Comput Optim Appl (2014) 57:45–69
DOI 10.1007/s10589-013-9590-3

Implementation of a block-decomposition algorithm for solving large-scale conic semidefinite programming problems

Renato D.C. Monteiro · Camilo Ortiz · Benar F. Svaiter

Received: 8 October 2012 / Published online: 10 August 2013

Abstract In this paper, we consider block-decomposition first-order methods for solving large-scale conic semidefinite programming problems given in standard form. Several ingredients are introduced to speed-up the method in its pure form such as: an aggressive choice of stepsize for performing the extragradient step; use of scaled inner products; dynamic update of the scaled inner product for properly balancing the primal and dual relative residuals; and proper choices of the initial primal and dual iterates, as well as the initial parameter for the scaled inner product. Finally, we present computational results showing that our method outperforms the two most competitive codes for large-scale conic semidefinite programs, namely: the boundary-point method introduced by Povh et al. and the Newton-CG augmented Lagrangian method by Zhao et al.

Keywords Complexity · Proximal · Extragradient · Block-decomposition · Convex optimization · Conic optimization · Semidefinite programing

The work of R.D.C. Monteiro was partially supported by NSF Grants CCF-0808863, CMMI-0900094 and CMMI-1300221, and ONR Grant ONR N00014-11-1-0062.
The work of B.F. Svaiter was partially supported by CNPq grants no. 474944/2010-7, 303583/2008-8 and FAPERJ grant E-26/110.821/2008.

Electronic supplementary material The online version of this article (doi:10.1007/s10589-013-9590-3) contains supplementary material, which is available to authorized users.

R.D.C. Monteiro · C. Ortiz (✉)
School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0205, USA
e-mail: camiort@gatech.edu

R.D.C. Monteiro
e-mail: monteiro@isye.gatech.edu

B.F. Svaiter
IMPA, Estrada Dona Castorina 110, 22460-320 Rio de Janeiro, Brazil
e-mail: benar@impa.br

1 Introduction

Let $\mathcal{X}$ and $\mathcal{Y}$ be finite dimensional inner product spaces, with inner products and associated norms denoted by $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$, respectively. The conic programming problem is

$$\min\{\langle c, x\rangle \ : \ \mathcal{A}x = b, \ x \in K\}, \tag{1}$$

where $\mathcal{A} : \mathcal{X} \to \mathcal{Y}$ is a linear map, $b \in \mathcal{Y}$, $c \in \mathcal{X}$ and $K \subset \mathcal{X}$ is a closed convex cone. The corresponding dual problem is

$$\max\{\langle b, y\rangle \ : \ c - \mathcal{A}^* y \in K^*\}, \tag{2}$$

where $\mathcal{A}^*$ denotes the adjoint of $\mathcal{A}$ and K^* is the dual cone of K defined as

$$K^* := \{v \in \mathcal{X} : \langle x, v\rangle \geq 0, \ \forall x \in K\}. \tag{3}$$

Let $\mathbb{R}^n$ denote the n dimensional Euclidean space and $\mathbb{R}^n_+$ denote the cone of nonnegative vectors in $\mathbb{R}^n$. Also, let $\mathcal{S}^n$ denote the linear space of all $n \times n$ symmetric matrices and $\mathcal{S}^n_+$ denote the cone of $n \times n$ symmetric positive semidefinite matrices. In this paper, we report our computational experience with a first-order block-decomposition (BD) method for solving large-scale conic semidefinite programming problems (1), where

$$\mathcal{X} = \mathbb{R}^{n_u+n_l} \times \mathcal{S}^{n_s}, \qquad \mathcal{Y} = \mathbb{R}^m, \qquad K = \mathbb{R}^{n_u} \times \mathbb{R}^{n_l}_+ \times \mathcal{S}^{n_s}_+, \tag{4}$$

and the inner products in $\mathcal{X}$ and $\mathcal{Y}$ are the standard Euclidean/Frobenius inner products. Iteration-complexity bounds for this method have been studied in [12] (see also [5]). In particular, paper [12] derives the iteration-complexity of this method by using the fact that it is a special case of the hybrid proximal extragradient (HPE) method (referred to here as the HPE framework) introduced in [17, 18] by Solodov and Svaiter, and whose complexity is derived in [10, 11]. Moreover, as will be seen later on, the use of the HPE framework to analyze BD methods results in some crucial ideas towards improving their practical performance.

Though the BD methods described in [5, 12] are simple and have nice convergence properties, their implementation in its pure form is far from being efficient. This paper introduces several ingredients to the BD method of [12] to obtain a highly efficient algorithm for solving (1). The first ingredient is the use of an aggressive choice of stepsize based on a certain error criterion for performing the extragradient step. The second important idea is the implementation of the method with $\mathcal{Y}$ endowed with a scaled inner product. The third idea is to allow the scaled inner product in the $\mathcal{Y}$ space to dynamically change as the algorithm progresses, with the aim of properly balancing the sizes of the primal and dual relative residuals so as to make them go to zero according to the same order of magnitude. The fourth idea is proper choices of the initial primal and dual iterates, as well as the initial parameter for the scaled inner product.

Recently, augmented Lagrangian approaches have been proposed to solve the dual formulation (2) with $\mathcal{X}$, $\mathcal{Y}$ and K as in (4) for the case when m, n_u and n_l are large (up

to a few millions) and n_s is moderate (up to a few thousands). In [9, 15], a boundary-point method for solving (1) is proposed which can be viewed as variants of the alternating direction method of multipliers introduced in [7, 8] applied to (2). In [22], an inexact augmented Lagrangian method is proposed which solves a reformulation of the augmented Lagrangian subproblem involving only the y variable via a semismooth Newton approach combined with the conjugate gradient method. Moreover, [9, 15] and [22] report numerical results indicating that their methods are currently the best alternatives for solving large-scale conic programming problems of the form (1) and (4). In this paper, we present computational results showing that our method is faster and more robust than the ones in [9, 15] and [22] in a larger percentage of conic programming instances.

It should be noted that another highly efficient variant of the BP method, which performs a more aggressive Lagrange multiplier update, has been studied and implemented by Wen et al. in [21]. The resulting package, namely SDPAD, contains in fact a number of codes designed to solve different classes of graph-related SDP relaxations. Moreover, each code is written in such a way as to exploit the special structure of each SDP class, without requiring the input to be given in standard form. Since SDPAD is not a general-purpose package (in the sense that it accepts any standard form conic SDP as input) such as the ones mentioned in the previous paragraph, we have not included it in our present computational experiments. However, in a follow-up paper [13], we have compared SDPAD with a specialized version of our BD method for solving different classes of graph-related SDP relaxations, and have found that the latter one outperforms the first one in all problem classes.

This paper is organized as follows. Section 2 presents an adaptive block-decomposition HPE framework in the context of a block-structured monotone inclusion problem. This framework is similar to the one presented in [12], but makes an aggressive choice of extragradient stepsize. An instance of this framework for solving the conic programming problem (1) in which the $\mathcal{Y}$ space is endowed with a scaled inner product is described in Sect. 3. Section 4 describes in detail all the ingredients needed to speed-up the pure form of the adaptive block-decomposition HPE method, and presents numerical results demonstrating the efficiency of the resulting algorithm for solving many large instances of (1) and (4).

1.1 Notation

The norm of a linear operator $\mathcal{A} : \mathcal{X} \to \mathcal{Y}$ is defined as

$$\|\mathcal{A}\| := \sup_{\|x\| \le 1} \|\mathcal{A}x\|.$$

The norm of the pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ is defined as

$$\|(x, y)\| := \sqrt{\|x\|^2 + \|y\|^2}.$$

Let a closed convex set $C \subset \mathcal{X}$ be given. The projection operator $\Pi_C : \mathcal{X} \to C$ onto C and the distance function $\mathrm{dist}_C : \mathcal{X} \to \mathbb{R}_+$ with respect to C are defined as

$$\Pi_C(x) := \arg\min_{\tilde{x} \in C} \{\|x - \tilde{x}\|\}, \qquad \mathrm{dist}_C(x) := \min_{\tilde{x} \in C} \{\|x - \tilde{x}\|\}, \quad \forall x \in \mathcal{X}. \tag{5}$$

Finally, the indicator function $\delta_C : \mathcal{X} \to \bar{\mathbb{R}}$ of C is defined as

$$\delta_C(x) := \begin{cases} 0, & x \in C, \\ \infty, & x \notin C, \end{cases}$$

and the normal cone operator $N_C : \mathcal{X} \rightrightarrows \mathcal{X}$ for C is the point-to-set map given by

$$N_C(x) := \begin{cases} \emptyset, & x \notin C, \\ \{w \in \mathcal{X} : \langle \tilde{x} - x, w \rangle \leq 0, \forall \tilde{x} \in C\}, & x \in C. \end{cases}$$

Clearly, the normal cone operator N_C of C can be expressed in terms of its indicator function as $N_C = \partial \delta_C$ (see Sect. 2.1 for the definition of the subdifferential of a map).

2 An adaptive block-decomposition HPE framework

In this section, we discuss an adaptive block-decomposition HPE (A-BD-HPE) framework which is an extension of the BD-HPE framework introduced in [12]. This framework is analyzed in the context of a block-structured monotone inclusion problem similar to the one in Sect. 3 of [12], but with the addition of an adaptive (and aggressive) stepsize choice for performing the extragradient step. This section is divided into two subsections. The first one reviews some basic definitions and facts about ε-subdifferentials of functions and ε-enlargements of monotone operators. The second one presents the A-BD-HPE framework.

2.1 The ε-subdifferential and ε-enlargement of monotone operators

Let $\mathcal{Z}$ denote a finite dimensional inner product space. A point-to-set operator $T : \mathcal{Z} \rightrightarrows \mathcal{Z}$ is a relation $T \subset \mathcal{Z} \times \mathcal{Z}$ and

$$T(z) := \big\{v \in \mathcal{Z} : (z, v) \in T\big\}.$$

Alternatively, one can consider T as a multi-valued function of $\mathcal{Z}$ into the family $\wp(\mathcal{Z}) = 2^{(\mathcal{Z})}$ of subsets of $\mathcal{Z}$. Regardless of the approach, it is usual to identify T with its graph defined as

$$Gr(T) := \big\{(z, v) \in \mathcal{Z} \times \mathcal{Z} : v \in T(z)\big\}.$$

The domain of T , denoted by $\mathrm{Dom}\, T$, is defined as

$$\mathrm{Dom}\, T := \big\{z \in \mathcal{Z} : T(z) \neq \emptyset\big\}.$$

An operator $T : \mathcal{Z} \rightrightarrows \mathcal{Z}$ is *affine* if its graph is an affine manifold. Moreover, $T : \mathcal{Z} \rightrightarrows \mathcal{Z}$ is monotone if

$$\langle v - \tilde{v}, z - \tilde{z} \rangle \geq 0, \quad \forall (z, v), (\tilde{z}, \tilde{v}) \in Gr(T),$$

and T is maximal monotone if it is monotone and maximal in the family of monotone operators with respect to the partial order of inclusion, i.e., $S : \mathcal{Z} \rightrightarrows \mathcal{Z}$ monotone and $Gr(S) \supset Gr(T)$ implies that $S = T$.

