Math. Prog. Comp. manuscript No.
(will be inserted by the editor)

An inexact block-decomposition method for extra large-scale conic semidefinite programming

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

Received: date / Accepted: date

Abstract In this paper, we present an inexact block-decomposition (BD) first-order method for solving standard form conic semidefinite programming (SDP) which avoids computations of exact projections onto the manifold defined by the affine constraints and, as a result, is able to handle extra large SDP instances. The method is based on a two-block reformulation of the optimality conditions where the first block corresponds to the complementarity conditions and the second one corresponds to the manifolds consisting of both the primal and dual affine feasibility constraints. While the proximal subproblem corresponding to the first block is solved exactly, the one corresponding to the second block is solved inexactly in order to avoid finding the exact solution of the underlying augmented primal-dual linear system. The error condition required by the BD method naturally suggests the use of a relative error condition in the context of the latter augmented primal-dual linear system. Our implementation uses the conjugate gradient (CG) method applied

The work of R. D. C. Monteiro was partially supported by NSF Grants 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. 303583/2008-8, 302962/2011-5, 480101/2008-6, 474944/2010-7, FAPERJ grants E-26/102.821/2008, E-26/102.940/2011.

R. D. C. Monteiro
School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, 30332-0205
E-mail: monteiro@isye.gatech.edu

C. Ortiz
School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, 30332-0205
Tel.: +1-678-644-2561
E-mail: camiort@gatech.edu

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

to a reduced positive definite dual linear system to obtain inexact solutions of the augmented linear system. In addition, we implemented the proposed BD method with the following refinements: an adaptive choice of stepsize for performing an extragradient step; and a new dynamic scaling scheme that uses two scaling factors to balance three inclusions comprising the optimality conditions of the conic SDP. Finally, we present computational results showing the efficiency of our method for solving various extra large SDP instances, several of which cannot be solved by other existing methods, including some with at least two million constraints and/or fifty million non-zero coefficients in the affine constraints.

Keywords complexity · proximal · extragradient · block-decomposition · large-scale optimization · conic optimization · semidefinite programing

Mathematics Subject Classification (2000) MSC 49M27 · MSC 49M37 · MSC 90C06 · MSC 90C22 · MSC 90C30 · MSC 90C35 · MSC 90C90

1 Introduction

Let $\mathbb{R}$ denote the set of real numbers, $\mathbb{R}^n$ denote the n-dimensional Euclidean space, $\mathbb{R}^n_+$ denote the cone of nonnegative vectors in $\mathbb{R}^n$, $\mathcal{S}^n$ denote the set of all $n \times n$ symmetric matrices and $\mathcal{S}^n_+$ denote the cone of $n \times n$ symmetric positive semidefinite matrices. Throughout this paper, we 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

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

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

$$\max_{(z,y) \in \mathcal{X} \times \mathcal{Y}} \{ \langle b, y \rangle : \mathcal{A}^* y + z = c, \ z \in K^* \}, \tag{2}$$

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

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

Several papers [2,7,9,8,17,18] in the literature discuss methods/codes for solving large-scale conic semidefinite programming problems (SDP), i.e., special cases of (1) in which

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

Presently, the most efficient methods/codes for solving large-scale conic SDP problems are the first-order projection-type discussed in [7,9,8,17,18] (see also [14] for a slight variant of [7]).

More specifically, augmented Lagrangian approaches have been proposed for the dual formulation of (1) 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 [7,14], a boundary point (BP) method for solving (1) is proposed which can be viewed as variants of the alternating direction method of multipliers introduced in [5,6] applied to the dual formulation (2). In [18], an inexact augmented Lagrangian method is proposed which solves a reformulation of the augmented Lagrangian subproblem via a semismooth Newton approach combined with the conjugate gradient method. Using the theory developed in [11], an implementation of a first-order block-decomposition (BD) algorithm, based on the hybrid proximal extragradient (HPE) method [16], for solving standard form conic SDP problems is discussed in [9], and numerical results are presented showing that it generally outperforms the methods of [7, 18]. In [17], an efficient variant of the BP method is discussed and numerical results are presented showing its impressive ability to solve important classes of large-scale graph-related SDP problems. In [8], another BD method, based on the theory developed in [11], for minimizing the sum of a convex differentiable function with Lipschitz continuous gradient, and two other proper closed convex (possibly, nonsmooth) functions with easily computable resolvents is discussed. Numerical results are presented in [8] showing that the latter BD method generally outperforms the variant of the BP method in [17], as well as the methods of [7,9,18], in a benchmark that included the same classes of large-scale graph-related SDP problems tested in [17]. It should be observed though that the implementations in [17] and [8], are very specific in the sense that they both take advantage of each SDP problem class structure so as to keep the number of variables and/or constraints as small as possible. This contrasts with the codes described in [7], [9] and [18], which always introduce additional variables and/or constraints into the original SDP formulation to bring it into the required standard form input.

Moreover, all the methods in the papers [7,9,8,17,18] assume that the following operations are easily computable (at least once per iteration):

O.1) Evaluation of the linear operator: For given points $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, compute the points $\mathcal{A}x$ and $\mathcal{A}^*y$.

O.2) Projection onto the cone: For a given point $x \in \mathcal{X}$, compute $x_K = \arg\min\{\|x - \tilde{x}\| : \tilde{x} \in K\}$.

O.3) Projection onto the manifold: For a given point $x \in \mathcal{X}$, compute $x_{\mathcal{M}} = \arg\min\{\|x - \tilde{x}\| : \tilde{x} \in \mathcal{M}\}$ where $\mathcal{M} = \{x \in \mathcal{X} : \mathcal{A}x = b\}$. If $\mathcal{A}$ is surjective, it is easy to see that $x_{\mathcal{M}} = x + \mathcal{A}^*(\mathcal{A}\mathcal{A}^*)^{-1}(b - \mathcal{A}x)$. Hence, the cost involved in this operation depends on the ability to solve linear systems of the form $\mathcal{A}\mathcal{A}^*y = \hat{b}$ where $\hat{b} \in \mathcal{Y}$.

The focus of this paper is on the development of algorithms for solving problems as in (1) where carrying out O.3 requires significantly more time and/or storage than both O.1 and O.2. More specifically, we present a BD method based on the BD-HPE framework of [11] that instead of exactly solving linear systems of the form $\mathcal{A}\mathcal{A}^*y = \hat{b}$, computes inexact solutions of augmented primal-dual linear systems satisfying a certain relative error condition. The latter relative error condition naturally appears as part of the HPE error condition which in turn guarantees the overall convergence of the BD method. Our implementation of the BD method obtains the aforementioned inexact solutions of the augmented linear systems by applying the conjugate gradient (CG) method to linear systems of the form $(\mathcal{I} + \alpha\mathcal{A}\mathcal{A}^*)y = \hat{b}$, where $\mathcal{I}$ is the identity operator and α is a positive scalar. Moreover, the BD method presented contains two important ingredients from a computational point of view that are based on the ideas introduced in [9] and [8], namely: an adaptive choice of stepsize for performing the extragradient step; and the use of two scaling factors that change dynamically to balance three blocks of inclusions that comprise the optimality conditions for (1). This latter ingredient generalizes the dynamic scaling ideas used in [8] by using two scaling parameters instead of one as in [8]. Finally, we present numerical results showing the ability of the proposed BD method to solve various conic SDP instances of the form (1) and (4) for which the operation of projecting a point onto the manifold $\mathcal{M}$ (see O.3) is prohibitively expensive, and as a result cannot be handled by the methods in [7,9,8,17,18]. In these numerical results, we also show that our method substantially outperforms the latest implementation (see [1]) of SDPLR introduced in [2,3]. SDPLR is a first-order augmented Lagrangian method applied to a nonlinear reformulation of the original problem (1) which restricts the $n_s \times n_s$ symmetric matrix component of the variable x to a low-rank matrix. Even though there are other first-order methods that avoid projecting onto the manifold $\mathcal{M}$ (see for example the BD variant in [9] with $\mathcal{U} = \mathcal{I}$), to the best of our knowledge, SDPLR is computationally the most efficient one.

Our paper is organized as follows. Section 2 reviews an adaptive block-decomposition HPE framework in the context of a monotone inclusion problem consisting of the sum of a continuous monotone map and a separable two-block subdifferential operator which is a special case of the one presented in [9] and [8]. Section 3 presents an inexact first-order instance of this framework, and corresponding iteration-complexity results, for solving the conic programming problem (1) which avoids the operation of projecting a point onto the manifold $\mathcal{M}$ (see O.3). Section 4 describes a practical variant of the BD method of Section 3 which incorporates the two ingredients described above and the use of the CG method to inexactly solve the augmented linear systems. Section 5 presents numerical results comparing SDPLR with the BD variant studied in this paper on a collection of extra large-scale conic SDP instances. Finally, Section 6 presents some final remarks.

1.1 Notation

Let a closed convex set $C \subset \mathcal{X}$ be given. The *projection* operator $\Pi_C : \mathcal{X} \to C$ onto C is defined by

$$\Pi_C(z) = \arg\min_{\tilde{x} \in C} \{\|x - \tilde{x}\|\} \qquad \forall x \in \mathcal{X},$$

the *indicator function* of C is the function $\delta_C : \mathcal{X} \to \bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$ defined as

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

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

$$N_C(x) = \begin{cases} \emptyset, & x \notin C, \\ \{w \in \mathcal{X} : \langle \tilde{x} - x, w \rangle \leq 0, \ \forall \tilde{x} \in C\}, & x \in C. \end{cases} \tag{5}$$

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

$$\|\mathcal{A}\| := \sup_{x \in \mathcal{X}} \{\|\mathcal{A}x\| : \|x\| \leq 1\},$$

and we denote by $\lambda_{\max}(\mathcal{A})$ and $\lambda_{\min}(\mathcal{A})$ the maximum and minimum eigenvalues of $\mathcal{A}$, respectively. Moreover, if $\mathcal{A}$ is invertible, the *condition number* $\kappa(\mathcal{A})$ of $\mathcal{A}$ is defined as

$$\kappa(\mathcal{A}) := \|\mathcal{A}\|\|\mathcal{A}^{-1}\|.$$

If $\mathcal{A}$ is identified with a matrix, $\text{nnz}(\mathcal{A})$ denotes the number of non-zeros of $\mathcal{A}$.

2 An adaptive block-decomposition HPE framework

This section is divided into two subsections. The first one reviews some basic definitions and facts about subdifferentials of functions and monotone operators. The second one reviews an adaptive block-decomposition HPE (A-BD-HPE) framework presented in [9] and [8], which is an extension of the BD-HPE framework introduced in [11]. To simplify the presentation, the latter framework is presented in the simpler context of a monotone inclusion problem consisting of the sum of a continuous monotone map and a separable two-block subdifferential operator (instead of a separable two-block maximal monotone operator).

2.1 The subdifferential and monotone operators

In this section, we review some properties of the subdifferential of a convex function and the monotone operator.

