ri}(X)$ is the relative interior of X.
(c) The functions ϕ_i, $i = 1, \ldots, N$, are proper, upper semicontinuous and concave and A is full-row rank.

Assumption A.1 is standard in convex optimization. Under this assumption, *strong duality* holds, i.e. the dual problem (1) is also solvable and $g^* = \phi^*$. Moreover, the set of Lagrange multipliers, Y^*, is bounded. However, under Assumption A.1, the dual function g may not be differentiable. Numerical methods such as subgradient-type and bundle methods can be used to solve (1). Nevertheless, these methods are in general numerically intractable and slow [14].

3 Smoothing via self-concordant barrier functions

In many practical problems, the feasible sets $X_i, i = 1, \ldots, N$ are usually simple, e.g. box, polyhedra and ball. Hence, X_i can be endowed with a *self-concordant barrier* (see, e.g. [14,15]) as in the following assumption.

Assumption A.2 Each feasible set $X_i, i = 1, \ldots, N$, is bounded and endowed with a self-concordant barrier function F_i with the parameter $\nu_i > 0$.

Note that the assumption on the boundedness of X_i can be relaxed by assuming that the set of sample points generated by the new algorithm described below is bounded.

Remark 1 The theory developed in this paper can be easily extended to the case X_i given as follows, see [12], for some $i \in \{1, \ldots, N\}$:

$$X_i := X_i^c \cap X_i^a, \quad X_i^a := \left\{x_i \in \mathbb{R}^{n_i} \mid D_i x_i = d_i\right\}, \tag{5}$$

by applying the standard linear algebra routines, where the set X_i^c has nonempty interior and associated with a ν_i-self-concordant barrier F_i. If, for some $i \in \{1, \ldots, M\}$, $X_i := X_i^c \cap X_i^g$, where X_i^g is a general convex set, then we can remove X_i^g from the set of constraints by adding the indicator function $\delta_{X_i^g}(\cdot)$ of this set to the objective function component ϕ_i, i.e. $\hat{\phi}_i := \phi_i + \delta_{X_i^g}$ (see [16]).

Let us denote by x_i^c the analytic center of X_i, i.e.

$$x_i^c := \arg\min_{x_i \in \text{int}(X_i)} F_i(x_i) \; \forall i = 1, \ldots, N, \tag{6}$$

where $\text{int}(X_i)$ is the interior of X_i. Since X_i is bounded, x_i^c is well-defined [14]. Moreover, the following estimates hold

$$F_i(x_i) - F_i(x_i^c) \geq \omega(\|x_i - x_i^c\|_{x_i^c}) \quad \text{and} \quad \|x_i - x_i^c\|_{x_i^c} \leq \nu_i + 2\sqrt{\nu_i}, \quad \forall x_i \in X_i, \; i = 1, \ldots, N. \tag{7}$$

Without loss of generality, we can assume that $F_i(x_i^c) = 0$. Otherwise, we can replace F_i by $\tilde{F}_i(\cdot) := F_i(\cdot) - F_i(x_i^c)$ for $i = 1, \ldots, N$. Since X is separable, $F := \sum_{i=1}^N F_i$ is a self-concordant barrier of X with the parameter $\nu := \sum_{i=1}^N \nu_i$.

Let us define the following function

$$g(y; t) := \sum_{i=1}^N g_i(y; t), \tag{8}$$

where

$$g_i(y; t) := \max_{x_i \in \text{int}(X_i)} \left\{\phi_i(x_i) + y^T(A_i x_i - b_i) - t F_i(x_i)\right\}, \quad i = 1, \ldots, N, \tag{9}$$

with $t > 0$ being referred to as a *smoothness parameter*. Note that the maximum problem in (9) has a unique optimal solution, which is denoted by $x_i^*(y; t)$, due to the strict concavity of the objective function. We call this problem the *primal subproblem*. Consequently, the functions $g_i(\cdot, t)$ and $g(\cdot, t)$ are well-defined and smooth on $\mathbb{R}^m$ for any $t > 0$. We also call $g_i(\cdot; t)$ and $g(\cdot; t)$ the *smoothed dual function* of g_i and g, respectively.

The optimality condition for (9) is written as

$$0 \in \partial\phi_i(x_i^*(y; t)) + A_i^T y - t\nabla F_i(x_i^*(y; t)), \quad i = 1, \ldots, N. \tag{10}$$

We note that (10) represents a system of *generalized equations*. Particularly, if ϕ_i is differentiable for some $i \in \{1, \ldots, N\}$, then the condition (10) collapses to $\nabla\phi_i(x_i^*(y;t)) + A_i^T y - t\nabla F_i(x_i^*(y;t)) = 0$, which is indeed a *system of nonlinear equations*. Since problem (9) is convex, the condition (10) is necessary and sufficient for optimality. Let us define the full optimal solution $x^*(y;t) := (x_1^*(y;t), \cdots, x_N^*(y;t))$. The gradients of $g_i(\cdot;t)$ and $g(\cdot;t)$ are given, respectively by

$$\nabla g_i(y;t) = A_i x_i^*(y;t) - b_i, \quad \nabla g(y;t) = Ax^*(y;t) - b. \tag{11}$$

Next, we show the relation between the smoothed dual function $g(\cdot;t)$ and the original dual function $g(\cdot)$ for a sufficiently small $t > 0$.

Lemma 1 *Suppose that Assumptions A.1 and A.2 are satisfied. Let $\bar{x}$ be a strictly feasible point for problem (SepCP), i.e. $\bar{x} \in int(X) \cap \{x \mid Ax = b\}$. Then, for any $t > 0$ we have*

$$g(y) - \phi(\bar{x}) \geq 0 \ \text{ and } \ g(y;t) + tF(\bar{x}) - \phi(\bar{x}) \geq 0. \tag{12}$$

Moreover, the following estimate holds

$$g(y;t) \leq g(y) \leq g(y;t) + t(\nu + F(\bar{x})) + 2\sqrt{t\nu}\,[g(y;t) + tF(\bar{x}) - \phi(\bar{x})]^{1/2}. \tag{13}$$

Proof The first two inequalities in (12) are trivial due to the definitions of $g(\cdot)$, $g(\cdot;t)$ and the feasibility of $\bar{x}$. We only prove (13). Indeed, since $\bar{x} \in \text{int}(X)$ and $x^*(y) \in X$, if we define $x_\tau^*(y) := \bar{x} + \tau(x^*(y) - \bar{x})$, then $x_\tau^*(y) \in \text{int}(X)$ if $\tau \in [0,1)$. By applying the inequality [15, 2.3.3] we have

$$F(x_\tau^*(y)) \leq F(\bar{x}) - \nu\ln(1-\tau).$$

Using this inequality together with the definition of $g(\cdot;t)$, the concavity of ϕ, $A\bar{x} = b$ and $g(y) = \phi(x^*(y)) + y^T[Ax^*(y) - b]$, we deduce that

$$\begin{aligned}
g(y;t) &= \max_{x\in\text{int}(X)} \left\{\phi(x) + y^T(Ax - b) - tF(x)\right\} \\
&\geq \max_{\tau\in[0,1)} \left\{\phi(x_\tau^*(y)) + y^T(Ax_\tau(y) - b) - tF(x_\tau^*(y))\right\} \\
&\geq \max_{\tau\in[0,1)} \Big\{(1-\tau)\left[\phi(\bar{x}) + (A\bar{x} - b)\right] \\
&\quad + \tau\left[\phi(x^*(y) + y^T(Ax^*(y) - b)\right] - tF(x_\tau^*(y))\Big\} \\
&\geq \max_{\tau\in[0,1)} \{(1-\tau)\phi(\bar{x}) + \tau g(y) + t\nu\ln(1-\tau)\} - tF(\bar{x}).
\end{aligned} \tag{14}$$

By solving the maximization problem on the right hand side of (14) and then rearranging the results, we obtain

$$g(y) \leq g(y;t) + t[\nu + F(\bar{x})] + t\nu\Big[\ln\Big(\frac{g(y) - \phi(\bar{x})}{t\nu}\Big)\Big]_+, \tag{15}$$

where $[\cdot]_+ := \max\{\cdot, 0\}$. Moreover, it follows from (14) that

$$\begin{aligned}
g(y) - \phi(\bar{x}) &\leq \frac{1}{\tau}\left[g(y;t) - \phi(\bar{x}) + tF(\bar{x}) + t\nu\ln\left(1 + \frac{\tau}{1-\tau}\right)\right] \\
&\leq \frac{1}{\tau}\Big[g(y;t) - \phi(\bar{x}) + tF(\bar{x})\Big] + \frac{t\nu}{1-\tau}.
\end{aligned}$$

If we minimize the right hand side of this inequality on $[0, 1)$, then we get $g(y) - \phi(\bar{x}) \leq [(g(y;t) - \phi(\bar{x}) + tF(\bar{x}))^{1/2} + \sqrt{t\nu}]^2$. Finally, we plug this inequality into (15) to obtain

$$\begin{aligned} g(y) &\leq g(y;t) + t\nu + 2t\nu \ln\left(1 + \sqrt{\frac{[g(y;t) - \phi(\bar{x}) + tF(\bar{x}]}{t\nu}}\right) + tF(\bar{x}) \\ &\leq g(y;t) + t\nu + tF(\bar{x}) + 2\sqrt{t\nu}\,[g(y;t) - \phi(\bar{x}) + tF(\bar{x})]^{1/2}, \end{aligned}$$

which is indeed (13). □

Remark 2 (*Approximation of g*) It follows from (13) that $g(y) \leq (1 + 2\sqrt{t\nu})g(y;t) + t(\nu + F(\bar{x})) + 2\sqrt{t\nu}(tF(\bar{x}) - \phi(\bar{x}))$. Hence, $g(y;t) \to g(y)$ as $t \to 0^+$. Moreover, this estimate is different from the one in [19], since we do not assume that the feasible set of the dual problem (1) is bounded.