In [3], Burachik, Iusem and Svaiter introduced the ε-enlargement of maximal monotone operators. In [10] this concept was extended to a generic point-to-set operator in $\mathcal{Z}$ as follows. Given $T : \mathcal{Z} \rightrightarrows \mathcal{Z}$ and a scalar ε, define $T^\varepsilon : \mathcal{Z} \rightrightarrows \mathcal{Z}$ as

$$T^\varepsilon(z) := \left\{v \in \mathcal{Z} : \langle z - \tilde{z}, v - \tilde{v}\rangle \geq -\varepsilon, \forall \tilde{z} \in \mathcal{Z}, \forall \tilde{v} \in T(\tilde{z})\right\}, \quad \forall z \in \mathcal{Z}.$$

The following result, whose proof can be found in Lemma 3.3 in [10], gives the characterization of the ε-enlargement of the normal cone of a closed convex cone.

Lemma 2.1 (Lemma 3.3 in [10]) *If K is a nonempty closed convex cone and K^* is its dual cone defined in (3), then for every $x \in K$, we have*

$$-q \in (N_K)^\varepsilon(x) \quad \Longleftrightarrow \quad q \in K^*, \quad \langle x, q\rangle \leq \varepsilon.$$

For a scalar $\varepsilon \geq 0$, the ε-subdifferential of a function $f : \mathcal{X} \to \bar{\mathbb{R}}$ is the operator $\partial_\varepsilon f : \mathcal{X} \rightrightarrows \mathcal{X}$ defined as

$$\partial_\varepsilon f(x) := \left\{w \in \mathcal{X} : f(\tilde{x}) \geq f(x) + \langle \tilde{x} - x, w\rangle - \varepsilon, \forall \tilde{x} \in \mathcal{X}\right\}, \quad \forall x \in \mathcal{X}.$$

When $\varepsilon = 0$, the operator $\partial_\varepsilon f$ is simply denoted by ∂f and is referred to as the subdifferential of f. The operator ∂f is trivially monotone if f is proper. If f is a proper lower semi-continuous convex function, then ∂f is maximal monotone [16].

Finally, we refer the reader to [2, 19] for further discussion on the ε-enlargement of a maximal monotone operator.

2.2 The A-BD-HPE framework

In this subsection, we discuss the A-BD-HPE framework which is an extension of the BD-HPE framework introduced in [12].

This framework is analyzed in the context of a block-structured monotone inclusion problem similar to the one in Sect. 3 of [12]. In what follows we give the details of this block-structured monotone inclusion problem.

Let $\langle \cdot, \cdot\rangle_{\mathcal{X}}$ and $\langle \cdot, \cdot\rangle_{\mathcal{Y}}$ be arbitrary inner products in $\mathcal{X}$ and $\mathcal{Y}$, respectively, and denote their induced norms by $\|\cdot\|_{\mathcal{X}}$ and $\|\cdot\|_{\mathcal{Y}}$, respectively. We endow the product space $\mathcal{X} \times \mathcal{Y}$ with the inner product $\langle \cdot, \cdot\rangle_{\mathcal{X},\mathcal{Y}}$ defined as

$$\left\langle (x, y), \left(x', y'\right)\right\rangle_{\mathcal{X},\mathcal{Y}} := \left\langle x, x'\right\rangle_{\mathcal{X}} + \left\langle y, y'\right\rangle_{\mathcal{Y}}, \quad \forall (x, y), \left(x', y'\right) \in \mathcal{X} \times \mathcal{Y},$$

and denote its induced norm by $\|\cdot\|_{\mathcal{X},\mathcal{Y}}$. Consider the block-structured monotone inclusion problem of finding $(x, y) \in \mathcal{X} \times \mathcal{Y}$ such that

$$0 \in \big[F + (C \otimes D)\big](x, y), \tag{6}$$

where $C : \mathcal{X} \rightrightarrows \mathcal{X}$, $D : \mathcal{Y} \rightrightarrows \mathcal{Y}$ and the operator $C \otimes D : \mathcal{X} \times \mathcal{Y} \rightrightarrows \mathcal{X} \times \mathcal{Y}$ is defined as

$$(C \otimes D)(x, y) = C(x) \times D(y), \quad \forall (x, y) \in \mathcal{X} \times \mathcal{Y}.$$

We make the following assumptions regarding (6):

A.1 C and D are maximal monotone operators (with respect to $\langle\cdot,\cdot\rangle_{\mathcal{X}}$ and $\langle\cdot,\cdot\rangle_{\mathcal{Y}}$, respectively);

A.2 $F : \mathrm{Dom}\, F \subseteq \mathcal{X}\times\mathcal{Y} \to \mathcal{X}\times\mathcal{Y}$ is a continuous map such that $\mathrm{Dom}\, F \supseteq \mathcal{X}\times \mathrm{cl}\,(\mathrm{Dom}\, D)$;

A.3 F is monotone on $\mathrm{Dom}\, C \times \mathrm{Dom}\, D$ (with respect to $\langle(\cdot,\cdot),(\cdot,\cdot)\rangle_{\mathcal{X},\mathcal{Y}}$);

A.4 there exists $L_{yx} > 0$ such that

$$\big\|F_y(x',y) - F_y(x,y)\big\|_{\mathcal{Y}} \le L_{yx}\big\|x'-x\big\|_{\mathcal{X}}, \quad \forall y \in \mathrm{Dom}\, D,\ \forall x,x' \in \mathcal{X}.$$

Assumption A.1 implies that $C\otimes D$ is maximal monotone. Hence, in view of Assumption A.2 above and Proposition A.1 of [11], it follows that the operator $F + C\otimes D$ in (6) is maximal monotone.

We now state an extension of the BD-HPE framework of [12] which uses an adaptive rule for aggressively choosing the extragradient stepsize.

A-BD-HPE Framework: An adaptive block-decomposition HPE framework for (6)

(0) Let $x_0 \in \mathcal{X}$, $y_0 \in \mathcal{Y}$, $\sigma \in (0,1]$, $\sigma_y \in [0,1]$ and $\sigma_x, \tilde{\sigma}_y \in [0,1)$ be given and set $k = 1$;

(1) choose $\tilde{\lambda}_k > 0$ such that

$$\sigma_k := \lambda_{\max}\left(\begin{bmatrix} \sigma_y^2 & \tilde{\lambda}_k\tilde{\sigma}_y L_{yx} \\ \tilde{\lambda}_k\tilde{\sigma}_y L_{yx} & \sigma_x^2 + \tilde{\lambda}_k^2 L_{yx}^2 \end{bmatrix}\right)^{1/2} \le \sigma; \tag{7}$$

(2) compute $\tilde{y}_k, \tilde{d}_k \in \mathcal{Y}$ and $\epsilon_k^y \ge 0$ such that

$$\tilde{d}_k \in D^{\epsilon_k^y}(\tilde{y}_k), \quad \big\|\tilde{\lambda}_k\big[F_y(x_{k-1},\tilde{y}_k)+\tilde{d}_k\big]+\tilde{y}_k - y_{k-1}\big\|_{\mathcal{Y}}^2 + 2\tilde{\lambda}_k\epsilon_k^y \le \sigma_y^2\|\tilde{y}_k - y_{k-1}\|_{\mathcal{Y}}^2, \tag{8}$$

$$\big\|\tilde{\lambda}_k\big[F_y(x_{k-1},\tilde{y}_k)+\tilde{d}_k\big]+\tilde{y}_k - y_{k-1}\big\|_{\mathcal{Y}} \le \tilde{\sigma}_y\|\tilde{y}_k - y_{k-1}\|_{\mathcal{Y}}; \tag{9}$$

compute $\tilde{x}_k, \tilde{c}_k \in \mathcal{X}$ and $\epsilon_k^x \ge 0$ such that

$$\tilde{c}_k \in C^{\epsilon_k^x}(\tilde{x}_k),$$

$$\big\|\tilde{\lambda}_k\big[F_x(\tilde{x}_k,\tilde{y}_k)+\tilde{a}_k\big]+\tilde{x}_k - x_{k-1}\big\|_{\mathcal{X}}^2 + 2\tilde{\lambda}_k\epsilon_k^x \le \sigma_x^2\|\tilde{x}_k - x_{k-1}\|_{\mathcal{X}}^2; \tag{10}$$

(3) choose λ_k to be the largest $\lambda > 0$ such that

$$\begin{aligned}&\big\|\lambda\big[F(\tilde{x}_k,\tilde{y}_k)+(\tilde{c}_k,\tilde{d}_k)\big]+(\tilde{x}_k,\tilde{y}_k)-(x_{k-1},y_{k-1})\big\|_{\mathcal{X},\mathcal{Y}}^2 + 2\lambda\big(\epsilon_k^x+\epsilon_k^y\big)\\ &\quad\le \sigma^2\big\|(\tilde{x}_k,\tilde{y}_k)-(x_{k-1},y_{k-1})\big\|_{\mathcal{X},\mathcal{Y}}^2;\end{aligned} \tag{11}$$

(4) set

$$(x_k,y_k) = (x_{k-1},y_{k-1}) - \lambda_k\big[F(\tilde{x}_k,\tilde{y}_k)+(\tilde{c}_k,\tilde{d}_k)\big], \tag{12}$$

and $k \leftarrow k+1$, and go to step 1.

The A-BD-HPE framework is a more aggressive version of the BD-HPE framework studied in [12]. In contrast to the A-BD-HPE framework which chooses the extragradient stepsize as the largest scalar satisfying (11), the BD-HPE framework in [12] performs the extragradient step with $\lambda_k = \tilde{\lambda}_k$. The following result shows that $\tilde{\lambda}_k$ satisfies (11) and, as a consequence, guarantees the well-definedness of the adaptive extragradient stepsize λ_k in step 3 of the A-BD-HPE framework.

Proposition 2.2 *Consider the sequences* $\{(x_k, y_k)\}, \{(\tilde{x}_k, \tilde{y}_k)\}, \{(\tilde{c}_k, \tilde{d}_k)\}$, $\{\tilde{\lambda}_k\}$ *and* $\{(\epsilon_k^x, \epsilon_k^y)\}$ *generated by the A-BD-HPE framework. Then, for every* $k \in \mathbb{N}$,

$$F(\tilde{x}_k, \tilde{y}_k) + (\tilde{c}_k, \tilde{d}_k) \in \left[F + (C \otimes D)^{\epsilon_k^x + \epsilon_k^y}\right](\tilde{x}_k, \tilde{y}_k) \subset \left[F + (C \otimes D)\right]^{\epsilon_k^x + \epsilon_k^y}(\tilde{x}_k, \tilde{y}_k) \tag{13}$$

and $\lambda = \tilde{\lambda}_k$ *satisfies* (11). *As a consequence* $\lambda_k \geq \tilde{\lambda}_k$.

Proof This result follows from Proposition 3.1 in [12]. □

In view of (11), (12) and (13), it follows that the A-BD-HPE framework is a special case of the HPE framework for solving (6). This observation allows us to obtain complexity results for the A-BD-HPE framework using the general complexity results derived in [10]. In what follows, we state two convergence results whose proofs are analogous to those of Theorems 3.2 and 3.3 of [12], but use Proposition 2.2 above instead of Proposition 3.1 in [12].