Let $\mathcal{Z}$ denote a finite dimensional inner product space with inner product and associated norm denoted by $\langle \cdot, \cdot \rangle_{\mathcal{Z}}$ and $\|\cdot\|_{\mathcal{Z}}$. A point-to-set operator $T : \mathcal{Z} \rightrightarrows \mathcal{Z}$ is a relation $T \subseteq \mathcal{Z} \times \mathcal{Z}$ and

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

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

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

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

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

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

$$\langle v - \tilde{v}, z - \tilde{z}\rangle_{\mathcal{Z}} \geq 0 \qquad \forall (z,v), (\tilde{z},\tilde{v}) \in Gr(T),$$

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 $S = T$. We now state a useful property of maximal monotone operators that will be needed in our presentation.

Proposition 2.1. *If $T : \mathcal{Z} \rightrightarrows \mathcal{Z}$ is maximal monotone and $F : Dom(F) \to \mathcal{Z}$ is a map such that $Dom F \supset cl(Dom T)$ and F is monotone and continuous on $cl(Dom T)$, then $F + T$ is maximal monotone.*

Proof. See proof of Proposition A.1 in [10]. □

The subdifferential of a function $f : \mathcal{Z} \to [-\infty, +\infty]$ is the operator $\partial f : \mathcal{Z} \rightrightarrows \mathcal{Z}$ defined as

$$\partial f(z) = \{v \mid f(\tilde{z}) \geq f(z) + \langle \tilde{z} - z, v\rangle_{\mathcal{Z}}\,,\ \forall \tilde{z} \in \mathcal{Z}\} \qquad \forall z \in \mathcal{Z}. \tag{6}$$

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 [15, Theorem A]. Clearly, the normal cone operator N_K of a closed convex cone K can be expressed in terms of its indicator function as $N_K = \partial \delta_K$.

2.2 The A-BD-HPE framework

In this subsection, we review the A-BD-HPE framework with adaptive stepsize for solving a special type of monotone inclusion problem consisting of the sum of a continuous monotone map and a separable two-block subdifferential operator.

Let $\mathcal{Z}$ and $\mathcal{W}$ be finite dimensional inner product spaces with associated inner products denoted by $\langle \cdot, \cdot\rangle_{\mathcal{Z}}$ and $\langle \cdot, \cdot\rangle_{\mathcal{W}}$, respectively, and associated norms denoted by $\|\cdot\|_{\mathcal{Z}}$ and $\|\cdot\|_{\mathcal{W}}$, respectively. We endow the product space $\mathcal{Z} \times \mathcal{W}$ with the canonical inner product $\langle \cdot, \cdot\rangle_{\mathcal{Z}\times\mathcal{W}}$ and associated canonical norm $\|\cdot\|_{\mathcal{Z}\times\mathcal{W}}$ defined as

$$\langle (z,w), (z',w')\rangle_{\mathcal{Z}\times\mathcal{W}} := \langle z, z'\rangle_{\mathcal{Z}} + \langle w, w'\rangle_{\mathcal{W}},\ \|(z,w)\|_{\mathcal{Z}\times\mathcal{W}} := \sqrt{\langle (z,w), (z.w)\rangle_{\mathcal{Z}\times\mathcal{W}}}, \tag{7}$$

for all $(z,w), (z',w') \in \mathcal{Z} \times \mathcal{W}$.

Our problem of interest in this section is the monotone inclusion problem of finding $(z,w) \in \mathcal{Z} \times \mathcal{W}$ such that

$$(0,0) \in [F + \partial h_1 \otimes \partial h_2](z,w), \tag{8}$$

where

$$F(z,w) = (F_1(z,w), F_2(z,w)) \in \mathcal{Z} \times \mathcal{W},$$
$$(\partial h_1 \otimes \partial h_2)(z,w) = \partial h_1(z) \times \partial h_2(w) \subseteq \mathcal{Z} \times \mathcal{W},$$

and the following conditions are assumed:

A.1 $h_1 : \mathcal{Z} \to \bar{\mathbb{R}}$ and $h_2 : \mathcal{W} \to \bar{\mathbb{R}}$ are proper lower semi-continuous convex functions;
A.2 $F : \mathrm{Dom}\, F \subseteq \mathcal{Z} \times \mathcal{W} \to \mathcal{Z} \times \mathcal{W}$ is a continuous map such that $\mathrm{Dom} F \supset \mathrm{cl}(\mathrm{Dom}\, \partial h_1) \times \mathcal{W}$;
A.3 F is monotone on $\mathrm{cl}(\mathrm{Dom}\, \partial h_1) \times \mathrm{cl}(\mathrm{Dom}\, \partial h_2)$;
A.4 there exists $L > 0$ such that

$$\|F_1(z,w') - F_1(z,w)\|_{\mathcal{Z}} \le L\|w' - w\|_{\mathcal{W}} \qquad \forall z \in \mathrm{Dom}\, \partial h_1,\ \forall w, w' \in \mathcal{W}. \tag{9}$$

Since $\partial h_1 \otimes \partial h_2$ is the subdifferential of the proper lower semi-continuous convex function $(z,w) \mapsto h_1(z) + h_2(w)$, it follows from [15, Theorem A] that $\partial h_1 \otimes \partial h_2$ is maximal monotone. Moreover, in view of Proposition 2.1, it follows that $F + \partial h_1 \otimes \partial h_2$ is maximal monotone. Note that problem (8) is equivalent to

$$0 \in F_1(z,w) + \partial h_1(z), \tag{10a}$$
$$0 \in F_2(z,w) + \partial h_2(w). \tag{10b}$$

For the purpose of motivating and analyzing the main algorithm (see Algorithm 1) of this paper for solving problem (1), we state a rather specialized version of the A-BD-HPE framework presented in [8] where the sequences $\{\varepsilon'_k\}$, $\{\varepsilon''_k\}$ and $\{\tilde{\lambda}_k\}$, and the maximal monotone operators C and D in [8] are set to $\varepsilon'_k = \varepsilon''_k = 0$, $\tilde{\lambda}_k = \tilde{\lambda}$, $C = \partial h_1$, and $D = \partial h_2$, and the scalars σ_x and $\tilde{\sigma}_x$ satisfy $\sigma_x = \tilde{\sigma}_x$.

Special-BD Framework: Specialized adaptive block-decomposition HPE (A-BD-HPE) framework

0) Let $(z^0,w^0) \in \mathcal{Z}\times\mathcal{W}$, $\sigma\in(0,1]$, $\sigma_z,\sigma_w\in[0,1)$ and $\tilde\lambda>0$ be given such that

$$\lambda_{\max}\left(\begin{bmatrix}\sigma_z^2 & \tilde\lambda\sigma_z L\\ \tilde\lambda\sigma_z L & \sigma_w^2+\tilde\lambda^2L^2\end{bmatrix}\right)^{1/2}\le\sigma \tag{11}$$

and set $k=1$;

1) compute $\tilde z^k, h_1^k\in\mathcal{Z}$ such that

$$h_1^k\in\partial h_1(\tilde z^k),\qquad \|\tilde\lambda[F_1(\tilde z^k,w^{k-1})+h_1^k]+\tilde z^k-z^{k-1}\|_{\mathcal{Z}}\le\sigma_z\|\tilde z^k-z^{k-1}\|_{\mathcal{Z}}, \tag{12}$$

2) compute $\tilde w^k, h_2^k\in\mathcal{W}$ such that

$$h_2^k\in\partial h_2(\tilde w^k),\qquad \|\tilde\lambda[F_2(\tilde z^k,\tilde w^k)+h_2^k]+\tilde w^k-w^{k-1}\|_{\mathcal{W}}\le\sigma_w\|\tilde w^k-w^{k-1}\|_{\mathcal{W}}; \tag{13}$$

3) choose λ_k as the largest $\lambda>0$ such that

$$\left\|\lambda\begin{pmatrix}F_1(\tilde z^k,\tilde w^k)+h_1^k\\ F_2(\tilde z^k,\tilde w^k)+h_2^k\end{pmatrix}+\begin{pmatrix}\tilde z^k\\ \tilde w^k\end{pmatrix}-\begin{pmatrix}z^{k-1}\\ w^{k-1}\end{pmatrix}\right\|_{\mathcal{Z}\times\mathcal{W}}\le\sigma\left\|\begin{pmatrix}\tilde z^k\\ \tilde w^k\end{pmatrix}-\begin{pmatrix}z^{k-1}\\ w^{k-1}\end{pmatrix}\right\|_{\mathcal{Z}\times\mathcal{W}}; \tag{14}$$

4) set

$$(z^k,w^k)=(z^{k-1},w^{k-1})-\lambda_k[F(\tilde z^k,\tilde w^k)+(h_1^k,h_2^k)], \tag{15}$$

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

The following result follows from Proposition 2.2 of [9] which in turn is an immediate consequence of Proposition 3.1 of [11].

Proposition 2.2. *Consider the sequence $\{\lambda_k\}$ generated by the Special-BD framework. Then, for every $k\in\mathbb{N}$, $\lambda=\tilde\lambda$ satisfies* (14). *As a consequence $\lambda_k\ge\tilde\lambda$.*

The following point-wise convergence result for the Special-BD framework follows from Theorem 3.2 of [11] and Theorem 2.3 of [9].

Theorem 2.3. *Consider the sequences $\{(\tilde z^k,\tilde w^k)\}, \{(h_1^k,h_2^k)\}$ and $\{\lambda_k\}$ generated by the Special-BD framework under the assumption that $\sigma<1$, and let d_0 denote the distance of the initial point $(z^0,w^0)\in\mathcal{Z}\times\mathcal{W}$ to the solution set of* (8) *with respect to $\|(\cdot,\cdot)\|_{\mathcal{Z}\times\mathcal{W}}$. Then, for every $k\in\mathbb{N}$, there exists $i\le k$ such that*

$$\|F(\tilde z^i,\tilde w^i)+(h_1^i,h_2^i)\|_{\mathcal{Z}\times\mathcal{W}}\le d_0\sqrt{\frac{1+\sigma}{1-\sigma}\left(\frac{1}{\lambda_i\sum_{j=1}^k\lambda_j}\right)}\le\sqrt{\frac{1+\sigma}{1-\sigma}}\frac{d_0}{\sqrt{k}\tilde\lambda}. \tag{16}$$

Note that the point-wise iteration-complexity bound in Theorem 2.3 is $\mathcal{O}(1/\sqrt{k})$. Theorem 2.4 of [9] derives an $\mathcal{O}(1/k)$ iteration-complexity ergodic bound for the Special-BD framework as an immediate consequence of Theorem 3.3 of [11].

3 An inexact scaled BD algorithm for conic programming

In this section, we introduce an inexact instance of the Special-BD framework applied to (1) which avoids the operation of projecting a point onto the manifold $\mathcal{M}$ (see O.3 in Section 1), and defines $\langle\cdot,\cdot\rangle_{\mathcal{Z}}$ and $\langle\cdot,\cdot\rangle_{\mathcal{W}}$ as scaled inner products constructed by means of the original inner products $\langle\cdot,\cdot\rangle$ of $\mathcal{X}$ and $\mathcal{Y}$. (Recall from Section 1 that the original inner products in $\mathcal{X}$ and $\mathcal{Y}$ used in (1) are both being denoted by $\langle\cdot,\cdot\rangle$.)