Now, we consider the following minimization problem which we call the *smoothed dual problem* to distinguish it from the original dual problem

$$g^*(t) := g(y^*(t);t) = \min_{y \in \mathbb{R}^m} g(y;t). \tag{16}$$

We denote by $y^*(t)$ the solution of (16). The following lemma shows the main properties of the functions $g(y;\cdot)$ and $g^*(\cdot)$.

Lemma 2 *Suppose that Assumptions A.1 and A.2 are satisfied. Then*

(a) The function $g(y;\cdot)$ is convex and nonincreasing on $\mathbb{R}_{++}$ for a given $y \in \mathbb{R}^m$. Moreover, we have:

$$g(y;\hat{t}) \geq g(y;t) - (\hat{t} - t)F(x^*(y;t)). \tag{17}$$

(b) The function $g^*(\cdot)$ defined by (16) is differentiable and nonincreasing on $\mathbb{R}_{++}$. Moreover, $g^*(t) \leq g^*$, $\lim_{t \downarrow 0^+} g^*(t) = g^* = \phi^*$ and $x^*(y^*(t);t)$ is feasible to the original problem (SepCP).

Proof We only prove (17), the proof of the remainders can be found in [12,19]. Indeed, since $g(y;\cdot)$ is convex and differentiable and $\frac{dg(y;t)}{dt} = -F(x^*(y;t)) \leq 0$, we have $g(y;\hat{t}) \geq g(y;t) + (\hat{t} - t)\frac{dg(y;t)}{dt} = g(y;t) - (\hat{t} - t)F(x^*(y;t))$. □

The statement (b) of Lemma 2 shows that if we find an approximate solution y^k for (16) for sufficiently small t_k, then $g^*(t_k)$ approximates g^* (recall that $g^* = \phi^*$) and $x^*(y^k;t_k)$ is approximately feasible to (SepCP).

4 Path-following gradient method

In this section we design a path-following gradient algorithm to solve the dual problem (1), analyze the convergence of the algorithm and estimate the local convergence rate.

4.1 The path-following gradient scheme

Since $g(\cdot;t)$ is strictly convex and smooth, we can write the optimality condition of (16) as

$$\nabla g(y;t) = 0. \tag{18}$$

This equation has a unique solution $y^*(t)$.

Now, for any given $x \in \text{int}(X)$, we note that $\nabla^2 F(x)$ is positive definite. We introduce a local norm of matrices as

$$|||A|||_x^* := \|A\nabla^2 F(x)^{-1}A^T\|_2, \tag{19}$$

The following lemma shows an important property of the function $g(\cdot; t)$.

Lemma 3 *Suppose that Assumptions A.1 and A.2 are satisfied. Then, for all $t > 0$ and $y, \hat{y} \in \mathbb{R}^m$, one has*

$$[\nabla g(y;t) - \nabla g(\hat{y};t)]^T(y-\hat{y}) \geq \frac{t\|\nabla g(y;t) - \nabla g(\hat{y};t)\|_2^2}{c_A\left[c_A + \|\nabla g(y,t) - \nabla g(\hat{y};t)\|_2\right]}, \tag{20}$$

where $c_A := |||A|||^*_{x^*(y;t)}$. Consequently, it holds hat

$$g(\hat{y};t) \leq g(y;t) + \nabla g(y;t)^T(\hat{y}-y) + t\omega_*(c_A t^{-1}\|\hat{y}-y\|_2), \tag{21}$$

provided that $c_A\|\hat{y}-y\|_2 < t$.

Proof For notational simplicity, we denote $x^* := x^*(y;t)$ and $\hat{x}^* := x^*(\hat{y};t)$. From the definition (11) of $\nabla g(\cdot;t)$ and the Cauchy–Schwarz inequality we have

$$[\nabla g(y;t) - \nabla g(\hat{y};t)]^T(y-\hat{y}) = (y-\hat{y})^T A(x^*-\hat{x}^*). \tag{22}$$

$$\|\nabla g(\hat{y};t) - \nabla g(y;t)\|_2 \leq |||A|||^*_{x^*}\|\hat{x}^*-x^*\|_{x^*}. \tag{23}$$

It follows from (10) that $A^T(y-\hat{y}) = t[\nabla F(x^*) - \nabla F(\hat{x}^*] - [\xi(x^*) - \xi(\hat{x}^*)]$, where $\xi(\cdot) \in \partial\phi(\cdot)$. By multiplying this relation with $x^*-\hat{x}^*$ and then using [14, Theorem 4.1.7] and the concavity of ϕ we obtain

$$\begin{aligned}(y-\hat{y})^T A(x^*-\hat{x}^*) &= t[\nabla F(x^*)-\nabla F(\hat{x}^*)]^T(x^*-\hat{x}^*) - [\xi(x^*)-\xi(\hat{x}^*)]^T(x^*-\hat{x}^*)\\ &\overset{\text{concavity of }\phi}{\geq} t[\nabla F(x^*) - \nabla F(\hat{x}^*)]^T(x^*-\hat{x}^*)\\ &\geq \frac{t\|x^*-\hat{x}^*\|^2_{x^*}}{1+\|x^*-\hat{x}^*\|_{x^*}}\\ &\overset{(23)}{\geq} \frac{t\left[\|\nabla g(y;t)-\nabla g(\hat{y};t)\|_2\right]^2}{|||A|||^*_{x^*}\left[|||A|||^*_{x^*} + \|\nabla g(y;t)-\nabla g(\hat{y};t)\|_2\right]}.\end{aligned}$$

Substituting this inequality into (22) we obtain (20).

By the Cauchy–Schwarz inequality, it follows from (20) that $\|\nabla g(\hat{y};t) - \nabla g(y;t)\| \leq \frac{c_A^2\|\hat{y}-y\|_2}{t-c_A\|\hat{y}-y\|}$, provided that $c_A\|\hat{y}-y\| \leq t$. Finally, by using the mean-value theorem, we have

$$\begin{aligned}g(\hat{y};t) &= g(y;t) + \nabla g(y;t)^T(\hat{y}-y) + \int_0^1 (\nabla g(y+s(\hat{y}-y);t) - \nabla g(y;t))^T(\hat{y}-y)ds\\ &\leq g(y;t) + \nabla g(y;t)^T(\hat{y}-y) + c_A\|\hat{y}-y\|_2\int_0^1 \frac{c_A s\|\hat{y}-y\|_2}{t-c_A s\|\hat{y}-y\|_2}ds\\ &= g(y;t) + \nabla g(y;t)^T(\hat{y}-y) + t\omega^*(c_A t^{-1}\|\hat{y}-y\|_2),\end{aligned}$$

which is indeed (21) provided that $c_A\|\hat{y}-y\|_2 < t$. □

Now, we describe one step of the path-following gradient method for solving (16). Let us assume that $y^k \in \mathbb{R}^m$ and $t_k > 0$ are the values at the current iteration $k \geq 0$, the values y^{k+1} and t_{k+1} at the next iteration are computed as

$$\begin{cases} t_{k+1} := t_k - \Delta t_k, \\ y^{k+1} := y^k - \alpha_k \nabla g(y^k, t_{k+1}), \end{cases} \tag{24}$$

where $\alpha_k := \alpha(y^k; t_k) > 0$ is the current step size and Δt_k is the decrement of the parameter t. In order to analyze the convergence of the scheme (24), we introduce the following notation

$$\tilde{x}_k^* := x^*(y^k; t_{k+1}), \quad \tilde{c}_A^k = \|\!|A|\!\|^*_{x^*(y^k;t_{k+1})} \quad \text{and} \quad \tilde{\lambda}_k := \|\nabla g(y^k; t_{k+1})\|_2. \tag{25}$$

First, we prove an important property of the *path-following gradient scheme* (24).

Lemma 4 *Under Assumptions A.1 and A.2, the following inequality holds*

$$g(y^{k+1}; t_{k+1}) \leq g(y^k; t_k) - \left[\alpha_k \tilde{\lambda}_k^2 - t_{k+1}\omega_*(\tilde{c}_A^k t_{k+1}^{-1}\alpha_k\tilde{\lambda}_k) - \Delta t_k F(\tilde{x}_k^*)\right], \tag{26}$$

where $\tilde{c}_A^k$ and $\tilde{\lambda}_k$ are defined by (25).