Theorem 2.3 *Assume that* $\sigma < 1$ *and consider the sequences* $\{(\tilde{x}_k, \tilde{y}_k)\}, \{(\tilde{c}_k, \tilde{d}_k)\}$, $\{\lambda_k\}$ *and* $\{(\epsilon_k^x, \epsilon_k^y)\}$ *generated by the A-BD-HPE framework and let* d_0 *denote the distance of the initial point* $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}$ *to the solution set of* (6) *with respect to* $\|(\cdot, \cdot)\|_{\mathcal{X},\mathcal{Y}}$. *Then, for every* $\alpha \in \mathbb{R}$ *and* $k \in \mathbb{N}$,

$$(\tilde{c}_k, \tilde{d}_k) \in C^{\epsilon_k^x}(\tilde{x}_k) \times D^{\epsilon_k^y}(\tilde{y}_k),$$

and there exists $i \leq k$ *such that*

$$\left\|F(\tilde{x}_i, \tilde{y}_i) + (\tilde{c}_i, \tilde{d}_i)\right\|_{\mathcal{X},\mathcal{Y}} \leq d_0 \sqrt{\frac{(1+\sigma)}{(1-\sigma)}\left(\frac{\lambda_i^{\alpha-2}}{\sum_{j=1}^k \lambda_j^\alpha}\right)},$$

$$\epsilon_i^x + \epsilon_i^y \leq \frac{d_0^2 \sigma^2}{2(1-\sigma^2)}\left(\frac{\lambda_i^{\alpha-1}}{\sum_{j=1}^k \lambda_j^\alpha}\right).$$

Theorem 2.4 *Assume that* F *is affine and consider the sequences* $\{(\tilde{x}_k, \tilde{y}_k)\}$, $\{(\tilde{c}_k, \tilde{d}_k)\}$, $\{\lambda_k\}$ *and* $\{(\epsilon_k^x, \epsilon_k^y)\}$ *generated by the A-BD-HPE framework and define*

for every $k \in \mathbb{N}$:

$$\begin{aligned}\left(\tilde{x}_k^a, \tilde{y}_k^a\right) &= \frac{1}{\Lambda_k}\sum_{i=1}^{k}\lambda_i(\tilde{x}_i, \tilde{y}_i),\\ \left(\tilde{c}_k^a, \tilde{d}_k^a\right) &= \frac{1}{\Lambda_k}\sum_{i=1}^{k}\lambda_i(\tilde{c}_k, \tilde{d}_k),\end{aligned} \tag{14}$$

and

$$\epsilon_k^{x,a} := \frac{1}{\Lambda_k}\sum_{i=1}^{k}\lambda_i\left(\epsilon_i^x + \left\langle \tilde{x}_i - \tilde{x}_k^a, \tilde{c}_i\right\rangle\right) \geq 0,$$

$$\epsilon_k^{y,a} := \frac{1}{\Lambda_k}\sum_{i=1}^{k}\lambda_i\left(\epsilon_i^y + \left\langle \tilde{y}_i - \tilde{y}_k^a, \tilde{d}_i\right\rangle\right) \geq 0,$$

where $\Lambda_k = \sum_{i=1}^{k}\lambda_i$. *Let* d_0 *denote the distance of the initial point* $(x_0, y_0) \in \mathcal{X}\times\mathcal{Y}$ *to the solution set of* (6) *with respect to* $\|(\cdot,\cdot)\|_{\mathcal{X},\mathcal{Y}}$ *and* $\sigma_{xy} = \max\{\sigma_x, \sigma_y\}$. *Then, for every* $k \in \mathbb{N}$,

$$\left(\tilde{c}_k^a, \tilde{d}_k^a\right) \in C^{\epsilon_k^{x,a}}\left(\tilde{x}_k^a\right) \times D^{\epsilon_k^{y,a}}\left(\tilde{y}_k^a\right), \qquad \left\|F\left(\tilde{x}_k^a, \tilde{y}_k^a\right) + \left(\tilde{c}_k^a, \tilde{d}_k^a\right)\right\|_{\mathcal{X},\mathcal{Y}} \leq \frac{2d_0}{\Lambda_k},$$

$$\epsilon_k^{x,a} + \epsilon_k^{y,a} \leq \frac{2d_0^2}{\Lambda_k}(1+\bar{\eta}_k),$$

where

$$\bar{\eta}_k := \frac{2\sqrt{2}\sigma}{1-\sigma_{xy}}\left(1 + \frac{1}{(1-\sigma_y)^2}\right)^{1/2}.$$

Observe that the convergence rate bounds described in Theorems 2.3 (with $\alpha = 1$) and 2.4 suggest that the rate of convergence of the A-BD-HPE framework becomes better the larger the stepsize λ_k is chosen (under the condition that (11) is satisfied so as to guarantee that Theorems 2.3 and 2.4 still apply). In fact, the choice of λ_k at step 3 of the A-BD-HPE framework is motivated by this observation.

3 A scaled A-BD method for conic programming

In this section, we introduce an instance of the A-BD-HPE framework applied to (1) in which $\langle\cdot,\cdot\rangle_{\mathcal{X}} = \langle\cdot,\cdot\rangle$ and $\langle\cdot,\cdot\rangle_{\mathcal{Y}}$ is a scaled inner product constructed by means of the original inner product $\langle\cdot,\cdot\rangle$ in $\mathcal{Y}$. (Recall that the original inner products in $\mathcal{X}$ and $\mathcal{Y}$ are both being denoted by $\langle\cdot,\cdot\rangle$.) Even though there is nothing new in this section from the theoretical point of view, we will use the convergence results of Sect. 2 to give a plausible argument showing that an appropriate choice of scaled inner product in the $\mathcal{Y}$ space may lead to a substantial speed-up of the BD method relative to its unscaled version.

We consider problem (1) with the following assumptions:

C.1 $\mathcal{A} : \mathcal{X} \to \mathcal{Y}$ is a surjective linear map and $b \in \mathcal{Y}$;
C.2 there exists $x^* \in \mathcal{X}$ satisfying the inclusion

$$0 \in c + \partial \delta_K(x) + N_{\mathcal{M}}(x), \tag{15}$$

where $\mathcal{M} := \{x \in \mathcal{X} : \mathcal{A}(x) = b\}$.

We now make a few observations about the above assumptions. First, any x^* as in C.2 is an optimal solution of (1). Second, observe that a sufficient condition for (15) to hold is that (1) has an optimal solution and satisfies the Slater condition, i.e. $\mathcal{A}\hat{x} = b$ for some $\hat{x} \in \mathrm{ri}(K)$. Third, (15) is equivalent to the existence of $y^* \in \mathcal{Y}$ such that the pair (x^*, y^*) satisfies the inclusion

$$c + \partial \delta_K(x) - \mathcal{A}^* y \ni 0, \quad \mathcal{A}x - b = 0. \tag{16}$$

Fourth, the set of solutions of the above inclusion is exactly $\mathcal{X}^* \times \mathcal{Y}^*$, where $\mathcal{X}^*$ and $\mathcal{Y}^*$ denote the set of optimal solutions of (1) and (2), respectively. Fifth, observe that (16) can be easily put into the form (6) and that Assumptions A.1–A.4 all hold when $\mathcal{X}$ and $\mathcal{Y}$ are both endowed with the original inner product $\langle \cdot, \cdot \rangle$. Hence, one can apply any instance of the A-BD-HPE framework to solve (16).

However, from the computational point of view, it is more efficient to introduce a scaled inner product in the $\mathcal{Y}$ space and work with a scaled version of (16). More specifically, given a self adjoint positive definite linear mapping $\mathcal{U} : \mathcal{Y} \to \mathcal{Y}$, endow $\mathcal{X}$ and $\mathcal{Y}$ with the inner products defined as

$$\langle \cdot, \cdot \rangle_{\mathcal{X}} := \langle \cdot, \cdot \rangle, \qquad \langle \cdot, \cdot \rangle_{\mathcal{Y}} := \langle \cdot, \mathcal{U} \cdot \rangle, \tag{17}$$

respectively, and define F, C and D as

$$F(x, y) := \begin{pmatrix} c - \mathcal{A}^* y \\ \mathcal{U}^{-1}(\mathcal{A}x - b) \end{pmatrix}, \qquad C(x) = \partial \delta_K(x) = N_K(x), \qquad D(y) = 0, \tag{18}$$

where the normal cone $N_K(x)$ is with respect to $\langle \cdot, \cdot \rangle_{\mathcal{X}} := \langle \cdot, \cdot \rangle$.

The following proposition can be easily shown.

Proposition 3.1 *F, C and D defined in* (18) *and the above inner products defined in* (17) *satisfy Assumptions* A.1–A.4 *of Sect.* 2 *with* $L_{yx} = \|\mathcal{U}^{-1/2}\mathcal{A}\|$.

As a consequence of the above proposition, any instance of the A-BD-HPE framework of Sect. 2 with F, C and D as in (18) and the inner products $\langle \cdot, \cdot \rangle_{\mathcal{X}}$ and $\langle \cdot, \cdot \rangle_{\mathcal{Y}}$ defined as in (17) will satisfy the global convergence rate properties described in Theorems 2.3 and 2.4. Below we describe such an instance.

Algorithm 1: Scaled adaptive block-decomposition (SA-BD) method for solving (1)

(0) Let $x_0 \in \mathcal{X}$, $y_0 \in \mathcal{Y}$, $0 < \sigma \leq 1$ and $\theta > 0$ be given, and set $k = 1$ and

$$\tilde{\lambda} = \frac{\sigma}{\|\mathcal{U}^{-1/2}\mathcal{A}\|}; \tag{19}$$

(1) compute

$$\tilde{y}_k = y_{k-1} - \tilde{\lambda}\mathcal{U}^{-1}(\mathcal{A}x_{k-1} - b), \qquad \tilde{x}_k = \Pi_K\big[x_{k-1} - \tilde{\lambda}\big(c - \mathcal{A}^*\tilde{y}_k\big)\big]; \tag{20}$$

(2) define

$$\tilde{v}_k = \begin{pmatrix} (x_{k-1} - \tilde{x}_k)/\tilde{\lambda} \\ \mathcal{U}^{-1}\big(\mathcal{A}\tilde{x}_k - b\big) \end{pmatrix}, \tag{21}$$

choose λ_k to be the largest $\lambda > 0$ such that

$$\left\| \lambda\tilde{v}_k + \begin{pmatrix} \tilde{x}_k \\ \tilde{y}_k \end{pmatrix} - \begin{pmatrix} x_{k-1} \\ y_{k-1} \end{pmatrix} \right\|_{\mathcal{X},\mathcal{Y}} \leq \sigma \left\| \begin{pmatrix} \tilde{x}_k \\ \tilde{y}_k \end{pmatrix} - \begin{pmatrix} x_{k-1} \\ y_{k-1} \end{pmatrix} \right\|_{\mathcal{X},\mathcal{Y}};$$

(3) set $(x_k, y_k) = (x_{k-1}, y_{k-1}) - \lambda_k\tilde{v}_k$ and $k \leftarrow k + 1$, and go to step 1.

Proposition 3.2 *Let $\sigma_x = \sigma_y = \tilde{\sigma}_y = 0$ and define the inner products $\langle\cdot,\cdot\rangle_{\mathcal{X}}$ and $\langle\cdot,\cdot\rangle_{\mathcal{Y}}$, and operators F, C and D according to (17) and (18). Consider the sequences $\{(x_k, y_k)\}$ and $\{(\tilde{x}_k, \tilde{y}_k)\}$ generated by Algorithm 1 and, for every $k \in \mathbb{N}$, define*

$$\tilde{\lambda}_k = \tilde{\lambda}, \qquad \epsilon_k^x = \epsilon_k^y = 0, \qquad \tilde{d}_k = 0, \qquad \tilde{c}_k = -\tilde{z}_k, \tag{22}$$

where

$$\tilde{z}_k := c - \mathcal{A}^*\tilde{y}_k - \frac{1}{\tilde{\lambda}}(x_{k-1} - \tilde{x}_k). \tag{23}$$

Then the following statements hold for every $k \in \mathbb{N}$:

(a) *$\tilde{\lambda}_k$ satisfies (7) with $L_{yx} = \|\mathcal{U}^{-1/2}\mathcal{A}\|$;*
(b) *$\tilde{x}_k \in K$ and $\tilde{z}_k \in -N_K(\tilde{x}_k)$, or equivalently, $\tilde{x}_k \in K$, $\tilde{z}_k \in K^*$ and $\langle\tilde{x}_k, \tilde{z}_k\rangle = 0$;*
(c) *$\tilde{\lambda}_k$, y_{k-1}, x_{k-1}, and the triples $(\tilde{y}_k, \tilde{d}_k, \epsilon_k^y)$ and $(\tilde{x}_k, \tilde{c}_k, \epsilon_k^x)$ satisfy (8), (9) and (10).*

As a consequence, Algorithm 1 is a special instance of the A-BD-HPE framework.