We consider problem (1) with the following assumptions:

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

$$0\in c+N_{\mathcal{M}}(x)+N_K(x), \tag{17}$$

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

We now make a few observations about the above assumptions. First, assumption B.2 is equivalent to the existence of $y^*\in\mathcal{Y}$ and $z^*\in\mathcal{X}$ such that the triple (z^*,y^*,x^*) satisfies

$$0\in x^*+N_{K^*}\left(z^*\right)=x^*+\partial\delta_{K^*}\left(z^*\right), \tag{18a}$$
$$0=\mathcal{A}x^*-b, \tag{18b}$$
$$0=c-\mathcal{A}^*y^*-z^*. \tag{18c}$$

Second, it is well-known that a triple (z^*,y^*,x^*) satisfies (18) if and only if x^* is an optimal solution of (1), the pair (z^*,y^*) is an optimal solution of (2) and duality gap between (1) and (2) is zero, i.e., $\langle c,x^*\rangle=\langle b,y^*\rangle$.

Our main goal in this section is to present an instance of the Special-BD framework which approximately solves (18) and does not require computation of exact projections onto the manifold $\mathcal{M}$ (see O.3 in Section 1). Instead of dealing directly with (18), it is more efficient from a computational point of view to consider its equivalent scaled reformulation

$$0\in\theta\left(x^*+N_{K^*}(z^*)\right)=\theta x^*+N_{K^*}(z^*), \tag{19a}$$
$$0=\mathcal{A}x^*-b, \tag{19b}$$
$$0=\xi(c-\mathcal{A}^*y^*-z^*), \tag{19c}$$

where θ and ξ are positive scalars.

We will now show that (19) can be viewed as a special instance of (10) which satisfies assumptions A.1–A.4. Indeed, let

$$\mathcal{Z}=\mathcal{X},\qquad \mathcal{W}=\mathcal{Y}\times\mathcal{X}, \tag{20}$$

and define the inner products as

$$\langle\cdot,\cdot\rangle_{\mathcal{Z}}:=\theta^{-1}\langle\cdot,\cdot\rangle,\quad \langle(y,x),(y',x')\rangle_{\mathcal{W}}:=\langle y,y'\rangle+\xi^{-1}\langle x,x'\rangle \tag{21}$$
$$\forall x,x'\in\mathcal{X},\ \forall y,y'\in\mathcal{Y},$$

and the functions F, h_1 and h_2 as

$$F(z,y,x) = \begin{bmatrix} \theta x \\ \mathcal{A}x - b \\ \xi(c - \mathcal{A}^* y - z) \end{bmatrix}, \quad h_1(z) = \delta_{K^*}(z), \quad h_2(y,x) = (0,0), \quad (22)$$

for all $(z,y,x) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{X}$.

The following proposition can be easily shown.

Theorem 3.1. *The inclusion problem* (19) *is equivalent to the inclusion problem* (10) *where the spaces $\mathcal{Z}$ and $\mathcal{W}$, the inner products $\langle\cdot,\cdot\rangle_{\mathcal{Z}}$ and $\langle\cdot,\cdot\rangle_{\mathcal{W}}$, and the functions F, h_1 and h_2 are defined as in* (20), (21) *and* (22). *Moreover, assumptions A.1–A.4 of Subsection 2.2 hold with $L = \sqrt{\theta\xi}$, and, as a consequence,* (19) *is maximal monotone with respect to $\langle\cdot,\cdot\rangle_{\mathcal{Z}\times\mathcal{W}}$* (see (7)).

In view of Proposition 3.1, from now on we consider the context in which (19) is viewed as a special case of (10) with $\mathcal{Z}$, $\mathcal{W}$, $\langle\cdot,\cdot\rangle_{\mathcal{Z}}$, $\langle\cdot,\cdot\rangle_{\mathcal{W}}$, F, h_1 and h_2 given by (20), (21) and (22). In what follows, we present an instance of the Special-BD framework in this particular context which solves the first block (12) exactly (i.e., with $\sigma_z = 0$), and solves the second block (13) inexactly (i.e., with $\sigma_w > 0$) with the help of a linear solver.

More specifically, let $\{(z^k, w^k)\}$, $\{(\tilde{z}^k, \tilde{w}^k)\}$ and $\{(h_1^k, h_2^k)\}$ denote the sequences as in the Special-BD framework specialized to the above context with $\sigma_z = 0$. For every $k \in \mathbb{N}$, in view of (20), we have that $z^k, \tilde{z}^k \in \mathcal{X}$, and w^k and $\tilde{w}^k$ can be written as $w^k =: (y^k, x^k)$ and $\tilde{w}^k =: (\tilde{y}^k, \tilde{x}^k)$, respectively, where $y^k, \tilde{y}^k \in \mathcal{Y}$ and $x^k, \tilde{x}^k \in \mathcal{X}$. Also, the fact that $\sigma_z = 0$ implies that (12) in step 1 is equivalent to

$$\tilde{\lambda}[\theta x^{k-1} + h_1^k] + \tilde{z}^k - z^{k-1} = 0, \qquad h_1^k \in \partial\delta_{K^*}(\tilde{z}^k) = N_{K^*}(\tilde{z}^k), \qquad (23)$$

and hence, that $\tilde{z}^k$ satisfies the optimality conditions of the problem

$$\min_{z \in K^*} \frac{1}{2} \left\| z - \left[z^{k-1} - \tilde{\lambda}\theta x^{k-1} \right] \right\|^2 .$$

Therefore, we conclude that $\tilde{z}^k$ and h_1^k are uniquely determined by

$$\tilde{z}^k = \Pi_{K^*}(z^{k-1} - \tilde{\lambda}\theta x^{k-1}), \qquad h_1^k = -\theta x^{k-1} + \frac{(z^{k-1} - \tilde{z}^k)}{\tilde{\lambda}}. \qquad (24)$$

Moreover, we can easily see that (13) in step 2 is equivalent to setting

$$(\tilde{y}^k, \tilde{x}^k) = (y^{k-1}, x^{k-1}) + \Delta^k, \qquad h_2^k = (0,0), \qquad (25)$$

where the displacement $\Delta^k \in \mathcal{Y} \times \mathcal{X}$ satisfies

$$\|\mathcal{Q}\Delta^k - q^k\|_{\mathcal{W}} \le \sigma_w \|\Delta^k\|_{\mathcal{W}}, \qquad (26)$$

and, the linear mapping $\mathcal{Q} : \mathcal{Y} \times \mathcal{X} \to \mathcal{Y} \times \mathcal{X}$ and the vector $q^k \in \mathcal{Y} \times \mathcal{X}$ are defined as

$$\mathcal{Q} := \begin{bmatrix} \mathcal{I} & \tilde{\lambda}\mathcal{A} \\ -\tilde{\lambda}\xi\mathcal{A}^* & \mathcal{I} \end{bmatrix}, \qquad q^k := \tilde{\lambda} \begin{bmatrix} b - \mathcal{A}x^{k-1} \\ \xi\left(\mathcal{A}^* y^{k-1} + \tilde{z}^k - c\right) \end{bmatrix}.$$

Observe that finding Δ^k satisfying (26) with $\sigma_w = 0$ is equivalent to solving augmented primal-dual linear system $\mathcal{Q}\Delta = q^k$ *exactly*, which can be easily seen to be at least as difficult as projecting a point onto the manifold $\mathcal{M}$ (see O.3 in Section 1). Instead, the approach outlined above assumes $\sigma_w > 0$ and *inexactly* solves this augmented linear system by allowing a relative error as in (26). Clearly a Δ^k satisfying (26) can be found with the help of a suitable iterative linear solver.

We now state an instance of the Special-BD framework based on the ideas outlined above.

Algorithm 1 : Inexact scaled BD method for (1)

0) Let $(z^0, y^0, x^0) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{X}$, $\theta, \xi > 0$, $\sigma_w \in [0,1)$ and $\sigma \in (\sigma_w, 1]$ be given, and set

$$\tilde{\lambda} := \sqrt{\frac{\sigma^2 - \sigma_w^2}{\theta\xi}}, \tag{27}$$

and $k = 1$;

1) set $\tilde{z}^k = \Pi_{K^*}(z^{k-1} - \tilde{\lambda}\theta x^{k-1})$;

2) use a linear solver to find $\Delta^k \in \mathcal{Y} \times \mathcal{X}$ satisfying (26), and set

$$(\tilde{y}^k, \tilde{x}^k) = (y^{k-1}, x^{k-1}) + \Delta^k;$$

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

$$\left\|\lambda(v_z^k, v_y^k, v_x^k) + (\tilde{z}^k, \tilde{y}^k, \tilde{x}^k) - (z^{k-1}, y^{k-1}, x^{k-1})\right\|_{\mathcal{Z}\times\mathcal{W}} \\ \leq \sigma \left\|(\tilde{z}^k, \tilde{y}^k, \tilde{x}^k) - (z^{k-1}, y^{k-1}, x^{k-1})\right\|_{\mathcal{Z}\times\mathcal{W}},$$

where

$$v_z^k = \theta(\tilde{x}^k - x^{k-1}) + \frac{(z^{k-1} - \tilde{z}^k)}{\tilde{\lambda}}, \qquad v_y^k = \mathcal{A}\tilde{x}^k - b, \qquad v_x^k = \xi(c - \mathcal{A}^*\tilde{y}^k - \tilde{z}^k); \tag{28}$$

4) set $(z^k, y^k, x^k) = (z^{k-1}, y^{k-1}, x^{k-1}) - \lambda_k(v_z^k, v_y^k, v_x^k)$ and $k \leftarrow k+1$, and go to step 1.

We now make a few observations about Algorithm 1 and its relationship with the Special-BD framework in the context of (20), (21) and (22). First, it is easy to check that $\tilde{\lambda}$ as in (27) satisfies (11) with $\sigma_z = 0$ and $L = \sqrt{\theta\xi}$ as equality. Second, the discussion preceding Algorithm 1 shows that steps 1 and 2 of Algorithm 1 are equivalent to the same ones of the Special-BD framework with $\sigma_z = 0$. Third, noting that (22), (24) and (25) imply that

$$(v_z^k, v_y^k, v_x^k) = F(\tilde{z}^k, \tilde{y}^k, \tilde{x}^k) + (h_1^k, h_2^k), \tag{29}$$

it follows that steps 3 and 4 of Algorithm 1 are equivalent to the same ones of the Special-BD framework. Fourth, in view of the previous two observations, Algorithm 1 is a special instance of the Special-BD framework for solving (10) where $\mathcal{Z}$, $\mathcal{W}$, $\langle\cdot,\cdot\rangle_{\mathcal{Z}}$, $\langle\cdot,\cdot\rangle_{\mathcal{W}}$, F, h_1 and h_2 are given by (20), (21) and (22). Fifth, λ_k in step 3 of Algorithm 1 can be obtained by solving an easy quadratic equation. Finally, in Subsection 4.1 we discuss how the CG method is used in our implementation to obtain a vector Δ^k satisfying (26).