Proof Since $t_{k+1} = t_k - \Delta t_k$, by using (17) with t_k and t_{k+1}, we have

$$g(y^k; t_{k+1}) \leq g(y^k; t_k) + \Delta t_k F(x^*(y^k; t_{k+1})). \tag{27}$$

Next, by (21) we have $y^{k+1} - y^k = -\alpha_k \nabla g(y^k; t_{k+1})$ and $\tilde{\lambda}_k := \|\nabla g(y^k; t_{k+1})\|_2$. Hence, we can derive

$$g(y^{k+1}; t_{k+1}) \leq g(y^k; t_{k+1}) - \alpha_k\tilde{\lambda}_k^2 + t_{k+1}\omega_*\left(\tilde{c}_A^k\alpha_k\tilde{\lambda}_k t_{k+1}^{-1}\right). \tag{28}$$

By inserting (27) into (28), we obtain (26). □

Lemma 5 *For any $y^k \in \mathbb{R}^m$ and $t_k > 0$, the constant $\tilde{c}_A^k := \|\!|A|\!\|^*_{x^*(y^k;t_{k+1})}$ is bounded. More precisely, $\tilde{c}_A^k \leq \bar{c}_A := \kappa\|\!|A|\!\|^*_{x^c} < +\infty$. Furthermore, $\tilde{\lambda}_k := \|\nabla g(y^k; t_{k+1})\|_2$ is also bounded, i.e.: $\tilde{\lambda}_k \leq \bar{\lambda} := \kappa\|\!|A|\!\|^*_{x^c} + \|Ax^c - b\|_2$, where $\kappa := \sum_{i=1}^N[\nu_i + 2\sqrt{\nu_i}]$.*

Proof For any $x \in \mathrm{int}(X)$, from the definition of $\|\!|\cdot|\!\|^*_x$, we can write

$$\begin{aligned} \|\!|A|\!\|^*_x &= \sup\left\{[v^T A\nabla^2 F(x)^{-1}A^T v]^{1/2} \ : \ \|v\|_2 = 1\right\} \\ &= \sup\left\{\|u\|^*_x \ : \ u = A^T v, \ \|v\|_2 = 1\right\}. \end{aligned}$$

By using [14, Corollary 4.2.1], we can estimate $\|\!|A|\!\|^*_x$ as

$$\begin{aligned} \|\!|A|\!\|^*_x &\leq \sup\left\{\kappa\|u\|^*_{x^c} \ : \ u = A^T v, \ \|v\|_2 = 1\right\} \\ &= \kappa\sup\left\{\left[v^T A\nabla^2 F(x^c)^{-1}A^T v\right]^{1/2}, \ \|v\|_2 = 1\right\} \\ &= \kappa\|\!|A|\!\|^*_{x^c}. \end{aligned}$$

Here, the inequality in this implication follows from [14, Corollary 4.2.1]. By substituting $x = x^*(y^k; t_{k+1})$ into the above inequality, we obtain the first conclusion. In order to prove

the second bound, we note that $\nabla g(y^k; t_{k+1}) = Ax^*(y^k; t_{k+1}) - b$. Therefore, by using (7), we can estimate

$$\begin{aligned}\|\nabla g(y^k; t_{k+1})\|_2 &= \|Ax^*(y^k; t_{k+1}) - b\|_2 \le \|A(x^*(y^k; t_{k+1}) - x^c)\|_2 + \|Ax^c - b\|_2\\ &\le |||A|||^*_{x^c}\|x^*(y^k; t_{k+1}) - x^c\|_{x^c} + \|Ax^c - b\|_2\\ &\overset{(7)}{\le} \kappa|||A|||^*_{x^c} + \|Ax^c - b\|_2,\end{aligned}$$

which is the second conclusion. □

Next, we show how to choose the step size α_k and also the decrement Δt_k such that $g(y^{k+1}; t_{k+1}) < g(y^k; t_k)$ in Lemma 4. We note that $x^*(y^k; t_{k+1})$ is obtained by solving the primal subproblem (9) and the quantity $c_F^k := F(x^*(y^k; t_{k+1}))$ is nonnegative (since we have that $F(x^*(y^k; t_{k+1})) \ge F(x^c) = 0$) and computable. By Lemma 5, we see that

$$\alpha_k := \frac{t_k}{\tilde{c}_A^k(\tilde{c}_A^k + \tilde{\lambda}_k)} \ge \underline{\alpha}_k^0 := \frac{t_k}{\bar{c}_A(\bar{c}_A + \bar{\lambda})}, \tag{29}$$

which shows that $\alpha_k > 0$ as $t_k > 0$. We have the following estimate.

Lemma 6 *The step size α_k defined by (29) satisfies*

$$g(y^{k+1}; t_{k+1}) \le g(y^k; t_k) - t_{k+1}\omega\left(\frac{\tilde{\lambda}_k}{\tilde{c}_A^k}\right) + \Delta t_k F(\tilde{x}_k^*), \tag{30}$$

where $\tilde{x}_k^*$, $\tilde{c}_A^k$ and $\tilde{\lambda}_k$ are defined by (25).

Proof Let $\varphi(\alpha) := \alpha\tilde{\lambda}_k^2 - t_{k+1}\omega_*(\tilde{c}_A^k t_{k+1}^{-1}\alpha\tilde{\lambda}_k) - t_{k+1}\omega(\tilde{\lambda}_k(\tilde{c}_A^k)^{-1})$. We can simplify this function as $\varphi(\alpha) = t_{k+1}[u + \ln(1-u)]$, where $u := t_{k+1}^{-1}\tilde{\lambda}_k^2\alpha + t_{k+1}^{-1}\tilde{c}_A^k\tilde{\lambda}_k\alpha - (\tilde{c}_A^k)^{-1}\tilde{\lambda}_k$. The function $\varphi(\alpha) \le 0$ for all u and $\varphi(\alpha) = 0$ at $u = 0$ which leads to $\alpha_k := \frac{t_k}{\tilde{c}_A^k(\tilde{c}_A^k+\tilde{\lambda}_k)}$. □

Since $t_{k+1} = t_k - \Delta t_k$, if we choose $\Delta t_k := \frac{t_k\omega\left(\tilde{\lambda}_k/\tilde{c}_A^k\right)}{2\left[\omega\left(\tilde{\lambda}_k/\tilde{c}_A^k\right)+F(\tilde{x}_k^*)\right]}$, then

$$g(y^{k+1}; t_{k+1}) \le g(y^k; t_k) - \frac{t}{2}\omega\left(\tilde{\lambda}_k/\tilde{c}_A^k\right). \tag{31}$$

Therefore, the update rule for t can be written as

$$t_{k+1} := (1 - \sigma_k)t_k, \text{ where } \sigma_k := \frac{\omega\left(\tilde{\lambda}_k/\tilde{c}_A^k\right)}{2\left[\omega\left(\tilde{\lambda}_k/\tilde{c}_A^k\right) + F(\tilde{x}_k^*)\right]} \in (0, 1). \tag{32}$$

4.2 The algorithm

Now, we combine the above analysis to obtain the following path-following gradient decomposition algorithm.

Algorithm 1. (*Path-following gradient decomposition algorithm*).
Initialization:

Step 1. Choose an initial value $t_0 > 0$ and tolerances $\varepsilon_t > 0$ and $\varepsilon_g > 0$.
Step 2. Take an initial point $y^0 \in \mathbb{R}^m$ and solve (3) *in parallel* to obtain $x_0^* := x^*(y^0; t_0)$.

Step 3. Compute $c_A^0 := |||A|||_{x_0^*}^*$, $\lambda_0 := \|\nabla g(y^0; t_0)\|_2$, $\omega_0 := \omega(\lambda_0/c_A^0)$ and $c_F^0 := F(x_0^*)$.

Iteration: For $k = 0, 1, \ldots, k_{\max}$, perform the following steps:

Step 1: Update the penalty parameter as $t_{k+1} := t_k(1 - \sigma_k)$, where $\sigma_k := \frac{\omega_k}{2(\omega_k + c_F^k)}$.

Step 2: Solve (3) *in parallel* to obtain $x_k^* := x^*(y^k, t_{k+1})$. Then, form the gradient vector $\nabla g(y^k; t_{k+1}) := Ax_k^* - b$.

Step 3: Compute $\lambda_{k+1} := \|\nabla g(y^k; t_{k+1})\|_2$, $c_A^{k+1} := |||A|||_{x_k^*}^*$, $\omega_{k+1} := \omega(\lambda_{k+1}/c_A^{k+1})$ and $c_F^{k+1} := F(x_k^*)$.

Step 4: If $t_{k+1} \le \varepsilon_t$ and $\lambda_k \le \varepsilon$, then terminate.

Step 5: Compute the step size $\alpha_{k+1} := \frac{t_{k+1}}{c_A^{k+1}(c_A^{k+1} + \lambda_{k+1})}$.

Step 6: Update y^{k+1} as $y^{k+1} := y^k - \alpha_{k+1}\nabla g(y^k, t_{k+1})$.

End.

The main step of Algorithm 1 is Step 2, where we need to solve in parallel the primal subproblems. To form the gradient vector $\nabla g(\cdot, t_{k+1})$, one can compute in parallel by multiplying column-blocks A_i of A by the solution $x_i^*(y^k, t_{k+1})$. This task only requires local information to be exchanged between the current node and its neighbors.