Proof (a) This statement follows straightforwardly from (19).

(b) Define $w_k := x_{k-1} - \tilde{\lambda}(c - \mathcal{A}^*\tilde{y}_k)$ and note that $\tilde{x}_k = \Pi_K(w_k) \in K$ in view of (20). This together with (23) then imply that

$$\tilde{\lambda}\tilde{z}_k = \tilde{\lambda}\big(c - \mathcal{A}^*\tilde{y}_k\big) - \big(x_{k-1} - \Pi_K(w_k)\big) = -\big(w_k - \Pi_K(w_k)\big) \in -N_K(\tilde{x}_k), \tag{24}$$

where the inclusion follows from the well-known fact that $x - \Pi_K(x) \in N_K(\Pi_K(x))$ for all $x \in \mathcal{X}$. Statement b) now follows from the above observations and the fact

that $N_K(\tilde{x}_k)$ is a cone. The equivalent statement of b) follows from Lemma 2.1 with $q = \tilde{z}_k$, $x = \tilde{x}_k$ and $\varepsilon = 0$.

(c) It follows from (22), (23), the definition of F in (18), and the definition of $\tilde{y}_k$ in (20) that

$$\tilde{\lambda}_k \left[F_y(x_{k-1}, \tilde{y}_k) + \tilde{d}_k \right] + \tilde{y}_k - y_{k-1} = 0, \qquad \tilde{\lambda}_k \left[F_x(\tilde{x}_k, \tilde{y}_k) + \tilde{c}_k \right] + \tilde{x}_k - x_{k-1} = 0.$$

Clearly, the fact that $\sigma_x = \sigma_y = \tilde{\sigma}_y = \epsilon_k^x = \epsilon_k^y = 0$ and the two identities above imply that all the inequalities in (8), (9) and (10) are satisfied. The inclusions in (8) and (10) hold from the definitions of D, ϵ_k^y, $\tilde{d}_k$, C, ϵ_k^x and $\tilde{c}_k$ in (18) and (22), and the inclusion in (24). Hence, statement c) follows.

Finally, the definitions of F in (18), $\tilde{v}_k$ in (21), and $\tilde{c}_k$ and $\tilde{d}_k$ in (22), imply that

$$\tilde{v}_k = F(\tilde{x}_k, \tilde{y}_k) + (\tilde{c}_k, \tilde{d}_k).$$

This observation together with the fact that $\epsilon_k^x = \epsilon_k^y = 0$ then imply that steps 2 and 3 of Algorithm 1 are equivalent to steps 3 and 4 of the A-BD-HPE framework. Therefore, the conclusion of the proposition follows. □

We now specialize the convergence rate results of Sect. 2, namely Theorems 2.3 and 2.4, to the context of Algorithm 1. First, define for every non-singular linear mapping $\mathcal{B} : \mathcal{Y} \to \mathcal{Y}$ and $y \in \mathcal{Y} \setminus \{0\}$ the following quantity

$$\xi(\mathcal{B}, y) := \frac{\|\mathcal{B}y\|}{\|\mathcal{B}\|\|y\|} \in \left[\frac{1}{\kappa(\mathcal{B})}, 1 \right], \tag{25}$$

where $\kappa(\mathcal{B}) := \|\mathcal{B}^{-1}\|\|\mathcal{B}\|$ denotes the condition number of $\mathcal{B}$. Note that for any scalar $\theta \in \mathbb{R} \setminus \{0\}$ we have that $\xi(\theta\mathcal{B}, y) = \xi(\mathcal{B}, y)$ and $\xi(\theta\mathcal{I}, y) = 1$.

Theorem 3.3 *Consider the sequences $\{(x_k, y_k)\}$ and $\{(\tilde{x}_k, \tilde{y}_k)\}$ generated by Algorithm 1, the sequence $\{\tilde{z}_k\}$ defined as in (23), the sequences $\{(\tilde{x}_k^a, \tilde{y}_k^a)\}$ and $\{\tilde{c}_k^a\}$ defined as in (14), and the sequence $\{\tilde{z}_k^a\}$ defined by $\tilde{z}_k^a = -\tilde{c}_k^a$ for every $k \in \mathbb{N}$. Let $\mathcal{X}^*$ and $\mathcal{Y}^*$ denote the set of optimal solutions of (1) and (2), respectively, and $(x^*, y^*) \in \mathcal{X}^* \times \mathcal{Y}^*$ be such that*

$$\begin{aligned} d_{0,x} &:= \min\{\|x_0 - x\| : x \in \mathcal{X}^*\} = \|x_0 - x^*\|, \\ d_{0,y} &:= \min\{\|y_0 - y\| : y \in \mathcal{Y}^*\} = \|y_0 - y^*\|. \end{aligned} \tag{26}$$

Then, for every $k \in \mathbb{N}$, the following statements hold:

(a) $\tilde{x}_k \in K$, $\tilde{z}_k \in K^*$, $\langle \tilde{x}_k, \tilde{z}_k \rangle = 0$, *and if $\sigma < 1$, there exists $i \le k$ such that*

$$\begin{aligned} &\left\| \mathcal{A}^* \tilde{y}_i + \tilde{z}_i - c \right\|^2 + \left\| \mathcal{U}^{-1/2}(\mathcal{A}\tilde{x}_i - b) \right\|^2 \\ &\quad \le \left(\frac{1+\sigma}{1-\sigma} \right) \left\| \mathcal{U}^{-1/2}\mathcal{A} \right\|^2 \frac{d_{0,x}^2 + \xi_0(\mathcal{U})\|\mathcal{U}\| d_{0,y}^2}{k\sigma^2}, \end{aligned} \tag{27}$$

where $\xi_0(\mathcal{U}) := [\xi(\mathcal{U}^{1/2}, y_0 - y^*)]^2$.

(b) $\tilde{x}_k^a \in K$, $\tilde{z}_k^a \in K^*$ *and*

$$\left\|\mathcal{A}^*\hat{y}_k^a + \tilde{z}_k^a - c\right\|^2 + \left\|\mathcal{U}^{-1/2}\left(\mathcal{A}\tilde{x}_k^a - b\right)\right\|^2 \le \left\|\mathcal{U}^{-1/2}\mathcal{A}\right\|^2 \frac{d_{0,x}^2 + \xi_0(\mathcal{U})\|\mathcal{U}\| d_{0,y}^2}{(k\sigma)^2},$$

$$\left\langle \tilde{x}_k^a, \tilde{z}_k^a \right\rangle \le (2+8\sigma)\left\|\mathcal{U}^{-1/2}\mathcal{A}\right\| \frac{d_{0,x}^2 + \xi_0(\mathcal{U})\|\mathcal{U}\| d_{0,y}^2}{k\sigma}.$$

Proof We first prove statement (a). Let $k \in \mathbb{N}$ be given. First note that from Proposition 3.2(d) we have $\tilde{z}_k \in K^*$ and $\langle \tilde{x}_k, \tilde{z}_k \rangle = 0$. Observe also that $\mathcal{X}^* \times \mathcal{Y}^*$ is the set of solutions of the inclusion problem (6) and (18) (see the fourth observation after (15)). Let d_0 denote the distance of (x_0, y_0) to $\mathcal{X}^* \times \mathcal{Y}^*$ with respect to the scaled norm $\|\cdot\|_{\mathcal{X},\mathcal{Y}}$, and observe that the definitions of (x^*, y^*), $d_{0,x}$, $d_{0,y}$ and $\xi_0(\mathcal{U})$ imply

$$\begin{aligned} d_0^2 &\le \left\|x_0 - x^*\right\|_{\mathcal{X}}^2 + \left\|y_0 - y^*\right\|_{\mathcal{Y}}^2 = \left\|x_0 - x^*\right\|^2 + \left\|\mathcal{U}^{1/2}\left(y_0 - y^*\right)\right\| \\ &= d_{0,x}^2 + \xi_0(\mathcal{U})\|\mathcal{U}\| d_{0,y}^2. \end{aligned}$$

Moreover, by Proposition 3.2 and Theorem 2.3 with $\alpha = 1$, we conclude that if $\sigma < 1$, there exists $i \le k$ such that

$$\begin{aligned} &\left\|c - \mathcal{A}^*\tilde{y}_i + \tilde{c}_i\right\|_{\mathcal{X}}^2 + \left\|\mathcal{U}^{-1}\left(\mathcal{A}\tilde{x}_i - b\right) + \tilde{d}_i\right\|_{\mathcal{Y}}^2 \\ &\le \left(\frac{1+\sigma}{1-\sigma}\right)\frac{d_0^2}{\lambda_i \sum_{j=1}^k \lambda_j} \le \left(\frac{1+\sigma}{1-\sigma}\right)\frac{d_0^2}{\tilde{\lambda}^2 k} \\ &= \left(\frac{1+\sigma}{1-\sigma}\right)\left\|\mathcal{U}^{-1/2}\mathcal{A}\right\|^2 \frac{d_0^2}{k\sigma^2}, \end{aligned}$$

where the second inequality follows from Proposition 2.2 with $\tilde{\lambda}_k = \tilde{\lambda}$, and the last equality follows from the definition of $\tilde{\lambda}$ in (19). Also, in view of (17) and the definitions of $\tilde{d}_i$ and $\tilde{c}_i$ in (22), we have

$$\begin{aligned} \left\|c - \mathcal{A}^*\tilde{y}_i + \tilde{c}_i\right\|_{\mathcal{X}}^2 &= \left\|\mathcal{A}^*\tilde{y}_i + \tilde{z}_i - c\right\|^2, \\ \left\|\mathcal{U}^{-1}(\mathcal{A}\tilde{x}_i - b) + \tilde{d}_i\right\|_{\mathcal{Y}}^2 &= \left\|\mathcal{U}^{-1/2}(\mathcal{A}\tilde{x}_i - b)\right\|^2. \end{aligned}$$

Then, combining the last four relations we obtain (27), and hence statement (a) follows.