We now specialize Theorem 2.3 to the context of Algorithm 1. Even though Algorithm 1 is an instance of the Special-BD framework applied to the scaled inclusion (19), the convergence result below is stated with respect to the unscaled inclusion (18).

Theorem 3.2. *Consider the sequences* $\{(z^k,y^k,x^k)\}$, $\{(\tilde z^k,\tilde y^k,\tilde x^k)\}$ *and* $\{(v_z^k,v_y^k,v_x^k)\}$ *generated by Algorithm 1 under the assumption that* $\sigma<1$ *and, for every* $k\in\mathbb{N}$, *define*

$$r_z^k := \theta^{-1}v_z^k, \qquad r_y^k := v_x^k, \qquad r_x^k := \xi^{-1}v_y^k. \tag{30}$$

Let $P^*\in\mathcal{X}$ *and* $D^*\subseteq\mathcal{X}\times\mathcal{Y}$ *denote the set of optimal solutions of* (1) *and* (2), *respectively, and define*

$$\begin{aligned} d_{0,P} &:= \min\{\|x-x^0\| : x\in P^*\},\\ d_{0,D} &:= \min\{\|(z,y)-(z^0,y^0)\| : (z,y)\in D^*\}. \end{aligned}$$

Then, for every $k\in\mathbb{N}$,

$$r_z^k \in \tilde x^k + N_{K^*}(\tilde z^k), \tag{31a}$$

$$r_y^k = \mathcal{A}\tilde x^k - b. \tag{31b}$$

$$r_x^k = c-\mathcal{A}^*\tilde y^k - \tilde z^k, \tag{31c}$$

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

$$\begin{aligned} &\sqrt{\theta\|r_z^i\|^2+\|r_y^i\|^2+\xi\|r_x^i\|^2}\\ &\qquad\le \sqrt{\frac{\xi\theta}{k\,(\sigma^2-\sigma_w^2)}}\sqrt{\left(\frac{1+\sigma}{1-\sigma}\right)\Big(\max\{1,\theta^{-1}\}d_{0,D}^2+\xi^{-1}d_{0,P}^2\Big)}. \end{aligned} \tag{32}$$

Proof. Consider the sequences $\{h_1^k\}$ and $\{h_2^k\}$ defined in (24) and (25). Identities (31b) and (31c) follow immediately from the two last identities in both (28) and (30). Also, it follows from the inclusion in (23) and the definitions of h_1^k, v_z^k and r_z^k in (24), (28) and (30), respectively, that

$$r_z^k = \theta^{-1}v_z^k = \tilde x^k + \theta^{-1}h_1^k \in \tilde x^k + N_{K^*}(\tilde z^k),$$

and hence, that (31a) holds. Let d_0 denote the distance of $((z^0,y^0),x^0)$ to $D^*\times P^*$ with respect to the scaled norm $\|\cdot\|_{\mathcal{Z}\times\mathcal{W}}$, and observe that (21) and, the definitions of $d_{0,P}$ and $d_{0,D}$, imply that

$$d_0 \le \sqrt{\max\{1,\theta^{-1}\}d_{0,D}^2+\xi^{-1}d_{0,P}^2}. \tag{33}$$

Moreover, (29) together with Theorem 2.3 imply the existence of $i \leq k$ such that

$$\|(v_z^i, v_y^i, v_x^i)\|_{\mathcal{Z}\times\mathcal{W}} \leq \sqrt{\frac{1+\sigma}{1-\sigma}}\frac{d_0}{\tilde{\lambda}\sqrt{k}}. \tag{34}$$

Now, combining (34) with the definitions of $\|\cdot\|_{\mathcal{Z}\times\mathcal{W}}$, $\langle\cdot,\cdot\rangle_{\mathcal{Z}}$, $\langle\cdot,\cdot\rangle_{\mathcal{W}}$, r_z^k, r_y^k and r_x^k in (7), (21) and (30) , we have

$$\sqrt{\theta\|r_z^i\|^2 + \|r_y^i\|^2 + \xi\|r_x^i\|^2} \leq \sqrt{\frac{1+\sigma}{1-\sigma}}\frac{d_0}{\tilde{\lambda}\sqrt{k}},$$

which, together with (33) and the definition of $\tilde{\lambda}$ in (27), imply (32). □

We now make some observations about Algorithm 1 and Theorem 3.2. First, the point-wise iteration-complexity bound in Theorem 3.2 is $\mathcal{O}(1/\sqrt{k})$. It is possible to derive an $\mathcal{O}(1/k)$ iteration-complexity ergodic bound for Algorithm 1 as an immediate consequence of Theorem 3.3 of [11] and Theorem 2.4 of [9]. Second, the bound in (32) of Theorem 3.2 sheds light on how the scaling parameters θ and ξ might affect the sizes of the residuals r_z^i, r_y^i and r_x^i. Roughly speaking, viewing all quantities in (32), with the exception of θ, as constants, we see that

$$\|r_z^i\| = \mathcal{O}\left(\max\left\{1, \theta^{-1/2}\right\}\right), \qquad \max\left\{\|r_y^i\|, \|r_x^i\|\right\} = \mathcal{O}\left(\max\left\{1, \theta^{1/2}\right\}\right).$$

Hence, the ratio $\mathcal{R}_\theta := \max\left\{\|r_y^i\|, \|r_x^i\|\right\} / \|r_z^i\|$ can grow significantly as $\theta \to \infty$, while it can become very small as $\theta \to 0$. This suggests that $\mathcal{R}_\theta$ increases (resp., decreases) as ξ increases (resp., decreases). Similarly, viewing all quantities in the bound (32), with the exception of ξ, as constants, we see that

$$\|r_x^i\| = \mathcal{O}\left(\max\left\{1, \xi^{-1/2}\right\}\right), \qquad \|r_y^i\| = \mathcal{O}\left(\max\left\{1, \xi^{1/2}\right\}\right).$$

Hence, the ratio $\mathcal{R}_\xi := \|r_y^i\| / \|r_x^i\|$ can grow significantly as $\xi \to \infty$, while it can become very small as $\xi \to 0$. This suggests that $\mathcal{R}_\xi$ increases (resp., decreases) as ξ increases (resp., decreases). In fact, we have observed in our computational experiments that the ratios $\mathcal{R}_\theta$ and $\mathcal{R}_\xi$ behave just as described. Finally, observe that while the dual iterate $\tilde{z}^k$ is in K^*, the primal iterate $\tilde{x}^k$ is not necessarily in K. However, the following result shows that it is possible to construct a primal iterate $\tilde{u}^k$ which lies in K, is orthogonal to $\tilde{z}^k$ (i.e., $\tilde{u}^k$ and $\tilde{z}^k$ are complementary) and $\liminf_{k\to\infty}\|\mathcal{A}\tilde{u}^k - b\| = 0$.

Corollary 3.3. *Consider the same assumptions as in Theorem 3.2 and, for every $k \in \mathbb{N}$, define*

$$\tilde{u}^k := \tilde{x}^k - r_z^k, \qquad r_u^k := \mathcal{A}\tilde{u}^k - b. \tag{35}$$

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

a) $\tilde{u}^k \in K$, $\tilde{z}^k \in K^*$ and $\langle\tilde{u}^k, \tilde{z}^k\rangle = 0$;

b) there exists $i \leq k$ such that

$$\|r_u^i\| \leq \sqrt{\frac{\xi \max\{\theta, \|\mathcal{A}\|^2\}}{k\,(\sigma^2 - \sigma_w^2)}} \sqrt{\left(\frac{1+\sigma}{1-\sigma}\right)\left(\max\{1,\theta^{-1}\}d_{0,D}^2 + \xi^{-1} d_{0,P}^2\right)}. \tag{36}$$

Proof. To show a), observe that inclusion (31a) and the definition of $\tilde{u}^k$ in (35) imply $\tilde{u}^k \in -N_{K^*}(\tilde{z}^k)$, and hence a) follows. Statement b) follows from (32) and the fact that (31b) and (35) imply

$$\|r_u^i\| = \|r_y^i - \mathcal{A}r_z^i\| \leq \|r_y^i\| + \|\mathcal{A}r_z^i\| \leq \max\{1, \theta^{-1/2}\|\mathcal{A}\|\} \cdot \sqrt{\theta\|r_z^i\|^2 + \|r_y^i\|^2}.$$

□

4 A practical dynamically scaled inexact BD method

This section is divided into two subsections. The first one introduces a practical procedure that uses an iterative linear solver for computing Δ^k as in step 2 of Algorithm 1. The second one describes the stopping criterion used for Algorithm 1 and presents two important refinements of Algorithm 1 based on the ideas introduced in [9] and [8] that allow the scaling parameters ξ and θ to change dynamically.

4.1 Solving step 2 of Algorithm 1 for large and/or dense problems

In this subsection, we present a procedure that uses an iterative linear solver for computing Δ^k as in step 2 of Algorithm 1. More specifically, we show how the CG method applied to a linear system with an $m \times m$ positive definite symmetric coefficient matrix yields a displacement $\Delta^k \in \mathcal{Y} \times \mathcal{X}$ satisfying (26), where $m = \dim \mathcal{Y}$.

In order to describe the procedure for computing $\Delta^k \in \mathcal{Y} \times \mathcal{X}$ satisfying (26), define $\tilde{\mathcal{Q}} : \mathcal{Y} \to \mathcal{Y}$ and $\tilde{q}^k \in \mathcal{Y}$ as

$$\tilde{\mathcal{Q}} := \mathcal{I} + \tilde{\lambda}^2 \xi \mathcal{A}\mathcal{A}^*, \tag{37}$$

$$\tilde{q}^k := \tilde{\lambda}\left(b - \mathcal{A}x^{k-1}\right) - \tilde{\lambda}^2 \xi \mathcal{A}\left(\mathcal{A}^* y^{k-1} + \tilde{z}^k - c\right) \tag{38}$$

and observe that $\tilde{\mathcal{Q}}$ is a self-adjoint positive definite linear operator. The CG method can then be applied to to the linear system $\tilde{\mathcal{Q}}\Delta_y = \tilde{q}^k$ to obtain a solution $\Delta_y^k \in \mathcal{Y}$ satisfying

$$\|\tilde{\mathcal{Q}}\Delta_y^k - \tilde{q}^k\| \leq \sigma_w \|\Delta_y^k\|. \tag{39}$$

(In our implementation, we choose $\Delta = 0$ as the initial point for the CG method.) Setting

$$\Delta^k = (\Delta_y^k, \Delta_x^k)$$

where

$$\Delta_x^k = \tilde{\lambda}\xi \left(\mathcal{A}^* \left(y^{k-1} + \Delta_y^k\right) + \tilde{z}^k - c\right),$$

we can easily check that Δ^k satisfy (26).