We note that, in augmented Lagrangian approaches, we need to carefully tune the penalty parameter in an appropriate way. The update rule for the penalty parameter is usually heuristic and can be changed from problem to problem. In contrast to this, Algorithm 1 does not require any tuning strategy to update the algorithmic parameters. The formula for updating these parameters is obtained from theoretical analysis.

We note that since x_k^* is always in the interior of the feasible set, $F(x_k^*) < +\infty$, formula (32) can be used and always decreases the parameter t_k. However, in practice, this formula may lead to slow convergence. Besides, the step size α_k computed at Step 5 depends on the parameter t_k. If t_k is small, then Algorithm 1 makes short steps toward a solution of (1). In our numerical test, we use the following safeguard update:

$$t_{k+1} := \begin{cases} t_k\left(1 - \frac{\omega_k}{2(\omega_k + c_F^k)}\right) & \text{if } c_F^k \le \bar{c}_F, \\ t_k & \text{otherwise,} \end{cases} \tag{33}$$

where $\bar{c}_F$ is a sufficiently large positive constant (e.g., $\bar{c}_F := 99\omega_0$). With this modification, we observed a good performance in our numerical tests below.

4.3 Convergence analysis

Let us assume that $\underline{t} = \inf_{k\ge 0} t_k > 0$. Then, the following theorem shows the convergence of Algorithm 1.

Theorem 1 *Suppose that Assumptions A.1 and A.2 are satisfied. Suppose further that the sequence $\{(y^k, t_k, \lambda_k)\}_{k\ge 0}$ generated by Algorithm 1 satisfies $\underline{t} := \inf_{k\ge 0}\{t_k\} > 0$. Then*

$$\lim_{k\to\infty} \|\nabla g(y^k, t_{k+1})\|_2 = 0. \tag{34}$$

Consequently, there exists a limit point y^* of $\{y^k\}$ such that y^* is a solution of (16) at $t = \underline{t}$.

Proof It is sufficient to prove (34). Indeed, from (31) we have

$$\sum_{i=0}^{k} \frac{t_k}{2}\omega(\lambda_{k+1}/c_A^{k+1}) \le g(y^0; t_0) - g(y^{k+1}; t_{k+1}) \le g(y^0; t_0) - g^*.$$

Since $t_k \geq \underline{t} > 0$ and $c_A^{k+1} \leq \bar{c}_A$ due to Lemma 5, the above inequality leads to

$$\frac{\underline{t}}{2}\sum_{i=0}^{\infty}\omega(\lambda_{k+1}/\bar{c}_A) \leq g(y^0;t_0) - g^* < +\infty.$$

This inequality implies $\lim_{k\to\infty}\omega(\lambda_{k+1}/\bar{c}_A) = 0$, which leads to $\lim_{k\to\infty}\lambda_{k+1} = 0$. By definition of λ_k we have $\lim_{k\to\infty}\|\nabla g(y^k;t_{k+1})\|_2 = 0$. □

Remark 3 From the proof of Theorem 1, we can fix $c_A^k \equiv \bar{c} := \kappa\|A\|_{x^c}^*$ in Algorithm 1. This value can be computed a priori.

4.4 Local convergence rate

Let us analyze the local convergence rate of Algorithm 1. Let y^0 be an initial point of Algorithm 1 and $y^*(t)$ be the unique solution of (16). We denote by:

$$r_0(t) := \|y^0 - y^*(t)\|_2. \tag{35}$$

For simplicity of discussion, we assume that the smoothness parameter t_k is fixed at $\underline{t} > 0$ sufficiently small for all $k \geq 0$ (see Lemma 1). The convergence rate of Algorithm 1 in the case $t_k = \underline{t}$ is stated in the following lemma.

Lemma 7 (Local convergence rate) *Suppose that the initial point y^0 is chosen such that $g(y^0;\underline{t}) - g^*(\underline{t}) \leq \bar{c}_A r_0(\underline{t})$. Then,*

$$g(y^k;\underline{t}) - g^*(\underline{t}) \leq \frac{4\bar{c}_A^2 r_0(\underline{t})^2}{4\bar{c}_A r_0(\underline{t}) + \underline{t}k}. \tag{36}$$

Consequently, the local convergence rate of Algorithm 1 is at least $\mathcal{O}\left(\frac{4\bar{c}_A^2 r_0(\underline{t})^2}{\underline{t}k}\right)$.

Proof Let $r_k := \|y^k - y^*\|$, $\Delta_k := g(y^k;\underline{t}) - g^*(\underline{t}) \geq 0$, $\underline{y}^* := y^*(\underline{t})$, $\underline{\lambda}_k := \|\nabla g(y^k;\underline{t})\|_2$ and $\underline{c}_k := \|A\|_{x^*(y^k;\underline{t})}^*$. By using the fact that $\nabla g(\underline{y}^*;\underline{t}) = 0$ and (20) we have:

$$\begin{aligned} r_{k+1}^2 &= \|y^{k+1} - y^*\|^2 = \|y^k - \alpha_k\nabla g(y^k;\underline{t}) - y^*\|^2 \\ &= r_k^2 - 2\alpha_k\nabla g(y^k;\underline{t})^T(y^k - y^*) + \alpha_k^2\|\nabla g(y^k;\underline{t})\|^2 \\ &\overset{(20)}{\leq} r_k^2 - 2\alpha_k\frac{\underline{t}\underline{\lambda}_k^2}{c_A^k(c_A^k + \underline{\lambda}_k)} + \alpha_k^2\underline{\lambda}_k^2 \\ &\overset{(29)}{=} r_k^2 - \alpha_k^2\underline{\lambda}_k^2. \end{aligned}$$

This inequality implies that $r_k \leq r_0$ for all $k \geq 0$. First, by the convexity of $g(\cdot;\underline{t})$ we have:

$$\Delta_k = g(y^k;\underline{t}) - g^*(\underline{t}) \leq \|\nabla g(y^k,\underline{t})\|_2\|y^k - \underline{y}^*\|_2 = \underline{\lambda}_k\|y^0 - \underline{y}^*\|_2 \leq \underline{\lambda}_k r_0(\underline{t}).$$

This inequality implies:

$$\underline{\lambda}_k \geq r_0(\underline{t})^{-1}\Delta_k. \tag{37}$$

Since $t_k = \underline{t} > 0$ is fixed for all $k \geq 0$, it follows from (26) that:

$$g(y^{k+1};\underline{t}) \leq g(y^k;\underline{t}) - \underline{t}\omega(\underline{\lambda}_k/\underline{c}_A^k),$$

where $\underline{\lambda}_k := \|\nabla g(y^k;\underline{t})\|_2$ and $\underline{c}_A^k := |||A|||^*_{x^*(y^k;\underline{t})}$. By using the definition of Δ_k, the last inequality is equivalent to:

$$\Delta_{k+1} \le \Delta_k - \underline{t}\omega(\underline{\lambda}_k/\underline{c}_A^k). \tag{38}$$

Next, since $\omega(\tau) \ge \tau^2/4$ for all $0 \le \tau \le 1$ and $\underline{c}_A^k \le \bar{c}_A$ due to Lemma 5, it follows from (37) and (38) that:

$$\Delta_{k+1} \le \Delta_k - (\underline{t}\Delta_k^2)/(4r_0(\underline{t})^2\bar{c}_A^2), \tag{39}$$

for all $\Delta_k \le \bar{c}_A r_0(\underline{t})$.

Let $\eta := \underline{t}/(4r_0(\underline{t})^2\bar{c}_A^2)$. Since $\Delta_k \ge 0$, (39) implies:

$$\frac{1}{\Delta_{k+1}} \ge \frac{1}{\Delta_k(1-\eta\Delta_k)} = \frac{1}{\Delta_k} + \frac{\eta}{(1-\eta\Delta_k)} \ge \frac{1}{\Delta_k} + \eta.$$

By induction, this inequality leads to $\frac{1}{\Delta_k} \ge \frac{1}{\Delta_0} + \eta k$ which is equivalent to $\Delta_k \le \frac{\Delta_0}{1+\eta\Delta_0 k}$ provided that $\Delta_0 \le \bar{c}_A r_0(\underline{t})$. Since $\eta := \underline{t}/(4r_0(\underline{t})^2\bar{c}_A^2)$, this inequality is indeed (36). The last conclusion follows from (36). □

Remark 4 Let us fix $\underline{t} := \varepsilon$. It follows from (36) that the worst-case complexity of Algorithm 1 to obtain an ε-solution y^k in the sense $g(y^k;\varepsilon) - g^*(\varepsilon) \le \varepsilon$ is $\mathcal{O}\left(\frac{\bar{c}_A^2 r_0^2}{\varepsilon^2}\right)$. We note that $\bar{c}_A = \kappa|||A|||^*_{x^c} = \sum_{i=1}^N(\nu_i + 2\sqrt{\nu_i})|||A_i|||^*_{x_i^c}$. However, in most cases, the parameter ν_i depends linearly on the dimension of the problem. Therefore, we can conclude that the worst-case complexity of Algorithm 1 is $\mathcal{O}\left(\frac{(n\|A\|^*_{x^c}r_0)^2}{\varepsilon^2}\right)$.