Statement (b) can be proved in a similar way using Theorem 2.4 instead of Theorem 2.3. □

We now make several remarks about Theorem 3.3. For the sake of simplicity, we will focus our discussion on the point-wise convergence rate bound (27). Define the self-adjoint positive definite linear mapping $\mathcal{U}_0 : \mathcal{Y} \to \mathcal{Y}$ as

$$\mathcal{U}_0 := \mathcal{A}\mathcal{A}^*.$$

First, the term $\|\mathcal{U}^{-1/2}\mathcal{A}\|$ in the right hand side of (27) is minimized over the class $\mathcal{C}(\mathcal{A})$ consisting of all self-adjoint positive definite linear mappings $\mathcal{U}:\mathcal{Y}\to\mathcal{Y}$ satisfying $\|\mathcal{U}^{-1/2}\mathcal{A}\|^2=\|\mathcal{A}\|^2/\|\mathcal{U}\|$, or equivalently,

$$\left\|\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}\right\|^2=\|\mathcal{U}_0\|/\|\mathcal{U}\|. \tag{28}$$

Note that any positive multiple of the identity operator $\mathcal{I}$ or the operator $\mathcal{U}_0$ belongs to $\mathcal{C}(\mathcal{A})$. In view of this remark, we will assume from now on that $\mathcal{U}\in\mathcal{C}(\mathcal{A})$, and within this class we will consider the subclass $\mathcal{C}_\theta(\mathcal{A})$ consisting of the operators $\mathcal{U}\in\mathcal{C}(\mathcal{A})$ such that $\|\mathcal{U}\|=\theta$, where $\theta>0$ is some pre-specified scalar. Second, recall that the definition of the term $\xi_0(\mathcal{U})$ implies that $\xi_0(\mathcal{U})\in[1/\kappa(\mathcal{U}),1]$ and that $\xi_0(\theta\mathcal{I})=1$. Hence, $\mathcal{U}=\theta\mathcal{I}$ maximizes $\xi_0(\mathcal{U})$ over $\mathcal{C}_\theta(\mathcal{A})$. Also, it is interesting to observe that the best possible value $\xi_0(\mathcal{U})$ might take over $\mathcal{C}(\mathcal{A})$, namely $1/\kappa(\mathcal{U})$, achieves its minimum value when $\mathcal{U}$ is a positive multiple of $\mathcal{U}_0$. Indeed, in view of (28) and the definition of $\kappa(\cdot)$, we have

$$\begin{aligned}\kappa(\mathcal{U})&=\|\mathcal{U}\|\left\|\mathcal{U}^{-1}\right\|=\|\mathcal{U}\|\left\|\mathcal{U}^{-1/2}\right\|^2\le\|\mathcal{U}\|\left\|\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}\right\|^2\left\|\mathcal{U}_0^{-1/2}\right\|^2\\&=\|\mathcal{U}\|\frac{\|\mathcal{U}_0\|}{\|\mathcal{U}\|}\left\|\mathcal{U}_0^{-1}\right\|=\kappa(\mathcal{U}_0),\quad\forall\mathcal{U}\in\mathcal{C}(\mathcal{A}).\end{aligned}$$

Third, if you view the vector $u_0:=(y_0-y^*)/\|y_0-y^*\|$ as being uniformly distributed on the unit sphere, then Lemma 5.1 and the definition of $\xi_0(\mathcal{U})$ implies that the expected value of $\xi_0(\mathcal{U})$ with respect to u_0 is $\mathrm{tr}(\mathcal{U})/(m\|\mathcal{U}\|)$, or in words, the average of the eigenvalues of $\mathcal{U}$ divided by the maximum eigenvalue of $\mathcal{U}$. Hence, if $\mathcal{U}_0$ is such that $\mathrm{tr}(\mathcal{U}_0)/(m\|\mathcal{U}_0\|)$ is of the same order of magnitude as $1/\kappa(\mathcal{U}_0)$ and $\mathcal{U}_0$ is ill-conditioned, then the choice of $\mathcal{U}=\theta\mathcal{U}_0/\|\mathcal{U}_0\|$ from the class $\mathcal{C}_\theta(\mathcal{A})$ for Algorithm 1 will be nearly optimal in the sense of minimizing $\xi_0(\mathcal{U})$. Note that the latter condition happens when $\mathcal{U}_0$ is ill-conditioned and most of the eigenvalues of $\mathcal{U}_0$ are relatively close to its minimum eigenvalue.

We will now interpret the bound (27) from a geometrical point of view. Define the primal and dual manifolds as

$$\mathcal{M}_p:=\{x:\mathcal{A}x=b\},\qquad\mathcal{M}_d:=\left\{c-\mathcal{A}^*y:y\in\mathcal{Y}\right\},$$

and define

$$\hat{\xi}_i(\mathcal{U}):=\left(\frac{1}{\xi(\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2},\mathcal{U}_0^{-1/2}(\mathcal{A}\tilde{x}_i-b))}\right)^2. \tag{29}$$

For every $\mathcal{U}\in\mathcal{C}_\theta(\mathcal{A})$, we can easily see that (27), (28), the definition of $\mathrm{dist}_C(\cdot)$ in (5), and the fact that $\|\mathcal{U}\|=\theta$ and $\|\mathcal{U}^{-1/2}\mathcal{A}\|=\|\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}\|$, imply

$$\left[\mathrm{dist}_{\mathcal{M}_d}(\tilde{z}_i)\right]^2\le\left\|\mathcal{A}^*\tilde{y}_i+\tilde{z}_i-c\right\|^2\le\left(\frac{1+\sigma}{1-\sigma}\right)\frac{\|\mathcal{U}_0\|}{k\sigma^2}\left(\frac{d_{0,x}^2}{\theta}+\xi_0(\mathcal{U})d_{0,y}^2\right). \tag{30}$$

We will now bound the distance $\mathrm{dist}_{\mathcal{M}_p}(\tilde{x}_i)$. First, it is easy to see that

$$\mathrm{dist}_{\mathcal{M}_p}(\tilde{x}_i)=\left\|\mathcal{U}_0^{-1/2}(\mathcal{A}\tilde{x}_i-b)\right\|.$$

Hence, for every $\mathcal{U} \in \mathcal{C}_\theta(\mathcal{A})$, we have that (27), (25), (29), and the fact that $\|\mathcal{U}\| = \theta$ and $\|\mathcal{U}^{-1/2}\mathcal{A}\| = \|\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}\|$, imply

$$\begin{aligned}\left[\operatorname{dist}_{\mathcal{M}_p}(\tilde{x}_i)\right]^2 &= \left\|\mathcal{U}_0^{-1/2}(\mathcal{A}\tilde{x}_i - b)\right\|^2 \\ &= \left(\frac{\|\mathcal{U}^{-1/2}(\mathcal{A}\tilde{x}_i - b)\|}{\xi(\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}, \mathcal{U}_0^{-1/2}(\mathcal{A}\tilde{x}_i - b))\|\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}\|}\right)^2 \\ &= \frac{\hat{\xi}_i(\mathcal{U})}{\|\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}\|^2}\left\|\mathcal{U}^{-1/2}(\mathcal{A}\tilde{x}_i - b)\right\|^2 \\ &\leq \left(\frac{1+\sigma}{1-\sigma}\right)\frac{\hat{\xi}_i(\mathcal{U})}{k\sigma^2}\left(d_{0,x}^2 + \xi_0(\mathcal{U})\theta d_{0,y}^2\right). \end{aligned} \tag{31}$$

Note that the definition of the term $\hat{\xi}_i(\mathcal{U})$ implies that $\hat{\xi}_i(\mathcal{U}) \in [1, (\kappa(\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}))^2]$. Hence, the choice of $\mathcal{U} = \theta\mathcal{U}_0/\|\mathcal{U}_0\|$ minimizes $\hat{\xi}_i(\mathcal{U})$ over $\mathcal{C}_\theta(\mathcal{A})$ which, in view of the observations about the term $\xi_0(\mathcal{U})$ above, can lead to nearly optimal bounds for the distances to the primal and dual manifolds in (31) and (30), respectively.

The convergence rate bounds in (30) and (31), not only highlight the benefits obtained by $\xi_0(\mathcal{U}) \leq 1$ for an ill-conditioned $\mathcal{U}$, but also suggest how the magnitude of $\|\mathcal{U}\| = \theta$ affects the size of the primal and dual residuals. More specifically, viewing all the quantities in (30) and (31), with the exception of θ, as constants, and noting that

$$\begin{aligned}\left[\operatorname{dist}_{\mathcal{M}_p}(\tilde{x}_i)\right]^2 &= \frac{\|\mathcal{U}_0^{-1/2}(\mathcal{A}\tilde{x}_i - b)\|^2\|\mathcal{U}_0^{1/2}\|^2}{\|\mathcal{A}\tilde{x}_i - b\|^2}\frac{\|\mathcal{A}\tilde{x}_i - b\|^2}{\|\mathcal{U}_0^{1/2}\|^2} \\ &= \frac{\hat{\xi}_i(\mathcal{I})\|\mathcal{A}\tilde{x}_i - b\|^2}{\|\mathcal{U}_0\|} \geq \frac{\|\mathcal{A}\tilde{x}_i - b\|^2}{\|\mathcal{U}_0\|}, \end{aligned}$$

we can see that the primal and dual residuals are

$$\|\mathcal{A}\tilde{x}_i - b\|^2 = \mathcal{O}\big(\max\{1, \theta^{1/2}\}\big), \qquad \left\|\mathcal{A}^*\tilde{y}_i + \tilde{z}_i - c\right\|^2 = \mathcal{O}\big(\max\{1, \theta^{-1/2}\}\big),$$

respectively. Hence, as $\theta \to 0$, the dual residual can become significantly larger than the primal one while, as $\theta \to \infty$, the primal residual can become significantly larger than the dual one. In fact, we have observed in our computational experiments that these residuals behave exactly as just described. In Sect. 4, we will use $\mathcal{U} = \theta\mathcal{U}_0$ and a dynamic choice of the scaling parameter θ in our implementation of Algorithm 1 so as to empirically balance the primal and dual residuals and as a consequence improve the practical performance of the method.

4 Implementation details and numerical results

In this section, we describe all the ingredients needed to speed-up the implementation of Algorithm 1, and present numerical results demonstrating the efficiency of the

resulting method for solving many large instances of (1) and (4). More specifically, we describe two important ingredients, namely: (i) convenient choice of initial primal and dual iterates, and initial parameter for the scaled inner product (17) on the space $\mathcal{X}$, and; (ii) dynamic change of the scaled inner product in the $\mathcal{X}$ space as the algorithm progresses, with the aim of properly balancing the sizes of the primal and dual relative residuals. This section also contains five subsections reporting computational results which compare our method with the ones in [9, 15] and [22] for various types of conic semidefinite programming problems.

For every $k \in \mathbb{N}$, define the primal and dual relative residuals as

$$\epsilon_{P,k} := \frac{\|\mathcal{A}\tilde{x}_k - b\|}{1 + \|b\|}, \qquad \epsilon_{D,k} := \frac{\|\mathcal{A}^*\tilde{y}_k + \tilde{z}_k - c\|}{1 + \|c\|}, \tag{32}$$

where $\{\tilde{x}_k\}$ and $\{\tilde{y}_k\}$ are the sequence generated by Algorithm 1, and $\{\tilde{z}_k\}$ is given by (23). In our implementation, we used the stopping criterion

$$\max\{\epsilon_{P,k}, \epsilon_{D,k}\} \leq \bar{\epsilon}, \tag{33}$$

where $\bar{\epsilon} > 0$ is a given tolerance. We observe that the complementarity measure is $\langle \tilde{x}_k, \tilde{z}_k \rangle = 0$ for every $k \in \mathbb{N}$, in view of Theorem 3.3(a). We note that the two methods we compare our code to also use the stopping criterion (33) and satisfy the later complementarity property.

Our implementation chooses the initial iterates x_0 and y_0 as

$$x_0 = 0, \qquad y_0 = \arg\min \left\| \mathcal{A}^* y - c \right\| = \mathcal{U}_0^{-1} \mathcal{A} c. \tag{34}$$

Another possibility would be to choose x_0 as the vector with minimum norm lying in the manifold $\{x \in \mathcal{X} : \mathcal{A}x = b\}$. However, the computational results reported in this paper are based on the choice of the initial iterates given by (34).