We now make some observations about the above procedure for computing the displacement Δ^k. First, the arithmetic complexity of an iteration of the CG method corresponds to that of performing single evaluations of the operators $\mathcal{A}$ and $\mathcal{A}^*$. For example, if $\mathcal{X}$ and $\mathcal{Y}$ are given by (4) and $\mathcal{A}$ is identified with and stored as a sparse matrix, then the arithmetic complexity of an iteration of the CG method is bounded by $\mathcal{O}(\mathrm{nnz}(\mathcal{A}))$. As another example, if $\mathcal{A}$ is the product of two matrices $\mathcal{A}_1$ and $\mathcal{A}_2$ where $\mathrm{nnz}(\mathcal{A}_1)$ and $\mathrm{nnz}(\mathcal{A}_2)$ are significantly smaller than $\mathrm{nnz}(\mathcal{A})$, the bound on the arithmetic complexity of an iteration of the CG method can be improved to $\mathcal{O}(\mathrm{nnz}(\mathcal{A}_1) + \mathrm{nnz}(\mathcal{A}_2))$. Second, in view of (37) and (27), we have $\tilde{\mathcal{Q}} = \mathcal{I} + (\sigma^2 - \sigma_w^2)\theta^{-1}\mathcal{A}\mathcal{A}^*$. Hence, assuming that σ_w^{-1} is $\mathcal{O}(1)$, and defining $\lambda_{\max}^{\mathcal{A}} := \lambda_{\max}(\mathcal{A}\mathcal{A}^*)$ and $\lambda_{\min}^{\mathcal{A}} := \lambda_{\min}(\mathcal{A}\mathcal{A}^*)$, it follows from Proposition A.1 with $\delta = \sigma_w$ that the CG method will compute Δ_y^k satisfying (39) in

$$\mathcal{O}\left(\sqrt{\kappa(\tilde{\mathcal{Q}})}\log\left[\sqrt{\kappa(\tilde{\mathcal{Q}})}\left(1 + \|\tilde{\mathcal{Q}}\|\right)\right]\right) = \mathcal{O}\left(\sqrt{\frac{\theta + \lambda_{\max}^{\mathcal{A}}}{\theta + \lambda_{\min}^{\mathcal{A}}}}\log\left[\sqrt{\frac{\theta + \lambda_{\max}^{\mathcal{A}}}{\theta + \lambda_{\min}^{\mathcal{A}}}}\left(1 + \frac{\lambda_{\max}^{\mathcal{A}}}{\theta}\right)\right]\right) \quad (40)$$

iterations, where the equality follows from the fact that $\|\tilde{Q}\| = \mathcal{O}(1 + \lambda_{\max}^{\mathcal{A}}/\theta)$ and $\|\tilde{Q}^{-1}\| = \mathcal{O}([1 + \lambda_{\min}^{\mathcal{A}}/\theta]^{-1})$. Third, in view of the latter two observations, the above procedure based on the CG method is more suitable for those instances such that: i) the amount of memory required to store $\mathcal{A}$, either explicitly or implicitly, is substantially smaller than the one required to store $\mathcal{A}\mathcal{A}^*$ and its Cholesky factorization; and ii) bound (40) multiplied by the arithmetic complexity of an iteration of the CG method is relatively smaller than the arithmetic complexity of directly solving the linear system $\tilde{\mathcal{Q}}\Delta_y^k = \tilde{q}^k$ (see O.3 in Section 1). Fourth, the bound in (40) does not depend on the iteration count k of Algorithm 1 for fixed θ (see Figure 1). Finally, the bound (40) is strictly decreasing as a function of θ.

4.2 Error measures and dynamic scaling

In this subsection, we describe three measures that quantify the optimality of an approximate solution of (1), namely: the primal infeasibility measure; the dual infeasibility measure; and the relative duality gap. We also describe two important refinements of Algorithm 1 based on the ideas introduced in [9] and [8]. More specifically, we describe: i) a scheme for choosing the initial scaling parameters θ and ξ; and ii) a procedure for dynamically updating the scaling parameters θ and ξ to balance the sizes of three error measures as the algorithm progresses.

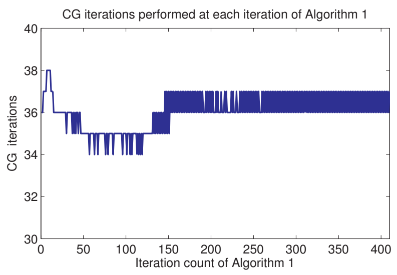

Fig. 1: This example (random SDP instance) illustrates how the number of iterations of the CG method does not change significantly from one iteration of Algorithm 1 to the next when scaling parameter θ remains constant.

For the purpose of describing a stopping criterion for Algorithm 1, define the primal infeasibility measure as

$$\epsilon_P(x) := \frac{\|\mathcal{A}x - b\|}{\|b\| + 1} \qquad \forall x \in \mathcal{X}, \tag{41}$$

and the dual infeasibility measure as

$$\epsilon_D(y,z) := \frac{\|c - \mathcal{A}^* y - z\|}{\|c\| + 1} \qquad \forall (y,z) \in \mathcal{Y} \times \mathcal{X}. \tag{42}$$

Finally, define the relative duality gap as

$$\epsilon_G(x,y) := \frac{\langle c, x\rangle - \langle b, y\rangle}{|\langle c, x\rangle| + |\langle b, y\rangle| + 1} \qquad \forall x \in \mathcal{X},\ \forall y \in \mathcal{Y}. \tag{43}$$

For given tolerances $\bar{\epsilon} > 0$, we stop Algorithm 1 whenever

$$\max\{\epsilon_{P,k}, \epsilon_{D,k}, |\epsilon_{G,k}|\} \le \bar{\epsilon}, \tag{44}$$

where

$$\epsilon_{P,k} := \epsilon_P(\Pi_K(\tilde{x}^k)), \qquad \epsilon_{D,k} := \epsilon_D(\tilde{y}^k, \tilde{z}^k), \qquad \epsilon_{G,k} := \epsilon_G(\Pi_K(\tilde{x}^k), \tilde{y}^k).$$

We now make some observations about the stopping criterion (44). First, the primal and dual infeasibility measures in (41) and (42) do not take into consideration violations with respect to the constraints $x \in K$ and $z \in K^*$, respectively. Since we evaluate them at $(x,y,z) = (\Pi_K(\tilde{x}^k), \tilde{y}^k, \tilde{z}^k)$ and in this case $(x,z) \in K \times K^*$, there is no need to take these violation into account. Second, from the definition of Π_K, Corollary 3.3(a), and identities (31b) and (31c), it follows that

$$\epsilon_{P,k} = \frac{\|r_y^k + \mathcal{A}(\Pi_K(\tilde{x}^k) - \tilde{x}^k)\|}{\|b\| + 1}, \qquad \epsilon_{D,k} = \frac{\|r_x^k\|}{\|c\| + 1},$$

$$\epsilon_{G,k} = \frac{\langle r_x^k, \tilde{x}^k\rangle + \langle r_y^k, \tilde{y}^k\rangle + \langle r_z^k, \tilde{z}^k\rangle + \langle c, \Pi_K(\tilde{x}^k) - x^k\rangle}{|\langle c, \Pi_K(\tilde{x}^k)\rangle| + |\langle b, \tilde{y}^k\rangle| + 1},$$

$$\|\Pi_K(\tilde{x}^k) - x^k\| \le \|\tilde{u}^k - x^k\| = \|r_z^k\|,$$

which together with Theorem 3.2 imply that zero is a cluster value of the sequences $\{\epsilon_{P,k}\}$, $\{\epsilon_{D,k}\}$ and $\{\epsilon_{G,k}\}$ as $k \to \infty$. Hence, Algorithm 1 with the termination criterion (44) will eventually terminate. Third, another possibility is to terminate Algorithm 1 based on the quantities $\epsilon'_{P,k} = \epsilon_P(\tilde{u}^k)$, $\epsilon_{D,k}$ and $\epsilon'_{G,k} := \epsilon_G(\tilde{u}^k, \tilde{y}^k)$, which also approach zero (in a cluster sense) due to Theorem 3.2 and Corollary 3.3. Our current implementation of Algorithm 1 ignores the latter possibility and terminates based on (44). Finally, it should be observed that the termination criterion (44) requires the evaluation of an additional projection for computing $\epsilon_{P,k}$ and $\epsilon_{G,k}$, namely, $\Pi_K(\tilde{x}^k)$. To avoid computing this projection at every iteration, our implementation of Algorithm 1 only checks whether (44) is satisfied when $\max\left\{\epsilon_P(\tilde{x}^k), \epsilon_{D,k}, |\epsilon_G(\tilde{x}^k, \tilde{y}^k)|\right\} \le \bar{\epsilon}$ holds.

We now discuss two important refinements of Algorithm 1 whose goal is to balance the magnitudes of the scaled residuals

$$\rho_{z,k} := \frac{\|r_z^k\|}{|\langle c, \tilde{x}^k\rangle| + |\langle b, \tilde{y}^k\rangle| + 1}, \quad \rho_{y,k} := \frac{\|r_y^k\|}{\|b\| + 1}, \quad \rho_{x,k} := \frac{\|r_x^k\|}{\|c\| + 1}, \quad , \tag{45}$$

where r_z^k, r_y^k and r_x^k are defined in Theorem 3.2. Observe that (45) imply that $\mathcal{R}_{\theta,k} := \max\{\rho_{y,k}, \rho_{x,k}\}/\rho_{z,k} = \mathcal{O}\left(\max\{\|r_y^k\|, \|r_x^k\|\}/\|r_z^k\|\right)$ and $\mathcal{R}_{\xi,k} := \rho_{y,k}/\rho_{x,k} = \mathcal{O}\left(\|r_y^k\|/\|r_x^k\|\right)$. Hence, in view of the second observation in the paragraph following Theorem 3.2, the ratio $\mathcal{R}_{\theta,k}$ (resp., $\mathcal{R}_{\xi,k}$) can grow significantly as $\theta \to \infty$ (resp, $\xi \to \infty$), while it can become very small as $\theta \to 0$ (resp., $\xi \to 0$). This suggests that the ratio $\mathcal{R}_{\theta,k}$ (resp., $\mathcal{R}_{\xi,k}$) increases as θ (resp., ξ) increases, and decreases as θ (resp., ξ) decreases. Indeed, our computational experiments indicate that the ratios $\mathcal{R}_{\theta,k}$ and $\mathcal{R}_{\xi,k}$ behave in this manner.

In the following, let θ_k and ξ_k denote the dynamic values of θ and ξ at the kth iteration of Algorithm 1, respectively. Observe that, in view of (28), (30) and (45), the measures $\rho_{z,k}$, $\rho_{y,k}$ and $\rho_{x,k}$ depend on $\tilde{z}^k$, $\tilde{y}^k$ and $\tilde{x}^k$, whose values in turn depend on the choice of θ_k and ξ_k, in view of steps 1 and 2 of Algorithm 1. Hence, these measures are indeed functions of θ and ξ, which are denoted as$\rho_{z,k}(\theta,\xi)$, $\rho_{y,k}(\theta,\xi)$ and $\rho_{x,k}(\theta,\xi)$.