5 Fast gradient decomposition algorithm

Let us fix $t = \underline{t} > 0$. The function $\underline{g}(\cdot) := g(\cdot;\underline{t})$ is convex and differentiable but its gradient is not Lipschitz continuous, we can not apply Nesterov's fast gradient algorithm [14] to solve (16). In this section, we modify Nesterov's fast gradient method in order to obtain an accelerating gradient method for solving (16).

One iteration of the modified fast gradient method is described as follows. Let y^k and v^k be given points in $\in \mathbb{R}^m$, we compute new points y^{k+1} and v^{k+1} as follows:

$$\begin{cases} y^{k+1} := v^k - \alpha_k\nabla\underline{g}(v^k), \\ v^{k+1} = a_k y^{k+1} + b_k y^k + c_k v^k, \end{cases} \tag{40}$$

where $\alpha_k > 0$ is the step size, a_k, b_k and c_k are three parameters which will be chosen appropriately. As we can see from (40), at each iteration k, we only require to evaluate one gradient $\nabla\underline{g}(v^k)$ of the function $\underline{g}$. First, we prove the following estimate.

Lemma 8 *Let $\theta_k \in (0,1)$ be a given parameter, $\alpha_k := \frac{\underline{t}}{\hat{c}_A(\hat{c}_A+\lambda_k)}$ and $\rho_k := \frac{\underline{t}}{2\theta_k(\hat{c}_A^k)^2}$ for some parameter $\hat{c}_A^k \ge c_A^k$, where $\lambda_k := \|\nabla\underline{g}(v^k)\|_2$ and $c_A^k := |||A|||^*_{x^*(v^k;\underline{t})}$. We define two vectors*

$$r^k := \theta_k^{-1}[v^k - (1-\theta_k)y^k] \text{ and } r^{k+1} := r^k - \rho_k\nabla\underline{g}(v^k). \tag{41}$$

Then, the new point y^{k+1} *generated by* (40) *satisfies*

$$\frac{1}{\theta_k^2}\left[\underline{g}(y^{k+1}) - \underline{g}^*\right] + \frac{(\hat{c}_A^k)^2}{\underline{t}}\|r^{k+1} - \underline{y}^*\|_2^2 \le \frac{(1-\theta_k)}{\theta_k^2}\left[\underline{g}(y^k) - \underline{g}^*\right] + \frac{(\hat{c}_A^k)^2}{\underline{t}}\|r^k - \underline{y}^*\|_2^2, \tag{42}$$

provided that $\lambda_k \le \hat{c}_A^k$, *where* $\underline{y}^* := y^*(\underline{t})$ *and* $\underline{g}^* := g(\underline{y}^*; \underline{t})$.

Proof Since $y^{k+1} = v^k - \alpha_k \nabla \underline{g}(v^k)$ and $\alpha_k = \frac{\underline{t}}{\hat{c}_A^k(\hat{c}_A^k + \lambda_k)}$, it follows from (21) that

$$\underline{g}(y^{k+1}) \le \underline{g}(v^k) - \underline{t}\,\omega\left(\frac{\|\nabla \underline{g}(v^k)\|_2}{\hat{c}_A^k}\right). \tag{43}$$

Now, since $\omega(\tau) \ge \tau^2/4$ for all $0 \le \tau \le 1$, the inequality (43) implies

$$\underline{g}(y^{k+1}) \le \underline{g}(v^k) - \frac{\underline{t}}{4(\hat{c}_A^k)^2}\|\nabla \underline{g}(v^k)\|_2^2, \tag{44}$$

provided that $\|\nabla \underline{g}(v^k)\|_2 \le \hat{c}_A^k$. For any $u^k := (1-\theta_k)y^k + \theta_k \underline{y}^*$ and $\theta_k \in (0,1)$ we have

$$\begin{aligned}\underline{g}(v^k) &\le \underline{g}(u^k) + \nabla \underline{g}(v^k)^T(v^k - u^k) \le (1-\theta_k)\underline{g}(y^k) + \theta_k \underline{g}(\underline{y}^*) \\ &\quad + \nabla \underline{g}(v^k)^T(v^k - (1-\theta_k)y^k - \theta_k \underline{y}^*).\end{aligned} \tag{45}$$

By substituting (45) and the relation $v^k - (1-\theta_k)y^k = \theta_k r^k$ into (44) we obtain:

$$\begin{aligned}\underline{g}(y^{k+1}) &\le (1-\theta_k)\underline{g}(y^k) + \theta_k \underline{g}^* + \theta_k \nabla \underline{g}(v^k)^T(r^k - \underline{y}^*) - \frac{\underline{t}}{4(\hat{c}_A^k)^2}\|\nabla \underline{g}(v^k)\|_2^2 \\ &= (1-\theta_k)\underline{g}(y^k) + \theta_k \underline{g}^* + \frac{\theta_k^2(\hat{c}_A^k)^2}{\underline{t}}\left[\|r^k - \underline{y}^*\|_2^2 - \|r^k - \frac{\underline{t}}{2\theta_k(\hat{c}_A^k)^2}\nabla \underline{g}(v^k) - \underline{y}^*\|_2^2\right] \\ &= (1-\theta_k)\underline{g}(y^k) + \theta_k \underline{g}^* + \frac{\theta_k^2(\hat{c}_A^k)^2}{\underline{t}}\left[\|r^k - \underline{y}^*\|_2^2 - \|r^{k+1} - \underline{y}^*\|_2^2\right].\end{aligned} \tag{46}$$

Since $1/\theta_k^2 = (1-\theta_k)/\theta_k^2 + 1/\theta_k$, by rearranging (46) we obtain (42). □

Next, we consider the update rule of θ_k. We can see from (42) that if θ_{k+1} is updated such that $(1-\theta_{k+1})/\theta_{k+1}^2 = 1/\theta_k^2$, then $\underline{g}(y^{k+1}) < \underline{g}(y^k)$. The last condition leads to:

$$\theta_{k+1} = 0.5\theta_k(\sqrt{\theta_k^2 + 4} - \theta_k). \tag{47}$$

The following lemma was proved in [20].

Lemma 9 *The sequence* $\{\theta_k\}$ *generated by* (47) *starting from* $\theta_0 = 1$ *satisfies*

$$\frac{1}{2k+1} \le \theta_k \le \frac{2}{k+2}, \quad \forall k \ge 0.$$

By Lemma 8, we have $r^{k+1} = r^k - \rho_k \nabla \underline{g}(v^k)$ and $r^{k+1} = \frac{1}{\theta_{k+1}}(v^{k+1} - (1-\theta_{k+1})y^{k+1})$. From these relations, we deduce

$$v^{k+1} = (1-\theta_{k+1})y^{k+1} + \theta_{k+1}(r^k - \rho_k \nabla \underline{g}(v^k)). \tag{48}$$

Note that if we combine (48) and (40) then

$$v^{k+1} = (1 - \theta_{k+1} - \frac{\rho_k \theta_{k+1}}{\alpha_k}) y^{k+1} - \frac{(1-\theta_k)\theta_{k+1}}{\theta_k} y^k + \left(\frac{1}{\theta_k} + \frac{\rho_k}{\alpha_k}\right) \theta_{k+1} v^k.$$

This is in fact the second line of (40), where $a_k := 1 - \theta_{k+1} - \rho_k \theta_{k+1} \alpha_k^{-1}$, $b_k := -(1 - \theta_k)\theta_{k+1}\theta_k^{-1}$ and $c_k := (\theta_k^{-1} + \rho_k \alpha_k^{-1})\theta_{k+1}$.

Before presenting the algorithm, we show how to choose $\hat{c}_A^k$ to ensure the condition $\lambda_k \le \hat{c}_A^k$. Indeed, from Lemma 5 we see that if we choose $\hat{c}_A^k := \hat{c}_A \equiv \bar{c}_A + \|Ax^c - b\|_2$, then $\lambda_k \le \hat{c}_A^k$. Now, by combining all the above analysis, we can describe the modified fast gradient algorithm in detail as follows.

Algorithm 2. (*Modified fast gradient decomposition algorithm*).

Initialization: Perform the following steps:

Step 1. Given a tolerance $\varepsilon > 0$. Fix the parameter t at a certain value $\underline{t} > 0$ and compute $\hat{c}_A := \kappa \|A\|_{x^c}^* + \|Ax^c - b\|_2$.
Step 2. Take an initial point $y^0 \in \mathbb{R}^m$.
Step 3. Set $\theta_0 := 1$ and $v^0 := y^0$.

Iteration: For $k = 0, 1, \ldots, k_{\max}$, perform the following steps:

Step 1: If $\lambda_k \le \varepsilon$, then terminate.
Step 2: Compute $r^k := \theta_k^{-1}[v^k - (1 - \theta_k) y^k]$.
Step 3: Update y^{k+1} as $y^{k+1} := v^k - \alpha_k \nabla \underline{g}(v^k)$, where $\alpha_k = \frac{\underline{t}}{\hat{c}_A(\hat{c}_A + \lambda_k)}$.
Step 4: Update $\theta_{k+1} := \frac{1}{2}\theta_k[(\theta_k^2 + 4)^{1/2} - \theta_k]$.
Step 5: Update $v^{k+1} := (1 - \theta_{k+1}) y^{k+1} + \theta_{k+1}(r^k - \rho_k \nabla \underline{g}(v^k))$, where $\rho_k := \frac{\underline{t}}{2\hat{c}_A^2 \theta_k}$.
Step 6: Solve (3) *in parallel* to obtain $x_{k+1}^* := x^*(v^{k+1}, \underline{t})$. Then, form a gradient vector $\nabla \underline{g}(v^{k+1}) := Ax_{k+1}^* - b$ and compute $\lambda_{k+1} := \|\nabla \underline{g}(v^{k+1})\|_2$.