Our benchmark is based on an implementation of Algorithm 1 in which $\sigma = 0.99$ and the operator $\mathcal{U}$ is chosen as

$$\mathcal{U} = \theta \mathcal{U}_0, \tag{35}$$

where θ is dynamically updated whenever a specified number of iterations is performed. In the next two paragraphs we discuss how to initialize θ and the scheme for dynamically updating it.

First we discuss how to initialize θ. Note that the choice (35) of $\mathcal{U}$ implies that $\|\mathcal{U}^{-1/2}\mathcal{A}\| = \theta^{-1/2}$, and hence

$$\tilde{\lambda} = \sigma \theta^{1/2},$$

in view of (19). This observation together with (32), (23) and (34) imply that the initial relative residuals $\epsilon_{P,1}$ and $\epsilon_{D,1}$ as a function of θ are given by

$$\begin{aligned} \epsilon_{P,1} = \epsilon_{P,1}(\theta) &:= \frac{\|\mathcal{A}\tilde{x}_1(\theta) - b\|}{1 + \|b\|}, \\ \epsilon_{D,1} = \epsilon_{D,1}(\theta) &:= \frac{\|(x_0 - \tilde{x}_1(\theta))/\tilde{\lambda}\|}{1 + \|c\|} = \frac{\|\sigma^{-1}\theta^{-1/2}\tilde{x}_1(\theta)\|}{1 + \|c\|}, \end{aligned} \tag{36}$$

where

$$\tilde{x}_1 = \tilde{x}_1(\theta) := \Pi_K\big[x_0 - \tilde{\lambda}\big(c - \mathcal{A}^*\tilde{y}_1\big)\big] = \Pi_K\big[x_0 - \tilde{\lambda}\big(c - \mathcal{A}^*\big(y_0 - \tilde{\lambda}\mathcal{U}^{-1}(\mathcal{A}x_0 - b)\big)\big)\big]$$
$$= \sigma\theta^{1/2}\Pi_K\big[-c + \mathcal{A}^*\big(y_0 + \sigma\theta^{1/2}\mathcal{U}^{-1}b\big)\big].$$

Using the definition of y_0 in (34), we easily see that $\|\mathcal{A}^*y_0 - c\| \le \|c\|$, and hence , as $\theta \to 0$, we have from (36) that

$$\epsilon_{P,1}(\theta) \to \frac{\|b\|}{1+\|b\|} < 1,$$
$$\epsilon_{D,1}(\theta) \to \frac{\|\Pi_K[\mathcal{A}^*y_0 - c]\|}{1+\|c\|} \le \frac{\|\mathcal{A}^*y_0 - c\|}{1+\|c\|} \le \frac{\|c\|}{1+\|c\|} < 1,$$

As a consequence, we can always choose an initial θ so as to enforce $\max\{\epsilon_{P,1}, \epsilon_{D,1}\}$ to be $\mathcal{O}(1)$. In fact, in our implementation we use the following procedure. Given a constant $\rho \ge 1$, we check whether $\max\{\epsilon_{P,1}(1), \epsilon_{D,1}(1)\} \le \rho$. If so, we set the initial θ to be 1, otherwise we successively divide the current value of θ by 2 until $\max\{\epsilon_{P,1}(\theta), \epsilon_{D,1}(\theta)\} \le \rho$ is satisfied, and use this value as the initial θ. The motivation behind this initial choice of θ is to guarantee that the initial primal and dual relative residuals $\epsilon_{P,k}$ and $\epsilon_{D,k}$ are not too large at the first iteration of Algorithm 1.

Even though, the convergence rate bounds of Theorem 3.3 are guaranteed for a fixed value of θ, we have used in our computational results the heuristic of changing θ every time a specified number $\bar{k}$ of iterations have been performed. The motivation for dynamically changing θ, is that our preliminary computational experiments have suggested us that the performance of the method is improved as $\epsilon_{P,k}$ and $\epsilon_{D,k}$ are of the same order of magnitude. More specifically, if θ_k denotes the dynamic value of θ at the kth iteration of the algorithm, we use the following rule for updating θ_k,

$$\theta_k = \begin{cases} \theta_{k-1}, & k \not\equiv 0 \bmod \bar{k} \text{ or } \gamma^{-1} \le \epsilon_{P,k-1}/\epsilon_{D,k-1} \le \gamma, \\ \theta_{k-1}\cdot\tau, & k \equiv 0 \bmod \bar{k} \text{ and } \epsilon_{P,k-1}/\epsilon_{D,k-1} > \gamma, \\ \theta_{k-1}/\tau, & k \equiv 0 \bmod \bar{k} \text{ and } \epsilon_{D,k-1}/\epsilon_{P,k-1} > \gamma, \end{cases} \qquad \forall k \ge 2,$$

for some pre-specified integer $\bar{k} \ge 1$, and scalars $\gamma > 1$ and $0 < \tau < 1$. In our computational experiments, we have used $\bar{k} = 5$, $\gamma = 1.5$ and $\tau = 0.9$. Note that this update rule is motivated by the last observation in Sect. 3. In summary, the update rule changes the value of θ at most a single time in the right direction, so as to balance the sizes of the primal and dual relative residuals based on the information provided by their values at the previous iteration. We should emphasize that convergence rate bounds for Algorithm 1 endowed with this updating rule are not available, but we have observed that this variant of Algorithm 1 performs extremely well.

In our computational experiments, we will refer to the variant of Algorithm 1 described above as the *dynamically scaled adaptive block-decomposition* (DSA-BD) method for solving (1). This variant was implemented for spaces $\mathcal{X}$ and $\mathcal{Y}$, and cone K given as in (4). Hence, our code is able to solve conic programming problems given in standard form (i.e., as in (1)) with n_u unrestricted scalar variables, n_l nonnegative scalar variables and an $n_s \times n_s$ positive semidefinite symmetric matrix variable. The

inner products (before scaling) used in $\mathcal{X}$ and $\mathcal{Y}$ are the standard ones, namely: the scalar inner product in $\mathcal{Y}$ and the following inner product in $\mathcal{X}$

$$\langle x, \tilde{x} \rangle := x_v^T \tilde{x}_v + X_s \bullet \tilde{X}_s,$$

for every $x = (x_v, X_s) \in \mathbb{R}^{n_u+n_l} \times \mathcal{S}^{n_s}$ and $\tilde{x} = (\tilde{x}_v, \tilde{X}_s) \in \mathbb{R}^{n_u+n_l} \times \mathcal{S}^{n_s}$, where $X \bullet Y := Tr(X^T Y)$ for every $X, Y \in \mathcal{S}^{n_s}$.

We present a computational benchmark of our algorithm (DSA-BD) compared to the semismooth Newton-CG augmented Lagrangian (SDPNAL) method in [22] and the boundary-point (BP) method in [9, 15]. We implemented the DSA-BD method in MATLAB using the SDPT3 data structures described in [20], but without exploiting any possible block sparsity on the semidefinite variable X_s. All the tests were made using a server with 2 Xeon X5460 processors at 3.16 GHz and 32 GB RAM.

Various large-scale SDP problems are solved to obtain this benchmark, ranging from purely random to SDP relaxations of combinatorial optimization problems such as the frequency assignment problem (FAP), the binary integer quadratic (BIQ) problem, the quadratic assignment problem (QAP) and the maximum stable set problem (Lovász θ-function and θ_+-function). In the following subsections, we describe in detail the problems included in our computational tests but before that, we make some general remarks about how the results are reported on the several tables given in the electronic supplementary material. In Tables 1 and 11, we compare our method to BP and SDPNAL methods while, in Tables 3, 5, 6, 9 and 13, we compare it against SDPNAL only due to the fact that the current version of the BP method available to us only accepts conic optimization SDP problems without nonnegative scalar variables. In some of these tables, we report computational results for the same problem using two different tolerances. They are listed in two different rows of the table to the right of the name and size of the instance. We mark the time and the residual for a method in red, and also with an asterisk (*), whenever the instance cannot be solved to the required accuracy, with the convention that the time and residual reported are the ones obtained at the last iteration of the method. Also, the time marked in blue in a row is the best one among the times listed in that row under the convention that when a method cannot solve the instance, the corresponding time is assumed to be ∞.

Observe that the final relative residuals obtained by BP and DSA-BD are very close to the desired accuracy when the latter is achieved. On the other hand, the ones obtained by SDPNAL can be noticeably smaller than the desired accuracy when the latter is achieved. This is due to the fact that SDPNAL is a second-order method and therefore it performs much fewer (and computationally more expensive) iterations than the other two methods. As a result, SDPNAL improves the relative residuals in a single iteration substantially more than the other two methods.

In Tables 2, 4, 7, 8, 10, 12 and 14, we report more detailed computational results obtained by our method DSA-BD. We do not report the violations to the conditions $\tilde{x} \in K$, $\tilde{z} \in K^*$, $\langle \tilde{x}, \tilde{z} \rangle = 0$, since they are satisfied up to machine precision at every iteration of the DSA-BD algorithm applied to all the instances in our benchmark.

Finally, we recall the following definition of a performance profile. For a given instance, a method A is said to be at most x times slower than method B, if the time taken by method A is at most x times the time taken by method B. A point (x, y) is in the performance profile curve of a method if it can solve exactly $(100y)$ % of all

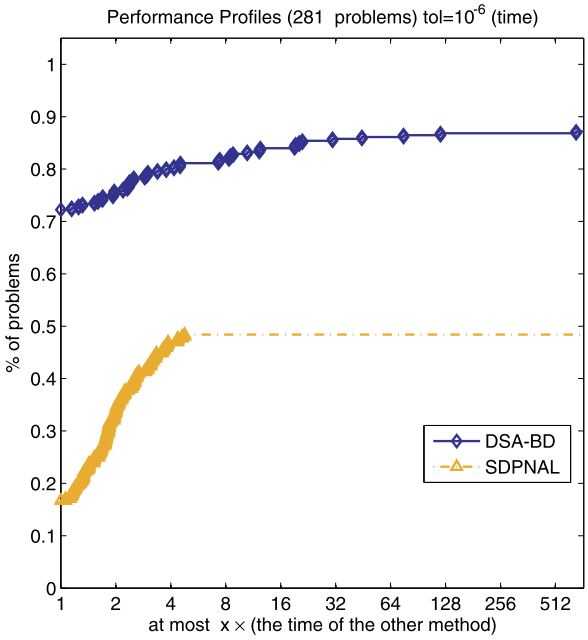

Fig. 1 Performance profiles of DSA-BD and SDPNAL for solving 281 conic semidefinite programming problems with accuracy $\bar{\epsilon} = 10^{-6}$

the tested instances x times slower than any other competing method. Figure 1 plots the performance profiles (see [6]) of DSA-BD and SDPNAL methods based on all instances used in our benchmark. Note that the curve for SDPNAL becomes flat for $x \geq 6$ at a y value equal to about 0.5. This is due to the fact that SDPNAL fails to solve about 50 % of the instances, although it is faster than DSA-BD on 18 % of the instances. Other performance profiles based on instances belonging to the same class of conic programming problems will be reported in the subsequent subsections.

4.1 Random SDPs

This subsection compares the performance of our method DSA-BD with that of BP and SDPNAL on a collection of random sparse SDP problems. These instances were also used in [9] to report the performance of BP introduced in [15].

Table 1 compares the three methods on a collection of random sparse SDP instances using the tolerance $\bar{\epsilon} = 10^{-6}$. Table 2 gives more detailed computational results on these instances obtained by our method DSA-BD, such as the objective values, number of iterations, and the primal and dual relative residuals. Figure 2 plots the performance profiles of the three methods based on these random sparse SDP instances only.