We first describe a scheme for choosing the initial scaling parameters θ_1 and ξ_1. Let a constant $\rho > 1$ be given and tentatively set $\theta = \xi = 1$. If $\rho_{y,1}(\theta,\xi)/\rho_{x,1}(\theta,\xi) > \rho$ (resp., $\rho_{y,1}(\theta,\xi)/\rho_{x,1}(\theta,\xi) < \rho^{-1}$), we successively divide (resp., successively multiply) the current value of ξ by 2 until $\rho_{y,1}(\theta,\xi)/\rho_{x,1}(\theta,\xi) \le \rho$ (resp., $\rho_{y,1}(\theta,\xi)/\rho_{x,1}(\theta,\xi) \ge \rho^{-1}$) is satisfied, and set $\xi_1 = \xi_1^*$ where ξ_1^* is the last value of ξ. Since we have not updated θ, at this stage we still have $\theta = 1$. At the second stage of this scheme, we update θ in exactly the same manner as above, but in place of $\rho_{y,1}(\theta,\xi)/\rho_{x,1}(\theta,\xi)$ we use the ratio $\max\{\rho_{y,1}(\theta,\xi), \rho_{x,1}(\theta,\xi)\}/\rho_{z,1}(\theta,\xi)$, and set $\theta_1 = \theta_1^*$ where θ_1^*

is the last value of θ. Since there is no guarantee that the latter scheme will terminate, we specify an upper bound on the number of times that ξ and θ can be updated. In our implementation, we set this upper bound to be 20.

We next describe a procedure for dynamically updating the scaling parameters θ and ξ to balance the sizes of the measures $\rho_{z,k}(\theta,\xi)$, $\rho_{y,k}(\theta,\xi)$ and $\rho_{x,k}(\theta,\xi)$ as the algorithm progresses. Similar to the dynamic procedures used in [9] and [8], we use the heuristic of changing θ_k and ξ_k every time a specified number of iterations have been performed. More specifically, given an integer $\bar{k} \geq 1$, and scalars $\gamma_1, \gamma_2 > 1$ and $0 < \tau < 1$, if

$$\gamma_2 \exp\left(\left|\ln\left(\max\{\bar{\rho}_{y,k}, \bar{\rho}_{x,k}\}/\bar{\rho}_{z,k}\right)\right|\right) > \gamma_1 \exp\left(\left|\ln\left(\bar{\rho}_{y,k}/\bar{\rho}_{x,k}\right)\right|\right)$$

we set $\xi_{k+1} = \xi_k$ and use the following dynamic scaling procedure for updating θ_{k+1},

$$\theta_{k+1} = \begin{cases} \theta_k, & k \not\equiv 0 \bmod \bar{k} \text{ or } \gamma_1^{-1} \leq \max\{\bar{\rho}_{y,k}, \bar{\rho}_{x,k}\}/\bar{\rho}_{z,k} \leq \gamma_1 \\ \tau^2 \theta_k, & k \equiv 0 \bmod \bar{k} \text{ and } \max\{\bar{\rho}_{y,k}, \bar{\rho}_{x,k}\}/\bar{\rho}_{z,k} > \gamma_1 \\ \tau^{-2}\theta_k, & k \equiv 0 \bmod \bar{k} \text{ and } \max\{\bar{\rho}_{y,k}, \bar{\rho}_{x,k}\}/\bar{\rho}_{z,k} < \gamma_1^{-1} \end{cases} \quad \forall k \geq 1, \tag{46}$$

otherwise, we set $\theta_{k+1} = \theta_k$ and use the following dynamic scaling procedure for updating ξ_{k+1},

$$\xi_{k+1} = \begin{cases} \xi_k, & k \not\equiv 0 \bmod \bar{k} \text{ or } \gamma_2^{-1} \leq \bar{\rho}_{y,k}/\bar{\rho}_{x,k} \leq \gamma_2 \\ \tau^2 \xi_k, & k \equiv 0 \bmod \bar{k} \text{ and } \bar{\rho}_{y,k}/\bar{\rho}_{x,k} > \gamma_2 \\ \tau^{-2}\xi_k, & k \equiv 0 \bmod \bar{k} \text{ and } \bar{\rho}_{y,k}/\bar{\rho}_{x,k} < \gamma_2^{-1} \end{cases} \quad \forall k \geq 1, \tag{47}$$

where

$$\bar{\rho}_{z,k} = \left(\prod_{i=k-\bar{k}+1}^{k} \rho_{z,i}\right)^{1/\bar{k}}, \quad \bar{\rho}_{y,k} = \left(\prod_{i=k-\bar{k}+1}^{k} \rho_{y,i}\right)^{1/\bar{k}}, \quad \bar{\rho}_{x,k} = \left(\prod_{i=k-\bar{k}+1}^{k} \rho_{x,i}\right)^{1/\bar{k}} \quad \forall k \geq \bar{k}. \tag{48}$$

In our computational experiments, we have used $\bar{k} = 10$, $\gamma_1 = 8$, $\gamma_2 = 2$ and $\tau = 0.9$. Roughly speaking, the above dynamic scaling procedure changes the values of θ and ξ at most a single time in the right direction, so as to balance the sizes of the residuals based on the information provided by their values at the previous $\bar{k}$ iterations. We observe that the above scheme is based on similar ideas as the ones used in [8], but involves two scaling parameters instead of only one as in [8]. Also, since the bound on the CG iterations (40) increases as θ decreases, we stop decreasing θ whenever the CG method starts to perform relatively high number of iterations in order to obtain Δ_y^k satisfying (39).

In our computational experiments, we refer to the variant of Algorithm 1 in which incorporates the above dynamic scaling scheme and uses the CG method to perform step 2 as explained in Subsection 4.1 as the *CG based inexact scaled block-decomposition* (CG-ISBD) method. Figure 2 compares the performance of the CG-ISBD method on a conic SDP instance against the following three

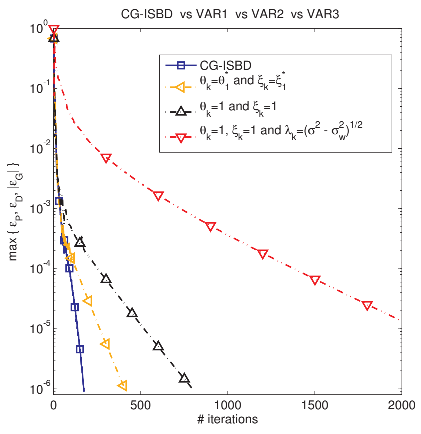

Fig. 2: This example (random SDP instance) illustrates how all the refinements made in the application of the Special-BD framework to problem (1) helped improve the performance of the algorithm.

variants: i) VAR1, the one that removes the dynamic scaling (i.e., set $\theta_k = \theta_1^*$ and $\xi_k = \xi_1^*$, for every $k \geq 1$); ii) VAR2, the one that removes the dynamic scaling and the initialization scheme for θ_1 (i.e., set $\theta_k = 1$ and $\xi_k = 1$, for every $k \geq 1$); and iii) VAR3, the one that removes these latter two refinements and the use of adaptive stepsize (i.e., set $\theta_k = 1$, $\xi_k = 1$ and $\lambda_k = \tilde{\lambda} = \sqrt{\sigma^2 - \sigma_w^2}$, for every $k \geq 1$).

5 Numerical results

In this section, we compare the CG-ISBD method described in Section 4 with the SDPLR method discussed in [2,3,1]. More specifically, we compare these two methods on a collection of extra large-scale conic SDP instances of (1) and (4) where the size and/or density of the linear operator $\mathcal{A}$ is such that the operation of projecting a point onto the manifold $\mathcal{M}$ (see O.3 in Section 1) is prohibitively expensive.

We have implemented CG-ISBD for solving (1) in a MATLAB code which can be found at `http://www.isye.gatech.edu/~cod3/COrtiz/software/`. 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 $X, Y \in \mathcal{S}^{n_s}$. Also, this implementation uses the preconditioned CG procedure `pcg.m` from MATLAB with a modified stopping criterion to obtain Δ_y^k as in Subsection 4.1. On the other hand, we have used the 1.03-beta C implementation of SDPLR[1]. All the computational results were obtained on a single core of a server with 2 Xeon E5-2630 processors at 2.30GHz and 64GB RAM.

In our benchmark, we have considered various large-scale random SDP (RAND) problems, and two large and dense SDP bounds of the Ramsey multiplicity problem (RMP). The RAND instances are pure SDPs, i.e., instances of (1) and (4) with $n_l = n_u = 0$, and were generated using the same random SDP generator used in [7]. In Appendix B, we describe in detail the RMP and the structure of its SDP bound. For each one of the above conic SDP instances, both methods stop whenever (44) with $\bar{\epsilon} = 10^{-6}$ is satisfied with an upper bound of 300,000 seconds running time.

We now make some general remarks about how the results are reported on the tables given below. Table 1 reports the size and the number of non-zeros of $\mathcal{A}$ and $\mathcal{A}\mathcal{A}^*$ for each conic SDP instance. All the instances are large and dense enough so that our server runs out of memory when trying to compute a projection onto the manifold $\mathcal{M}$ (see O.3 in Section 1) based on the Cholesky factorization of $\mathcal{A}\mathcal{A}^*$. Table 2, which compares CG-ISBD against SDPLR, reports the primal and dual infeasibility measures as described in (41) and (42), respectively, the relative duality gap as in (43) and the time (in seconds) for both methods at the last iteration. In addition, Table 2 includes the number of (outer) iterations and the total number of CG iterations performed by CG-ISBD. The residuals (i.e., the primal and dual infeasibility measures, and the relative duality gap) and time taken by any of the two methods for any particular instance are marked in red, and also with an asterisk (*), whenever it cannot solve the instance by the required accuracy, in which case the residuals reported are the ones obtained at the last iteration of the method. Moreover, the time is marked with two asterisks (**) whenever the method is stopped due to numerical errors (e.g., NaN values are obtained). Also, the fastest time in each row is marked in bold.

Finally, Figure 3 plots the performance profiles (see [4]) of CG-ISBD and SDPLR methods based on all instances used in our benchmark. 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 the tested instances x times slower than any other competing method.

[1] Available at `http://dollar.biz.uiowa.edu/~sburer/files/SDPLR-1.03-beta.zip`.

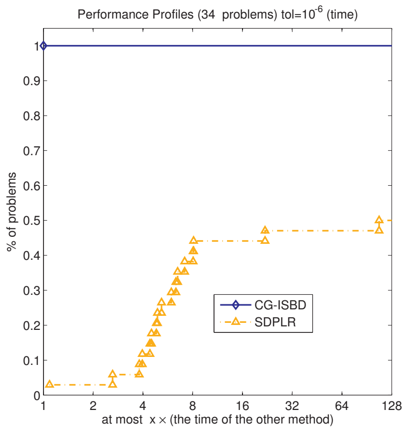

Fig. 3: Performance profiles of CG-ISBD and SDPLR for solving 34 extra large-scale conic SDP instances with accuracy $\bar{\epsilon} = 10^{-6}$.