End.

The core step of Algorithm 2 is Step 6, where we need to solve N primal subproblems of the form (3) in parallel. The following theorem shows the convergence of Algorithm 2.

Theorem 2 *Let $y^0 \in \mathbb{R}^m$ be an initial point of Algorithm 2. Then the sequence $\{(y^k, v^k)\}_{k \ge 0}$ generated by Algorithm 2 satisfies*

$$\underline{g}(y^k) - g^*(\underline{t}) \le \frac{4\hat{c}_A^2}{\underline{t}(k+1)^2} \|y^0 - y^*(\underline{t})\|^2. \tag{49}$$

Proof By the choice of $\hat{c}_A$ the condition $\lambda_k \le \hat{c}_A$ is always satisfied. From (42) and the update rule of θ_k, we have

$$\frac{1}{\theta_k^2}\left[\underline{g}(y^{k+1}) - \underline{g}^*\right] + \frac{\hat{c}_A^2}{\underline{t}} \|r^{k+1} - \underline{y}^*\|_2^2 \le \frac{1}{\theta_{k-1}^2}\left[\underline{g}(y^k) - \underline{g}^*\right] + \frac{\hat{c}_A^2}{\underline{t}} \|r^k - \underline{y}^*\|_2^2$$

By induction, we obtain from this inequality that

$$\frac{1}{\theta_{k-1}^2}\left[\underline{g}(y^k) - \underline{g}^*\right] \le \frac{1}{\theta_0^2}\left[\underline{g}(y^1) - \underline{g}^*\right] + \frac{\hat{c}_A^2}{\underline{t}} \|r^1 - \underline{y}^*\|_2^2 \le \frac{1 - \theta_0}{\theta_0^2}\left[\underline{g}(y^0) - \underline{g}^*\right] + \frac{\hat{c}_A^2}{\underline{t}} \|r^0 - \underline{y}^*\|_2^2,$$

for $k \geq 1$. Since $\theta_0 = 1$ and $y^0 = v^0$, we have $r^0 = y^0$ and the last inequality implies $\underline{g}(y^k) - \underline{g}^* \leq \frac{\hat{c}_A^2 \theta_{k-1}^2}{\underline{t}} \|y^0 - \bar{y}\|_2^2$. Since $\theta_{k-1} \leq \frac{2}{k+1}$ due to Lemma 9, we obtain (49). □

Remark 5 Let $\varepsilon > 0$ be a given accuracy. If we fix the penalty parameter $\underline{t} := \varepsilon$, then the worst-case complexity of Algorithm 2 is $\mathcal{O}(\frac{2\hat{c}_A \underline{r}_0}{\varepsilon})$, where $\underline{r}_0 := r_0(\underline{t})$ is defined as above.

Similarly to Algorithm 1, in Algorithm 2, we do not require any tuning strategy for the algorithmic parameters. The parameters α_k, θ_k and ρ_k are updated automatically by using the formulas obtained from convergence analysis.

Theoretically, we can use the worst-case upper bound constant $\hat{c}_A$ in any implementation of Algorithm 2. However, this constant may be large. Using this value may lead to a slow convergence. One way to evaluate a better practical upper bound is as follows. Let us take a constant $\hat{c}_A > 0$ and define

$$\mathscr{R}(\hat{c}_A; \underline{t}) := \left\{ y \in \mathbb{R}^m \mid \|\nabla g(y; \underline{t})\|_2 \leq \hat{c}_A \right\}. \tag{50}$$

It is obvious that $y^*(\underline{t}) \in \mathscr{R}(\hat{c}_A; \underline{t})$. This set is a neighbourhood of the solution $y^*(\underline{t})$ of problem (16). Moreover, by observing that the sequence $\{v^k\}$ converges to the solution $y^*(\underline{t})$, we can assume that for k sufficiently large, $\{v^l\}_{l \geq k} \subseteq \mathscr{R}(\hat{c}_A; \underline{t})$. In this case, we can apply the following switching strategy.

Remark 6 (*Switching strategy*) We can combine Algorithms 1 and 2 to obtain a switching variant:

- First, we apply Algorithm 1 to find a point $\hat{y}^0 \in \mathbb{R}^m$ and $\underline{t} > 0$ such that $\|\nabla g(\hat{y}^0; \underline{t})\|_2 \leq \hat{c}_A$.
- Then, we switch to use Algorithm 2.

Finally, we note that by a change of variable $x := P\tilde{x}$, the linear constraint $Ax = b$ can be written as $\tilde{A}\tilde{x} = b$, where $\tilde{A} := AP$. By an appropriate choice of P, we can reduce the norm $\|\tilde{A}\|_x$ significantly.

6 Numerical tests

In this section, we test the switching variant of Algorithms 1 and 2 proposed in Remark 6 which we name by PFGDA for solving the following convex programming problem:

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & \gamma \|x\|_1 + f(x) \\ \text{s.t.} & Ax = b,\ l \leq x \leq u, \end{array} \tag{51}$$

where $\gamma > 0$ is a given regularization parameter, $f(x) := \sum_{i=1}^n f_i(x_i)$, and $f_i : \mathbb{R} \to \mathbb{R}$ is a convex function, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $l, u \in \mathbb{R}^n$ such that $l \leq 0 < u$.

We note that the feasible set $X := [l, u]$ can be decomposed into n intervals $X_i := [l_i, u_i]$ and each interval is endowed with a 2-self concordant barrier $F_i(x_i) := -\ln(x_i - l_i) - \ln(u_i - x_i) + 2\ln((u_i - l_i)/2)$ for $i = 1, \ldots, n$. Moreover, if we define $\phi(x) := -\sum_{i=1}^n [f_i(x_i) + \gamma |x_i|]$ then ϕ is concave and separable. Problem (51) can be reformulated equivalently to (SepCP).

The smoothed dual function components $g_i(y; t)$ of (51) can be written as

$$g_i(y; t) = \max_{l_i < x_i < u_i} \left\{ -f_i(x_i) - \gamma |x_i| + (A_i^T y)x_i - t F_i(x_i) \right\} - b^T y/n,$$

for $i = 1, \ldots, n$. This one-variable minimization problem is nonsmooth but it can be solved easily. In particular, if f_i is affine or quadratic then this problem can be solved in a *closed form*. In case f_i is smooth, we can reformulate (51) into a smooth convex program by adding n slack variables and $2n$ additional inequality constraints to handle the $\|x\|_1$ part.

We have implemented PFGDA in C++ running on a PC Intel ®Xeon X5690 at 3.47 GHz per core with 94 Gb RAM. The algorithm was parallelized by using OpenMP. We terminated PFGDA if

$$\texttt{optim} := \|\nabla g(y^k; t_k)\|_2 / \max\left\{1, \|\nabla g(y^0; t_0)\|_2\right\} \leq 10^{-3} \text{ and } t_k \leq 10^{-2}.$$

We have also implemented other three algorithms from the literature for comparisons, namely *a dual decomposition algorithm with two primal steps* developed in [20, Algorithm 1], a parallel variant of *the alternating direction method of multipliers* from [10] and *decomposition algorithm with two dual steps* from [19, Algorithm 1] which we named 2pDecompAlg, pADMM and 2dDecompAlg, respectively, for solving problem (51). We terminated pADMM, 2pDecompAlg and 2dDecompAlg by using the same conditions as in [10,19,20] with the tolerances $\varepsilon_{\text{feas}} = \varepsilon_{\text{fun}} = \varepsilon_{\text{obj}} = 10^{-3}$ and $j_{\max} = 3$. We also terminated all three algorithms if the maximum number of iterations maxiter := 20,000 was reached. In the last case we state that the algorithm has failed.

a. Basis pursuit problem If the function $f(x) \equiv 0$ for all x, then problem (51) becomes a bound constrained basis pursuit problem to recover the sparse coefficient vector x of given signals based on a transform operator A and a vector of observations b. We assume that $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $x \in \mathbb{R}^n$, where $m < n$ and x has k nonzero elements ($k \ll n$).

In this case, we only illustrate PFGDA by applying it to solve some small size test problems. In order to generate a test problem, we generate an orthogonal random matrix A and a random vector x_0 which has k nonzero elements (k-sparse). Then we define vector b as $b := Ax_0$. The parameter γ is set to 1.0.

We test PFGDA on the four problems such that $[m, n, k]$ are [50, 128, 14], [100, 256, 20], [200, 512, 30] and [500, 1,024, 50]. The results reported by PFGDA are plotted in Fig. 1.