Note that DSA-BD finds a solution with an accuracy of at least 10^{-6} faster than BP and SDPNAL do in most of the random sparse SDP instances tested. In particular, DSA-BD is the fastest method on the larger instances.

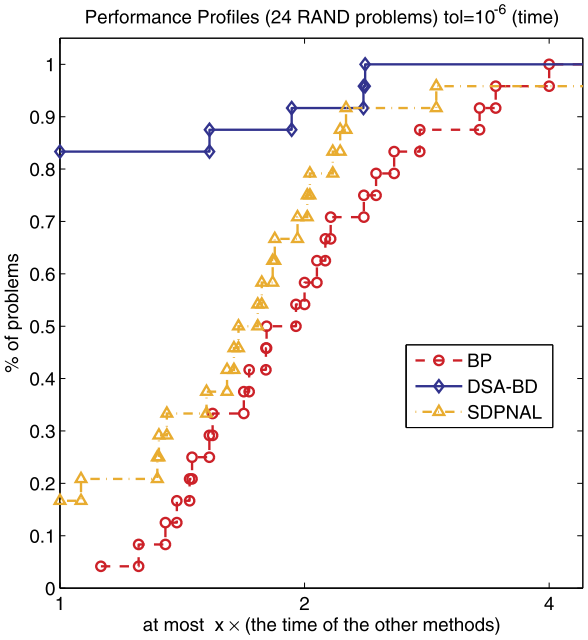

Fig. 2 Performance profiles of DSA-BD, SDPNAL and BP for solving 24 random SDP problems with accuracy $\bar{\epsilon} = 10^{-6}$

4.2 Frequency assignment problems

This subsection compares the performance of our method DSA-BD with that of SDPNAL on a collection of SDP relaxations of FAPs.

The SDP relaxation of the FAP can be described as follows (see for example Sect. 2.4 in [4]). Given a network represented by a graph G and an edge-weight matrix W, the frequency assignment problem on G can be formulated as a k-cut problem

$$\begin{aligned} \max \quad & \left[\left(\frac{k-1}{2k}\right)L(G,W) - \frac{1}{2}\mathrm{Diag}(We)\right] \bullet X \\ \text{s.t.} \quad & -E^{ij} \bullet X \leq 2/(k-1) \quad \forall (i,j) \\ & -E^{ij} \bullet X = 2/(k-1) \quad \forall (i,j) \in U \subseteq E \\ & \mathrm{diag}(X) = e, \quad X \succeq 0, \quad \mathrm{rank}(X) = k, \end{aligned}$$

where $k > 1$ is an integer, $L(G,W) := \mathrm{Diag}(We) - W$ is the Laplacian matrix, $E^{ij} = e_i e_j^T + e_j e_i^T$ with $e_i \in \mathbb{R}^n$ the vector with all zeros except in the ith position and $e \in \mathbb{R}^n$ is the vector with all ones. An SDP relaxation of the problem above is obtained by dropping the rank restriction and the inequality constraint for the non-edges to obtain the following formulation

$$\max \quad \left[\left(\frac{k-1}{2k}\right)L(G,W) - \frac{1}{2}\mathrm{Diag}(We)\right] \bullet X$$

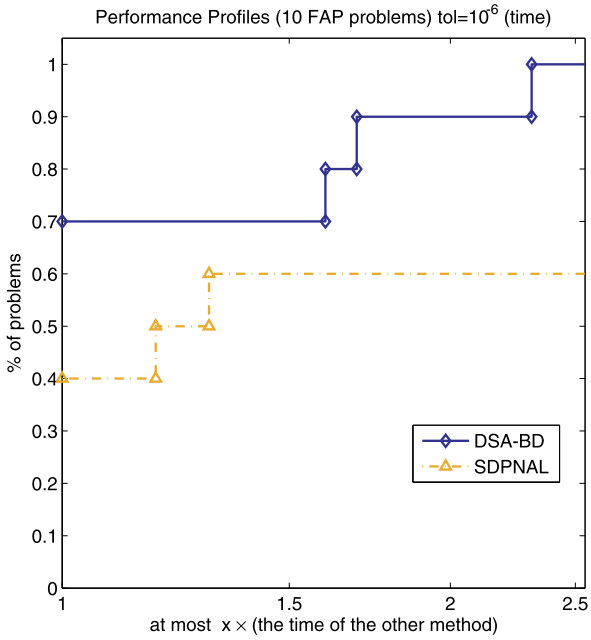

Fig. 3 Performance profiles of DSA-BD and SDPNAL for solving 10 SDP relaxations of FAPs with accuracy $\bar{\epsilon} = 10^{-6}$

$$\begin{aligned} \text{s.t.} \quad & -E^{ij} \bullet X \le 2/(k-1) \quad \forall (i,j) \in E \setminus U \\ & -E^{ij} \bullet X = 2/(k-1) \quad \forall (i,j) \in U \subseteq E \\ & \text{diag}(X) = e, \quad X \succeq 0. \end{aligned}$$

Table 3 compares the two methods on a collection of SDP relaxations of FAPs using the tolerances $\bar{\epsilon} = 10^{-5}, 10^{-6}$. In this table, computational results for each instance are reported in two rows, the first one for $\bar{\epsilon} = 10^{-5}$, and the second one for $\bar{\epsilon} = 10^{-6}$. Table 4 gives more detailed computational results on these instances obtained by our method DSA-BD, such as the objective values, number of iterations, and the primal and dual relative residuals. Figure 3 plots the performance profiles of both methods based on these SDP relaxations of FAPs.

Note that our method performs better than SDPNAL on large SDP relaxations of FAPs (i.e., `fap25` and `fap36`).

4.3 Binary integer quadratic problems

This subsection compares the performance of our method DSA-BD with that of SDPNAL on a collection of SDP relaxations of BIQ problems.

The SDP relaxation of the BIQ problem can be described as follows (see for example Sect. 7 in [22]). Given an $n \times n$ symmetric matrix Q the BIQ problem can be formulated as

$$\min\{x^T Q x : x \in \{0,1\}^n\}.$$

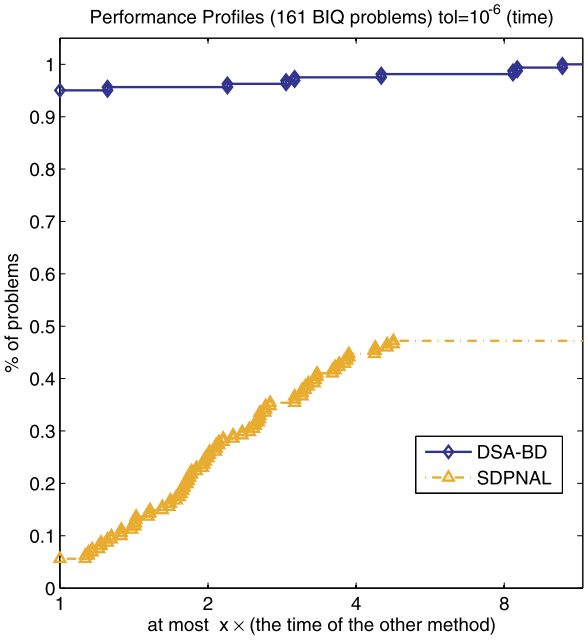

Fig. 4 Performance profiles of DSA-BD and SDPNAL for solving 161 SDP relaxations of BIQ problems with accuracy $\bar{\epsilon} = 10^{-6}$

By representing the binary set $\{0, 1\}^n$ as $\{x \in \mathbb{R}^n | x_i^2 - x_i = 0\}$, we obtain an SDP relaxation as follows

$$\begin{aligned} \min \quad & Q \bullet X \\ \text{s.t.} \quad & \operatorname{diag}(X) - x = 0, \quad \alpha = 1, \\ & \begin{bmatrix} X & x \\ x^T & \alpha \end{bmatrix} \geq 0, \quad X \geq 0, \ x \geq 0. \end{aligned}$$

Tables 5 and 6 compare the two methods on a collection of SDP relaxations of BIQ problems using the tolerance $\bar{\epsilon} = 10^{-6}$. Tables 7 and 8 give more detailed computational results on these instances obtained by our method DSA-BD, such as the objective values, number of iterations, and the primal and dual relative residuals. Figure 4 plots the performance profiles of both methods based on these SDP relaxations of BIQ problems.

Note that SDPNAL takes more time than DSA-BD to find a solution with an accuracy of at least 10^{-6} in almost all of the SDP relaxations of BIQ problems tested, and it fails to compute such a solution on more than half of these instances. On the other hand, our method DSA-BD was able to find a solution with an accuracy of at least 10^{-6} for all of the SDP relaxations of BIQ problems tested.

4.4 Quadratic assignment problems

This subsection compares the performance of our method DSA-BD with that of SDPNAL on a collection of SDP relaxations of QAPs.

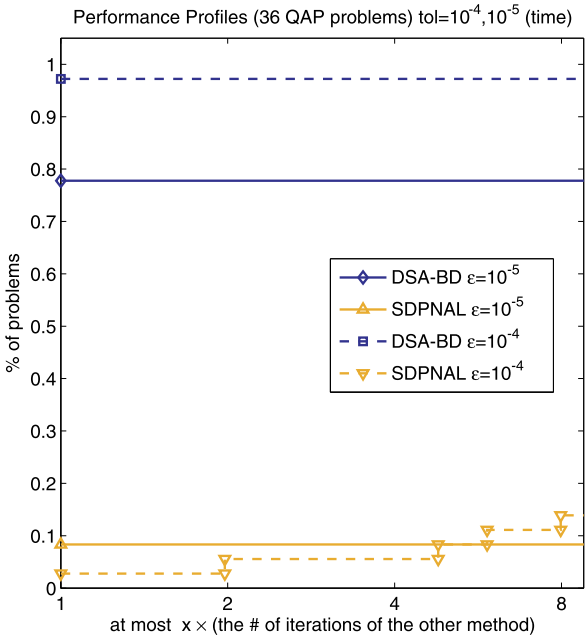

Fig. 5 Performance profiles of DSA-BD and SDPNAL for solving 36 SDP relaxations of QAPs with accuracies $\bar{\epsilon} = 10^{-4}, 10^{-5}$

Given the set Π of $n \times n$ permutation matrices and $A, B \in \mathbb{R}^{n\times n}$, the quadratic assignment problem can be formulated as

$$\min\{\langle X, AXB\rangle : X \in \Pi\}.$$

For a matrix $X = [x_1, \ldots, x_n] \in \mathbb{R}^{n\times n}$, we will identify it with the n^2-vector $x = (x_1; \ldots; x_n)$. For a matrix $Y \in \mathbb{R}^{n^2\times n^2}$, we let Y^{ij} be the $n \times n$ block corresponding to $x_i x_j^T$ in the matrix xx^T. In [14], it is shown that an SDP relaxation of the QAP is

$$\begin{aligned}
\max \quad & \langle B \otimes A, Y\rangle \\
\text{s.t.} \quad & \sum_{i=1}^{n} Y^{ii} = I, \quad \langle I, Y^{ij}\rangle = \delta_{ij} \ \forall 1 \le i \le j \le n, \\
& \langle E, Y^{ij}\rangle = 1, \quad \forall 1 \le i \le j \le n, \\
& Y \succeq 0, \quad Y \ge 0,
\end{aligned}$$

where $E \in \mathbb{R}^{n\times n}$ is the matrix of ones, and $\delta_{ij} = 1$ if $i = j$, and 0 otherwise.