Table 1: Extra large-scale conic SDP test instances.

Problem		**Sparsity**		**Problem**		**Sparsity**	
INSTANCE	$n_s; n_l \| m$	nnz($\mathcal{A}$)	nnz($\mathcal{A}\mathcal{A}^*$)	INSTANCE	$n_s; n_l \| m$	nnz($\mathcal{A}$)	nnz($\mathcal{A}\mathcal{A}^*$)
RAND_n1K_m200K_p4	1000; 0 — 200000	1,194,036	3,050,786	RAND_n2K_m800K_p4	2000; 0 — 800000	4,776,139	12,199,146
RAND_n1.5K_m250K_p4	1500; 0 — 250000	1,492,374	2,225,686	RAND_n2K_m800K_p5	2000; 0 — 800000	7,959,854	32,408,642
RAND_n1.5K_m400K_p4	1500; 0 — 400000	2,387,989	5,462,130	RAND_n2K_m900K_p4	2000; 0 — 900000	5,372,860	15,328,086
RAND_n1.5K_m400K_p5	1500; 0 — 400000	3,980,228	14,444,644	RAND_n2K_m1M_p4	2000; 0 — 1000000	5,969,926	18,804,856
RAND_n1.5K_m500K_p4	1500; 0 — 500000	2,984,924	8,409,200	RAND_n2.5K_m1.4M_p4	2500; 0 — 1400000	8,357,945	23,740,864
RAND_n1.5K_m500K_p5	1500; 0 — 500000	4,974,944	22,420,950	RAND_n2.5K_m1.5M_p4	2500; 0 — 1500000	8,954,920	27,140,896
RAND_n1.5K_m600K_p4	1500; 0 — 600000	3,582,041	11,989,112	RAND_n2.5K_m1.6M_p4	2500; 0 — 1600000	9,551,974	30,769,976
RAND_n1.5K_m600K_p5	1500; 0 — 600000	5,970,139	32,168,670	RAND_n2.5K_m1.7M_p4	2500; 0 — 1700000	10,148,881	34,633,858
RAND_n1.5K_m700K_p4	1500; 0 — 700000	4,179,023	16,202,622	RAND_n2.5K_m1.8M_p4	2500; 0 — 1800000	10,746,281	38,718,638
RAND_n1.5K_m800K_p4	1500; 0 — 800000	4,775,861	21,042,204	RAND_n3K_m1.6M_p4	3000; 0 — 1600000	9,551,975	21,863,148
RAND_n1.5K_m800K_p5	1500; 0 — 800000	7,959,941	56,937,526	RAND_n3K_m1.7M_p4	3000; 0 — 1700000	10,149,092	24,581,412
RAND_n2K_m250K_p4	2000; 0 — 250000	1,492,475	1,364,154	RAND_n3K_m1.8M_p4	3000; 0 — 1800000	10,745,808	27,453,710
RAND_n2K_m300K_p4	2000; 0 — 300000	1,790,965	1,899,818	RAND_n3K_m1.9M_p4	3000; 0 — 1900000	11,343,196	30,485,510
RAND_n2K_m600K_p4	2000; 0 — 600000	3,582,095	7,011,000	RAND_n3K_m2M_p4	3000; 0 — 2000000	11,939,933	33,663,958
RAND_n2K_m600K_p5	2000; 0 — 600000	5,970,069	18,367,654	RMP_1	90; 274668 — 274668	53,986,254	75,442,510,224
RAND_n2K_m700K_p4	2000; 0 — 700000	4,179,126	9,428,520	RMP_2	90; 274668 — 274668	53,986,254	75,442,510,224
RAND_n2K_m700K_p5	2000; 0 — 700000	6,965,068	24,889,158				

Based on Table 2 and the performance profiles in Figure 3 and the remarks that follow, we can conclude that CG-ISBD substantially outperforms SDPLR in this benchmark. First, CG-ISBD is able to solve all of the test instances with accuracy $\bar{\epsilon} \leq 10^{-6}$ faster than SDPLR, even though SDPLR fails to obtain a solution with such accuracy for 50% of them. Second, the performance profiles in Figure 3 show that SDPLR takes at least 4 and sometimes as much as 100

Table 2: Extra large SDP instances solved using SDPLR and CG-ISBD with accuracy $\bar{\varepsilon} \leq 10^{-6}$.

Problem		**Error Measures**						**Performance**			
		CG-ISBD			SDPLR			CG-ISBD			SDPLR
INSTANCE	$n_s; n_l\|m$	ϵ_P	ϵ_D	ϵ_G	ϵ_P	ϵ_D	ϵ_G	**OUT-IT**	**CG-IT**	**TIME**	**TIME**
RAND_n1K_m200K_p4	1000; 0 — 200000	9.96 -7	4.33 -8	+6.62 -7	5.92 -12	9.98 -7	+6.37 -7	343	20778	**1736**	186308
RAND_n1.5K_m200K_p4	1500; 0 — 200000	9.77 -7	7.23 -9	+8.41 -7	1.07 -8	8.18 -7	+8.17 -7	452	18043	**4330**	16466
RAND_n1.5K_m250K_p4	1500; 0 — 250000	3.17 -7	2.09 -9	+9.51 -7	1.37 -9	4.38 -7	+4.12 -7	448	20050	**4880**	28980
RAND_n1.5K_m400K_p4	1500; 0 — 400000	9.83 -7	2.62 -8	+2.66 -7	1.48 -9	9.15 -7	-1.80 -7	365	20932	**6309**	39937
RAND_n1.5K_m400K_p5	1500; 0 — 400000	8.33 -7	1.66 -8	+9.13 -7	3.80 -10	3.47 -7	+9.73 -7	366	14108	**5978**	30997
RAND_n1.5K_m500K_p4	1500; 0 — 500000	1.44 -7	3.10 -9	+9.60 -7	8.77 -12	1.77 -7	+7.38 -6*	348	23895	**6880**	300000*
RAND_n1.5K_m500K_p5	1500; 0 — 500000	2.41 -7	5.57 -9	+9.44 -7	3.38 -10	1.99 -7	-8.92 -8	338	15707	**7120**	34877
RAND_n1.5K_m600K_p4	1500; 0 — 600000	9.39 -7	4.79 -8	+3.32 -7	1.11 -11	9.62 -7	+2.35 -6*	281	22408	**8004**	284521**
RAND_n1.5K_m600K_p5	1500; 0 — 600000	9.67 -7	3.84 -8	+8.01 -8	4.85 -12	1.02 -6*	+2.22 -6*	285	15844	**8260**	266376**
RAND_n1.5K_m700K_p4	1500; 0 — 700000	9.86 -7	2.18 -9	+6.17 -9	2.80 -11	2.78 -8	-1.42 -6*	284	26690	**10987**	300000*
RAND_n1.5K_m800K_p4	1500; 0 — 800000	2.77 -7	3.90 -9	+6.35 -7	1.42 -8	6.22 -8	+1.08 -7	291	39274	**17508**	19056
RAND_n1.5K_m800K_p5	1500; 0 — 800000	9.74 -7	1.27 -8	+7.40 -7	1.34 -9	3.08 -8	-5.43 -7	275	25877	**14463**	37926
RAND_n2K_m250K_p4	2000; 0 — 250000	9.87 -7	4.81 -8	+3.00 -7	1.84 -10	7.75 -7	+3.28 -6*	292	20743	**14362**	300000*
RAND_n2K_m300K_p4	2000; 0 — 300000	9.37 -7	3.45 -9	+3.06 -7	9.35 -10	3.75 -6*	-9.58 -7	480	17391	**8348**	300000*
RAND_n2K_m600K_p4	2000; 0 — 600000	9.93 -7	4.84 -9	+3.04 -7	1.99 -10	3.06 -7	+7.60 -7	464	20193	**10433**	228621
RAND_n2K_m600K_p5	2000; 0 — 600000	9.84 -7	1.72 -8	+3.31 -8	6.72 -9	8.64 -7	-3.30 -7	390	21856	**11612**	51172
RAND_n2K_m700K_p4	2000; 0 — 700000	9.48 -7	1.27 -8	+8.72 -7	6.63 -10	3.16 -7	+9.76 -7	388	13654	**10657**	51347
RAND_n2K_m700K_p5	2000; 0 — 700000	9.50 -7	2.12 -8	+9.05 -7	1.06 -10	6.87 -7	+4.12 -6*	369	21296	**11033**	300000*
RAND_n2K_m800K_p4	2000; 0 — 800000	9.58 -7	1.98 -8	+7.94 -7	8.23 -11	5.67 -7	+2.32 -6*	359	13712	**11442**	300000*
RAND_n2K_m800K_p5	2000; 0 — 800000	6.72 -7	1.79 -8	+9.77 -7	3.22 -10	9.11 -7	+3.16 -6*	349	23564	**13943**	300000*
RAND_n2K_m900K_p4	2000; 0 — 900000	5.34 -7	1.13 -8	+9.82 -7	6.94 -10	4.00 -7	+2.15 -7	347	14710	**12273**	98772
RAND_n2K_m1M_p4	2000; 0 — 1000000	9.62 -7	1.80 -8	+3.64 -6	4.23 -10	9.22 -7	+8.59 -6*	329	23185	**14229**	300000*
RAND_n2.5K_m1.4M_p4	2500; 0 — 1400000	6.67 -7	1.96 -8	+9.89 -7	5.14 -9	7.97 -7	-3.91 -7	325	22954	**22746**	184138
RAND_n2.5K_m1.5M_p4	2500; 0 — 1500000	9.39 -7	4.01 -8	+7.77 -7	1.75 -8	8.78 -7	+6.78 -7	300	22011	**22572**	89482
RAND_n2.5K_m1.6M_p4	2500; 0 — 1600000	9.83 -7	4.15 -8	+2.16 -7	5.49 -10	4.61 -7	+2.05 -6*	290	22605	**24094**	300000*
RAND_n2.5K_m1.7M_p4	2500; 0 — 1700000	7.71 -7	2.52 -8	+9.99 -7	5.57 -9	5.90 -7	+5.26 -8	283	25173	**29111**	131184
RAND_n2.5K_m1.8M_p4	2500; 0 — 1800000	9.78 -7	2.74 -8	+7.77 -7	2.88 -9	5.75 -7	+7.37 -7	271	24614	**27274**	195103
RAND_n3K_m1.6M_p4	3000; 0 — 1600000	9.78 -7	1.81 -8	+3.30 -7	1.17 -8	9.19 -7	+2.72 -6*	365	21464	**29711**	300000*
RAND_n3K_m1.7M_p4	3000; 0 — 1700000	7.94 -7	1.59 -8	+8.96 -7	6.48 -9	4.80 -7	+7.35 -6*	359	27262	**35973**	300000*
RAND_n3K_m1.8M_p4	3000; 0 — 1800000	9.72 -7	1.36 -8	+2.36 -6	9.01 -9	6.44 -7	+4.11 -7	351	22780	**32586**	211309
RAND_n3K_m1.9M_p4	3000; 0 — 1900000	3.08 -7	5.52 -9	+9.63 -7	1.31 -9	4.73 -7	-4.07 -6*	352	26610	**37020**	300000*
RAND_n3K_m2M_p4	3000; 0 — 2000000	7.49 -7	2.29 -8	+9.16 -7	6.06 -9	1.07 -6*	+7.38 -6*	322	23200	**33294**	300000*
RMP_1	90; 274668 — 274668	7.96 -7	7.36 -7	+5.93 -7	3.22 -11	1.98 +0*	+6.60 -2*	5899	204632	**158023**	300000*
RMP_2	90; 274668 — 274668	4.61 -7	9.49 -9	+3.76 -9	6.78 -13	2.06 +1*	-2.39 -3*	1034	58910	**52698**	164626**

times more running time than CG-ISBD for almost all of the instances that it was able to solve with accuracy $\bar{\epsilon} \leq 10^{-6}$. Third, CG-ISBD is able to solve the two largest problems (RAND_3k_1.9M_p4 and RAND_3k_2M_p4) in terms of number of constraints in approximately one tenth of the maximum running time allocated to SDPLR, which in turn was not able to solve them. Finally, CG-ISBD is able to solve the extremely dense instances RMP_1 and RMP_2 with accuracy 10^{-6} on all the residuals, but SDPLR significantly failed to decrease the dual infeasibility measures and relative duality gaps for both problems.