As we can see from these plots, the vector of recovered coefficients x matches very well the vector of original coefficients x_0 in these four problems. Moreover, PFGDA requires 376, 334, 297 and 332 iterations, respectively in the four problems.

b. Nonlinear separable convex problems In order to test the performance of PFGDA, we generate in this case a large test-set of problems and compare the performance of PFGDA with 2pDecompAlg, 2dDecompAlg and pADMM (a parallel variant of the alternating direction method of multipliers [10]). Further comparisons with other methods such as the proximal based decomposition method [3] and the proximal-center based decomposition method [11] can be found in [19,20].

The test problems were generated as follows. We chose the objective function $f_i(x_i) := e^{-\gamma_i x_i} - 1$, where $\gamma_i > 0$ is a given parameter for $i = 1, \ldots, n$. Matrix A was generated randomly in $[-1, 1]$ and then was normalized by $A/\|A\|_\infty$. We generated a sparse vector x_0 randomly in $[-2, 2]$ with the density $\mu \leq 1\,\%$ and defined a vector $b := A\bar{x}$. Vector $\gamma := (\gamma_1, \cdots, \gamma_n)^T$ was sparse and generated randomly in $[0, 0.5]$. The lower bound l_i and the upper bounds u_i were set to -3 and 3, respectively for all $i = 1, \ldots, n$.

We benchmarked four algorithms with performance profiles [4]. 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 $T_{p,s} :=$ *computational time required to solve problem p by solver s*. We compare the performance of algorithm s on problem p with the best performance of any algorithm on this problem;

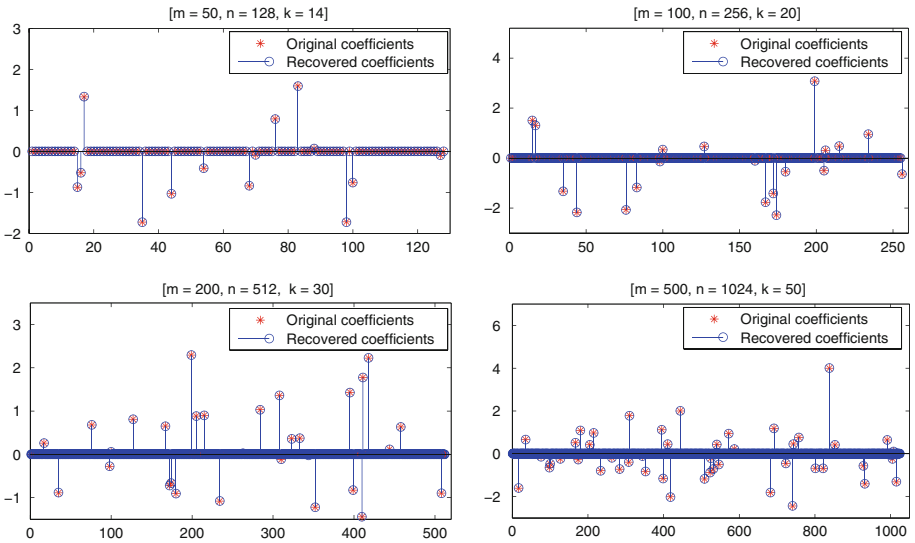

Fig. 1 Illustration of PFGDA via the basis pursuit problem

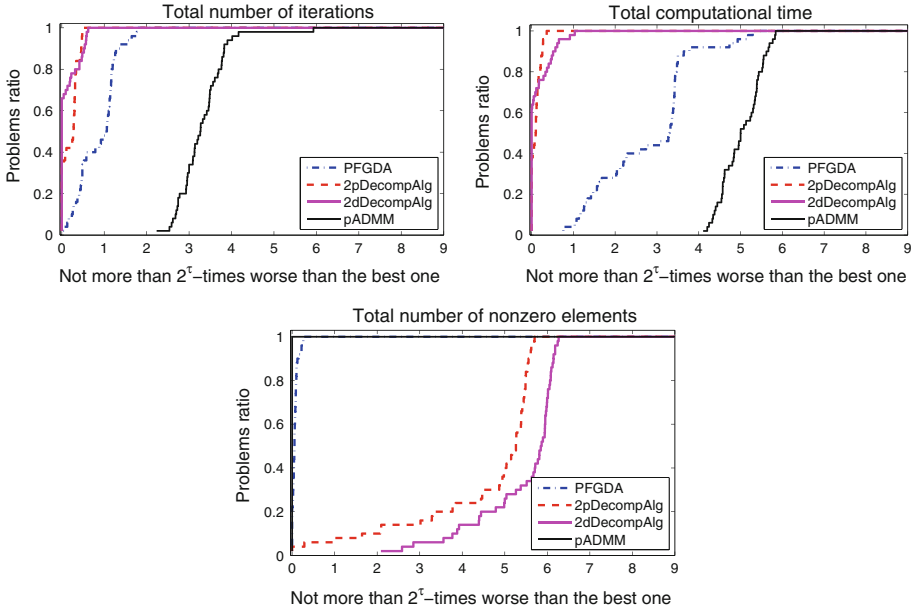

Fig. 2 Performance profiles in $\log_2$ scale of three algorithms

that is we compute the performance ratio $r_{p,s} := \frac{T_{p,s}}{\min\{T_{p,\hat{s}} \mid \hat{s} \in \mathscr{S}\}}$. Now, let $\tilde{\rho}_s(\tilde{\tau}) := \frac{1}{n_p}\text{size}\left\{p \in \mathscr{P} \mid r_{p,s} \leq \tilde{\tau}\right\}$ 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. $\rho_s(\tau) := \frac{1}{n_p}\text{size}\left\{p \in \mathscr{P} \mid \log_2(r_{p,s}) \leq \tau := \log_2 \tilde{\tau}\right\}$.

We tested the four algorithms on a collection of 50 random problems with m ranging from 200 to 1,500 and n ranging from 1,000 to 15,000. The profiles are plotted in Fig. 2. Based on this test, we can make the following observations. `2dDecompAlg` has the best performance in terms of iterations and computational time. It solves 66 % problems with the best performance in terms of iterations and 63 % problems with the best performance in time. These quantities are 34 and 38 %, respectively in `2pDecompAlg`. However, the final solution given by two algorithms, `2pDecompAlg` and `2dDecompAlg`, is rather dense. The number of nonzero elements is much larger (up to 77 times) than in the true solution. By analyzing their solutions, we observed that these solutions have many small entries. `PFGDA` and `pADMM` provided good solutions in terms of sparsity. These solutions approximate well the true solution. Nevertheless, `pADMM` is much slower than `PFGDA` in terms of computational time as well as the number of iterations. The objective values obtained by `PFGDA` is better than in `pADMM` in the majority of problems and the computational times for our algorithm are also superior to `pADMM`.

For more insights into the behavior of our algorithm, we report the performance information of the four algorithms (`PFGDA`, `2pDecompAlg`, `2dDecompAlg` and `pADMM`) for 10 problems with different sizes in Table 1. Here, `iter` is the number of iterations, `time[s]` is the computational time in second, `#nnz` is the number of nonzero elements, `#nnz0` is the number of nonzero elements of the true solution x^*, `match` is the number of nonzero

Table 1 Performance information of four algorithms (`PFGDA`, `2pDecompAlg`, `2dDecompAlg` and `pADMM`) on 10 synthetic data problems

Algorithm	m	n	iter	time[s]	#nnz	#nnz0	match	fgap	fval
PFGDA	200	1,000	979	1.69	10	10	10	0.782×10^{-3}	**12.168**
2pDecompAlg	200	1,000	**655**	**0.41**	144	10	10	0.992×10^{-3}	14.720
2dDecompAlg	200	1,000	984	0.85	210	10	10	0.357×10^{-3}	17.220
pADMM	200	1,000	6,334	16.47	10	10	10	0.893×10^{-3}	12.368
PFGDA	500	1,000	991	2.91	10	9	9	0.812×10^{-3}	**8.711**
2pDecompAlg	500	1,000	883	1.57	11	9	9	0.994×10^{-3}	9.273
2dDecompAlg	500	1,000	**829**	**1.22**	65	9	9	0.882×10^{-3}	11.497
pADMM	500	1,000	5,542	28.97	9	9	9	0.933×10^{-3}	8.713
PFGDA	700	2,000	1,330	9.12	12	12	12	0.934×10^{-3}	**16.112**
2pDecompAlg	700	2,000	**926**	**4.17**	261	12	12	0.993×10^{-3}	22.341
2dDecompAlg	700	2,000	1,347	5.53	461	12	12	0.722×10^{-3}	26.953
pADMM	700	2,000	9,890	174.41	12	12	12	0.987×10^{-3}	16.248
PFGDA	1,000	3,000	1,640	53.86	20	19	19	0.726×10^{-3}	**26.058**
2pDecompAlg	1,000	3,000	**1,186**	**13.09**	600	19	19	0.746×10^{-3}	39.434
2dDecompAlg	1,000	3,000	1,644	18.69	1,001	19	19	0.630×10^{-3}	51.157
pADMM	1,000	3,000	13,164	514.60	19	19	19	0.976×10^{-3}	26.070
PFGDA	1,500	8,000	2,405	493.87	57	56	56	0.967×10^{-3}	**73.699**
2pDecompAlg	1,500	8,000	1,395	53.55	2,520	56	56	0.989×10^{-3}	143.381
2dDecompAlg	1,500	8,000	**1,150**	**49.31**	3,714	56	55	0.993×10^{-3}	207.594
pADMM	1,500	8,000	13,120	2,072.32	56	56	56	0.976×10^{-3}	74.453