Table 9 compares the two methods on a collection of SDP relaxations of QAPs using the tolerances $\bar{\epsilon} = 10^{-4}, 10^{-5}$. In this table, computational results for each instance are reported in two rows, the first one for $\bar{\epsilon} = 10^{-4}$, and the second one for $\bar{\epsilon} = 10^{-5}$. Table 10 gives more detailed computational results on these instances obtained by our method DSA-BD, such as the objective values, number of iterations, and the primal and dual relative residuals.

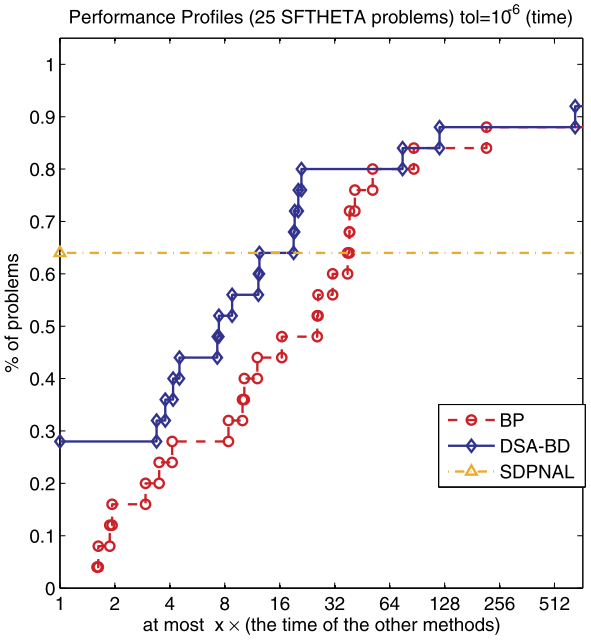

Fig. 6 Performance profiles of DSA-BD, SDPNAL and BP for solving 25 $\theta(G)$ problems with accuracy $\bar{\epsilon} = 10^{-6}$

Figure 5 plots the performance profiles of both methods based on these SDP relaxations of QAPs. Note that SDPNAL fails to find a solution with an accuracy of at least 10^{-4} for almost all of the SDP relaxations of QAPs tested. On the other hand, our method DSA-BD was able to find a solution with an accuracy of at least 10^{-5} for almost all of the SDP relaxations of QAPs tested. Observe also that a flat line in this figure means that the corresponding method is faster for some instances, but fails to solve the rest of them. For example, SDPNAL is the fastest method for solving ~8 % of the instances with an accuracy of $\bar{\epsilon} = 10^{-5}$, but fails to obtain a solution for the other ~92 % of the instances.

4.5 SDPs arising from relaxation of maximum stable set problems

This subsection compares the performance of our method DSA-BD with that of BP and SDPNAL on a collection of SDPs corresponding to θ-functions and θ_+-functions of graph stable set problems.

The SDPs for θ-functions and θ_+-functions of graph stable set problems can be described as follows. Given a graph G with n nodes and an edge set E, the SDP relaxations $\theta(G)$ and $\theta_+(G)$ of the maximum stable set problem are defined as

$$\theta(G) := \max\{C \bullet X,\ X_{ij} = 0,\ (i,j) \in E,\ I \bullet X = 1,\ X \succeq 0\},$$
$$\theta_+(G) := \max\{C \bullet X,\ X_{ij} = 0,\ (i,j) \in E,\ I \bullet X = 1,\ X \succeq 0,,\ X \geq 0\},$$

where $C = ee^T$, $X \in \mathcal{S}^n$ and $e \in \mathbb{R}^n$ is the vector with all ones.

Tables 11 and 13 compare the three methods on a collection of $\theta(G)$ and $\theta_+(G)$ problems using the tolerance $\bar{\epsilon} = 10^{-6}$. Tables 12 and 14 give more detailed com-

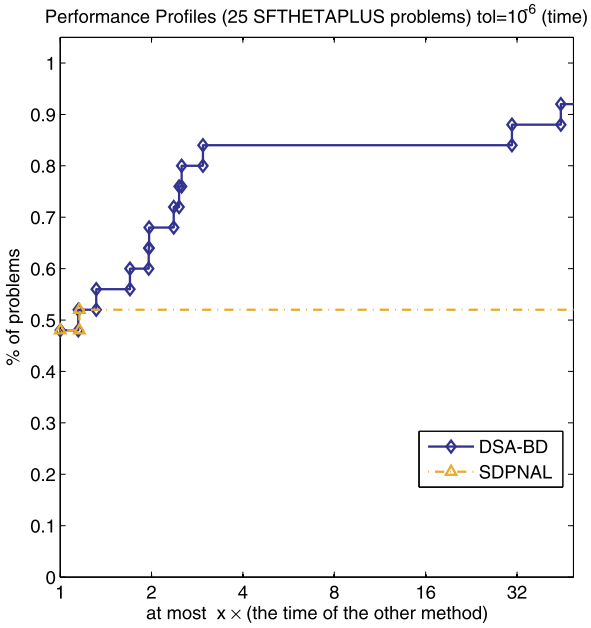

Fig. 7 Performance profiles of DSA-BD and SDPNAL for solving 25 $\theta_+(G)$ problems with accuracy $\bar{\epsilon} = 10^{-6}$

putational results on these instances obtained by our method DSA-BD, such as the objective values, number of iterations, and the primal and dual relative residuals. Figures 6 and 7 plot the performance profiles of the three methods based on these $\theta(G)$ and $\theta_+(G)$ problems.

Note that even though SDPNAL is faster than BP and DSA-BD on more than 60 % of the $\theta(G)$ instances, BP and DSA-BD are more robust, as they are able to solve almost all of the $\theta(G)$ instances to an accuracy of at least 10^{-6}, while SDPNAL fails to do so in more than 35 % of them. Also, BP takes more time than DSA-BD to find a solution with an accuracy of at least 10^{-6} in almost all of the $\theta(G)$ instances tested.

Note also that our method DSA-BD was able to find a solution with an accuracy of at least 10^{-6} for almost all of the $\theta_+(G)$ instances tested, while SDPNAL fails to do so for almost half of them.

Appendix: Technical results

Lemma 5.1 Theorem 2.2 in [1] *Given a self adjoint positive definite linear mapping $\mathcal{U} : \mathcal{Y} \to \mathcal{Y}$ and a random vector $y \in \mathcal{Y}$ uniformly distributed on a ball, we have that*

$$\mathbb{E}\left(\frac{\|\mathcal{U}^{1/2}y\|^2}{\|\mathcal{U}\|\,\|y\|^2}\right) = \frac{1}{\sigma_m}\frac{\sum\sigma_i}{m} \leq 1$$

where $\sigma_1 \leq \cdots \leq \sigma_m$ are the eigenvalues values of $\mathcal{U}$.

References

1. Bottcher, A., Grudsky, S.: The norm of the product of a large matrix and a random vector. Electron. J. Probab. **8**, 7 (2003), 29 p., electronic only. URL: http://eudml.org/doc/124759
2. Burachik, R.S.S., Svaiter, B.F.: Maximal monotone operators, convex functions and a special family of enlargements. Set-Valued Anal. **10**, 297–316 (2002). doi:10.1023/A:1020639314056
3. Burachik, R.S., Iusem, A.N., Svaiter, B.F.: Enlargement of monotone operators with applications to variational inequalities. Set-Valued Anal. **5**, 159–180 (1997). doi:10.1023/A:1008615624787
4. Burer, S., Monteiro, R.D.C., Zhang, Y.: A computational study of a gradient-based log-barrier algorithm for a class of large-scale SDPs. Math. Program. **95**, 359–379 (2003). doi:10.1007/s10107-002-0353-7
5. Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with applications to imaging. J. Math. Imaging Vis. **40**, 120–145 (2011). doi:10.1007/s10851-010-0251-1
6. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles (January 2002). doi:10.1007/s101070100263
7. Gabay, D., Mercier, B.: A dual algorithm for the solution of nonlinear variational problems via finite element approximation. Comput. Math. Appl. **2**(1), 17–40 (1976). doi:10.1016/0898-1221(76)90003-1. URL: http://www.sciencedirect.com/science/article/pii/0898122176900031
8. Glowinski, R., Marrocco, A.: Sur l'approximation par éléments finis et la résolution par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. RAIRO. Anal. Numér. **2**, 41–76 (1975)
9. Malick, J., Povh, J., Rendl, F., Wiegele, A.: Regularization methods for semidefinite programming. SIAM J. Optim. **20**(1), 336–356 (2009). doi:10.1137/070704575
10. Monteiro, R.D.C., Svaiter, B.F.: On the complexity of the hybrid proximal extragradient method for the iterates and the ergodic mean. SIAM J. Optim. **20**(6), 2755–2787 (2010). doi:10.1137/090753127. URL: http://link.aip.org/link/?SJE/20/2755/1
11. Monteiro, R.D.C., Svaiter, B.F.: Complexity of variants of Tseng's modified F-B splitting and Korpelevich's methods for hemivariational inequalities with applications to saddle-point and convex optimization problems. SIAM J. Optim. **21**(4), 1688–1720 (2011). doi:10.1137/100801652. URL: http://link.aip.org/link/?SJE/21/1688/1
12. Monteiro, R., Svaiter, B.: Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers. SIAM J. Optim. **23**(1), 475–507 (2013). doi:10.1137/110849468
13. Monteiro, R.D.C., Ortiz, C., Svaiter, B.F.: A first-order block-decomposition method for solving two-easy-block structured semidefinite programs. Optimization-online preprint 3544, pp. 1–33 (2012). URL: http://www.optimization-online.org/DB_HTML/2012/07/3544.html
14. Povh, J., Rendl, F.: Copositive and semidefinite relaxations of the quadratic assignment problem. Discrete Optim. **6**(3), 231–241 (2009). doi:10.1016/j.disopt.2009.01.002. URL: http://www.sciencedirect.com/science/article/pii/S1572528609000036
15. Povh, J., Rendl, F., Wiegele, A.: A boundary point method to solve semidefinite programs. Computing **78**, 277–286 (2006). doi:10.1007/s00607-006-0182-2
16. Rockafellar, R.T.: On the maximal monotonicity of subdifferential mappings. Pac. J. Math. **33**, 209–216 (1970)
17. Solodov, M.V., Svaiter, B.F.: A hybrid approximate extragradient—proximal point algorithm using the enlargement of a maximal monotone operator. Set-Valued Anal. **7**(4), 323–345 (1999)
18. Solodov, M.V., Svaiter, B.F.: A hybrid projection-proximal point algorithm. J. Convex Anal. **6**(1), 59–70 (1999)
19. Svaiter, B.F.: A family of enlargements of maximal monotone operators. Set-Valued Anal. **8**, 311–328 (2000). doi:10.1023/A:1026555124541
20. Toh, K.C., Todd, M.J., Tütüncü, R.H.: Sdpt3—a Matlab software package for semidefinite programming. Optim. Methods Softw. **11**, 545–581 (1999)
21. Wen, Z., Goldfarb, D., Yin, W.: Alternating direction augmented Lagrangian methods for semidefinite programming. Math. Program. Comput. **2**, 203–230 (2010). doi:10.1007/s12532-010-0017-1
22. Zhao, X.-Y., Sun, D., Toh, K.-C.: A Newton-CG augmented Lagrangian method for semidefinite programming. SIAM J. Optim. **20**(4), 1737–1765 (2010). doi:10.1137/080718206. URL: http://link.aip.org/link/?SJE/20/1737/1