Table 1 continued

Algorithm	m	n	iter	time[s]	#nnz	#nnz0	match	fgap	fval
PFGDA	1,900	10,000	2,869	909.85	81	76	76	0.899×10^{-3}	**91.158**
2pDecompAlg	1,900	10,000	1,607	95.15	3,188	76	76	0.996×10^{-3}	179.404
2dDecompAlg	1,900	10,000	**1,292**	**86.47**	4,798	76	76	0.995×10^{-3}	253.960
pADMM	1,900	10,000	17,620	3,251.22	76	76	76	0.943×10^{-3}	91.487
PFGDA	2,000	10,400	3,080	1,061.38	87	82	82	0.896×10^{-3}	99.755
2pDecompAlg	2,000	10,400	1,605	105.82	3,492	82	82	0.996×10^{-3}	196.732
2dDecompAlg	2,000	10,400	**1,315**	**100.34**	5,082	82	81	0.996×10^{-3}	275.439
pADMM	2,000	10,400	7,630	2,184.13	82	82	82	0.985×10^{-3}	**99.139**
PFGDA	2,500	14,500	3,828	2,514.84	109	106	106	0.900×10^{-3}	**133.720**
2pDecompAlg	2,500	14,500	2,027	215.64	4,706	106	106	0.994×10^{-3}	270.498
2dDecompAlg	2,500	14,500	**1,474**	**183.78**	7,250	106	106	0.994×10^{-3}	381.443
pADMM	2,500	14,500	11,420	4,511.21	106	106	106	0.954×10^{-3}	133.818
PFGDA	1,400	15,000	3,073	2,160.51	101	99	99	0.962×10^{-3}	118.879
2pDecompAlg	1,400	15,000	1,369	85.74	3,571	99	97	0.978×10^{-3}	213.078
2dDecompAlg	1,400	15,000	**981**	**70.90**	5,697	99	96	0.972×10^{-3}	268.632
pADMM	1,400	15,000	11,021	2,484.57	99	99	99	0.952×10^{-3}	**118.597**
PFGDA	1,500	15,000	3,007	2,118.78	92	92	92	0.966×10^{-3}	**110.145**
2pDecompAlg	1,500	15,000	1,426	95.08	3,619	92	88	0.985×10^{-3}	207.733
2dDecompAlg	1,500	15,000	**1,026**	**79.68**	5,698	92	88	0.985×10^{-3}	265.100
pADMM	1,500	15,000	18,420	4,569.05	92	92	92	0.974×10^{-3}	111.701

The definition for significance of bold means to highlight which one is the best

elements of the approximate solution x^k which match the true solution x^*, `fgap` is the feasibility gap and `fval` is the objective value.

As we can observe from this table the `PFGDA` and `pADMM` provided better solutions in terms of sparsity as well as the final objective value than `2pDecompAlg` and `2dDecompAlg`. In fact, `2pDecompAlg` and `2dDecompAlg` provided a poor quality solution (with many small elements) in this example. The nonzero elements in the solutions obtained by `PFGDA` and `pADMM` match very well the nonzero elements in the true solutions. Further, the corresponding objective values in both methods is close to each other. However, the number of iterations as well as the computational times in `PFGDA` are much lower than in `pADMM` (in the range of 2 to 10 times faster).

7 Concluding remarks

In this paper we have proposed two new dual gradient-based decomposition algorithms for solving large-scale separable convex optimization problems. We have analyzed the convergence of these two schemes and derived the rate of convergence. The first property of these methods is that they can handle general convex objective functions. Therefore, they can be applied to a wide range of applications compared to second order methods. Secondly, the new algorithms can be implemented in parallel and all the algorithmic parameters are updated

automatically without using any tuning strategy. Thirdly, the convergence rate of Algorithm 2 is $\mathcal{O}(1/k)$ which is optimal in the dual decomposition framework. Finally, the complexity estimates of the algorithms do not depend on the diameter of the feasible set as in proximity function smoothing methods, they only depend on the parameter of the barrier functions.

Acknowledgments We thank the editor and two anonymous reviewers for their comments and suggestions to improve the presentation of the paper. This research was supported by Research Council KUL: PFV/10/002 Optimization in Engineering Center OPTEC, GOA/10/09 MaNet and GOA/10/11 Global real-time optimal control of autonomous robots and mechatronic systems. Flemish Government: IOF/KP/SCORES4CHEM, FWO: PhD/postdoc grants and projects: G.0320.08 (convex MPC), G.0377.09 (Mechatronics MPC); IWT: PhD Grants, projects: SBO LeCoPro; Belgian Federal Science Policy Office: IUAP P7 (DYSCO, Dynamical systems, control and optimization, 2012–2017); EU: FP7-EMBOCON (ICT-248940), FP7-SADCO (MC ITN-264735), ERC ST HIGHWIND (259 166), Eurostars SMART, ACCM; the European Union, Seventh Framework Programme (FP7/2007–2013), EMBOCON, under grant agreement no 248940; CNCS-UEFISCDI (project TE, no. 19/11.08.2010); ANCS (project PN II, no. 80EU/2010); Sectoral Operational Programme Human Resources Development 2007–2013 of the Romanian Ministry of Labor, Family and Social Protection through the Financial Agreements POSDRU/89/1.5/S/62557.

References

1. Bertsekas, D., Tsitsiklis, J.N.: Parallel and Distributed Computation: Numerical Methods. Prentice Hall, Englewood Cliffs (1989)
2. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the 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. Dolan, E., Moré, J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
5. Duchi, J., Agarwal, A., Wainwright, M.: Dual averaging for distributed optimization: convergence analysis and network scaling. IEEE Trans. Autom. Control **57**(3), 592–606 (2012)
6. Fraikin, C., Nesterov, Y., Dooren, P.V.: Correlation between two projected matrices under isometry constraints. CORE Discussion Paper 2005/80, UCL (2005)
7. Hamdi, A.: Two-level primal-dual proximal decomposition technique to solve large-scale optimization problems. Appl. Math. Comput. **160**, 921–938 (2005)
8. Hamdi, A., Mishra, S.: Decomposition methods based on augmented Lagrangians: a survey. In: Mishra S.K. (ed.) Topics in Nonconvex Optimization: Theory and Application, pp. 175–203. Springer-Verlag (2011)
9. Kojima, M., Megiddo, N., Mizuno, S.: Horizontal and vertical decomposition in interior point methods for linear programs. Technical report, Information Sciences, Tokyo Institute of Technology, Tokyo (1993)
10. Lenoir, A., Mahey, P.: Accelerating convergence of a separable augmented Lagrangian algorithm. Technical report, LIMOS/RR-07-14, 1–34 (2007).
11. Necoara, I., Suykens, J.: Applications of a smoothing technique to decomposition in convex optimization. IEEE Trans. Autom. Control **53**(11), 2674–2679 (2008)
12. Necoara, I., Suykens, J.: Interior-point lagrangian decomposition method for separable convex optimization. J. Optim. Theory Appl. **143**(3), 567–588 (2009)
13. Nedíc, A., Ozdaglar, A.: Distributed subgradient methods for multi-agent optimization. IEEE Trans. Autom. Control **54**, 48–61 (2009)
14. Nesterov, Y.: Introductory Lectures on Convex Optimization: A Basic Course, Applied Optimization, vol. 87. Kluwer Academic Publishers, Dordrecht (2004)
15. Nesterov, Y., Nemirovski, A.: Interior-Point Polynomial Algorithms in Convex Programming. Society for Industrial Mathematics, Philadelphia (1994)
16. Nesterov, Y., Protasov, V.: Optimizing the spectral radius. CORE Discussion Paper pp. 1–16 (2011)
17. Palomar, D., Chiang, M.: A tutorial on decomposition methods for network utility maximization. IEEE J. Sel. Areas Commun. **24**(8), 1439–1451 (2006)
18. Ruszczyński, A.: On convergence of an augmented lagrangian decomposition method for sparse convex optimization. Math. Oper. Res. **20**, 634–656 (1995)

19. Tran-Dinh, Q., Necoara, I., Savorgnan, C., Diehl, M.: An inexact perturbed path-following method for Lagrangian decomposition in large-scale separable convex optimization. SIAM J. Optim. **23**(1), 95–125 (2013)
20. Tran-Dinh, Q., Savorgnan, C., Diehl, M.: Combining lagrangian decomposition and excessive gap smoothing technique for solving large-scale separable convex optimization problems. Comput. Optim. Appl. **55**(1), 75–111 (2012)
21. Xiao, L., Johansson, M., Boyd, S.: Simultaneous routing and resource allocation via dual decomposition. IEEE Trans. Commun. **52**(7), 1136–1144 (2004)
22. Zhao, G.: A Lagrangian dual method with self-concordant barriers for multistage stochastic convex programming. Math. Progam. **102**, 1–24 (2005)