6 Concluding remarks

In this section we make a few remarks about solving large-scale conic SDPs and the method CG-ISBD proposed in this paper.

Roughly speaking a large-scale conic SDP instance as in (1) and (4) falls into at least one of the following three categories: i) computation of projections onto the cone K (see O.2 in Section 1) cannot be carried out; ii) large number of constraints m but computation of projections onto the manifold $\mathcal{M}$ (see O.3 in Section 1) can be carried out; and iii) large m but computation of projections onto the manifold $\mathcal{M}$ cannot be carried out. Clearly, any conic SDP instance that falls into category i) is beyond the reach of (second-order) interior-point methods. Also, for categories ii) and iii), it is assumed that m is large enough so that the instance cannot be solved by interior-point methods.

For conic SDP instances that fall into category i), the only suitable methods are the low-rank ones as in [2,3,1] (e.g., SDPLR and variations thereof) which avoid computing projections onto the cone K by representing the $n_s \times n_s$ symmetric matrix component X of the variable in (1) as $X = RR^T$ where R is a low-rank matrix. Projection-type methods such as the ones developed in [7,9,8,17,18] are generally quite efficient for conic SDP instances that fall into category ii) but not i) since all their main operations (see O.1, O.2 and O.3 in Section 1) can be carried out. In particular, the BD-type methods presented in [9] and [8] generally outperform the methods of [7,18] and [17], respectively (see the benchmarks in [9] and [8]). It is worth emphasizing that none of the methods in [7,9,8,17,18] can be used to solve instances that fall into category iii) due to the fact that they all require computation of projections onto the manifold $\mathcal{M}$ and/or solutions of linear systems with coefficient matrix related to $\mathcal{A}\mathcal{A}^*$. Even though there exist exact BD methods (see for example the variant of the method in [9] with $\mathcal{U} = \mathcal{I}$) that avoid computation of projections onto the manifold $\mathcal{M}$ and, as a consequence, can be used to solve instances that fall into category iii) but not i), we have observed that they generally do not perform well. In particular, we have observed that the variant of the method in [9] with $\mathcal{U} = \mathcal{I}$ failed to solve most of the instances of our benchmark. Moreover, for the few instances where the latter BD method succeeded, it required an average of 20 times more running time than CG-ISBD. Our contribution on this paper was to present a method, namely CG-ISBD, that avoids computation of projections onto the manifold $\mathcal{M}$ and, as shown in Section 5, is highly efficient for conic SDP instances that fall into category iii) but not i). The most challenging conic SDP instances are the ones that fall into category i) and iii) simultaneously and, although one can (often hopelessly) use one of the methods mentioned above to try to solve such instances, there is clearly a need to develop alternative methods which can efficiently solve them.

Finally, we make a few remarks about the method CG-ISBD proposed in this paper. First, the CG-ISBD method is able to avoid the operation of projecting onto the manifold $\mathcal{M}$ by iterating a CG subroutine until (13) of step 2 in the Special BD framework is satisfied (see Subsection 4.1). However, an alternative (more aggressive) possibility is to iterate the CG subroutine until the HPE condition (14) is satisfied with $\lambda = \tilde{\lambda}$. Second, it should be noted that the goal of the dynamic scaling scheme described in Subsection 4.2 is to reduce the number of outer iterations performed by CG-ISBD, and does not take into account the possibility of also reducing the number of inner iterations performed by the CG subroutine. The development of dynamic scaling schemes which take into account both the outer and inner iterations in order to improve the overall performance of the method is a topic for further study.

Acknowledgements We want to thank Susanne Nieß for generously providing us with the code and data sets to generate SDP bounds of RMPs as in [12].

References

1. Burer, S., Choi, C.: Computational enhancements in low-rank semidefinite programming. Optimization Methods and Software **21**(3), 493–512 (2006). DOI 10.1080/10556780500286582. URL `http://www.tandfonline.com/doi/abs/10.1080/10556780500286582`
2. Burer, S., Monteiro, R.D.C.: A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. Mathematical Programming **95**(2), 329–357 (2003). DOI 10.1007/s10107-002-0352-8. URL `http://dx.doi.org/10.1007/s10107-002-0352-8`
3. Burer, S., Monteiro, R.D.C.: Local minima and convergence in low-rank semidefinite programming. Mathematical Programming **103**(3), 427–444 (2005). DOI 10.1007/s10107-004-0564-1. URL `http://dx.doi.org/10.1007/s10107-004-0564-1`
4. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles (2002). DOI http://dx.doi.org/doi:10.1007/s101070100263. URL `http://dx.doi.org/http://dx.doi.org/doi:10.1007/s101070100263`
5. Gabay, D., Mercier, B.: A dual algorithm for the solution of nonlinear variational problems via finite element approximation. Computers Mathematics with Applications **2**(1), 17 – 40 (1976). DOI 10.1016/0898-1221(76)90003-1. URL `http://www.sciencedirect.com/science/article/pii/0898122176900031`
6. Glowinski, R., Marrocco, A.: Sur l'approximation, par éléments finis d'ordre un, et la résolution, par penalisation-dualité, d'une classe de problèmes de dirichlet non linéaires. RAIRO Anal. Numér. **2**, 41 – 76 (1975)
7. Malick, J., Povh, J., Rendl, F., Wiegele, A.: Regularization methods for semidefinite programming. SIAM J. on Optimization **20**(1), 336–356 (2009). DOI 10.1137/070704575. URL `http://dx.doi.org/10.1137/070704575`
8. Monteiro, R.D.C., Ortiz, C., Svaiter, B.F.: A first-order block-decomposition method for solving two-easy-block structured semidefinite programs. Mathematical Programming Computation pp. 1–48 (2013). DOI 10.1007/s12532-013-0062-7. URL `http://dx.doi.org/10.1007/s12532-013-0062-7`
9. Monteiro, R.D.C., Ortiz, C., Svaiter, B.F.: Implementation of a block-decomposition algorithm for solving large-scale conic semidefinite programming problems. Computational Optimization and Applications **57**(1), 45–69 (2014). DOI 10.1007/s10589-013-9590-3. URL `http://dx.doi.org/10.1007/s10589-013-9590-3`
10. 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 Journal on Optimization **21**(4), 1688–1720 (2011). DOI 10.1137/100801652. URL `http://link.aip.org/link/?SJE/21/1688/1`
11. Monteiro, R.D.C., Svaiter, B.F.: Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers. SIAM Journal on Optimization

23(1), 475–507 (2013). DOI 10.1137/110849468. URL `http://epubs.siam.org/doi/abs/10.1137/110849468`
12. Nieß, S.: Counting monochromatic copies of K_4: a new lower bound for the Ramsey multiplicity problem. arXiv preprint arXiv:1207.4714 (2012)
13. Nocedal, J., Wright, S.J.: Numerical Optimization, second edn. Springer Series in Operations Research and Financial Engineering. Springer (2006)
14. Povh, J., Rendl, F., Wiegele, A.: A boundary point method to solve semidefinite programs. Computing **78**, 277–286 (2006). URL `http://dx.doi.org/10.1007/s00607-006-0182-2`. 10.1007/s00607-006-0182-2
15. Rockafellar, R.T.: On the maximal monotonicity of subdifferential mappings. Pacific J. Math. **33**, 209–216 (1970)
16. Solodov, M.V., Svaiter, B.F.: A hybrid approximate extragradient–proximal point algorithm using the enlargement of a maximal monotone operator. SetValued Analysis **7**(4), 323–345 (1999)
17. Wen, Z., Goldfarb, D., Yin, W.: Alternating direction augmented lagrangian methods for semidefinite programming. Mathematical Programming Computation **2**, 203–230 (2010). URL `http://dx.doi.org/10.1007/s12532-010-0017-1`. 10.1007/s12532-010-0017-1
18. Zhao, X.Y., Sun, D., Toh, K.C.: A Newton-CG augmented lagrangian method for semidefinite programming. SIAM Journal on Optimization **20**(4), 1737–1765 (2010). DOI 10.1137/080718206. URL `http://link.aip.org/link/?SJE/20/1737/1`

A Iteration-complexity of the CG method

Let a self adjoint positive definite linear operator $\tilde{\mathcal{Q}} : \mathcal{Y} \to \mathcal{Y}$ and $\tilde{q} \in \mathcal{Y}$ be given. Consider the problem of finding $\tilde{\Delta} \in \mathcal{Y}$ such that

$$\left\|\tilde{\mathcal{Q}}\tilde{\Delta} - \tilde{q}\right\| \le \delta\|\tilde{\Delta}\|. \tag{49}$$

The following result estimates the number of iterations for the CG method to obtain a solution of (49). (See for example Algorithm 5.2 of [13] for a nice description of the CG method.)

Proposition A.1. *Let $\tilde{\Delta}^i$ be the ith iterate generated by the CG method applied to the linear system $\tilde{\mathcal{Q}}\tilde{\Delta} = \tilde{q}$ and with initial point $\tilde{\Delta}^0 = 0$. Then, $\tilde{\Delta}^i$ satisfies* (49) *for every*

$$i \ge \left\lceil \frac{1}{2}\sqrt{\kappa}\ln\left[2\sqrt{\kappa}\left(1 + \frac{\|\tilde{\mathcal{Q}}\|}{\delta}\right)\right]\right\rceil, \tag{50}$$

where $\kappa = \kappa(\tilde{\mathcal{Q}})$.

Proof. Let $\tilde{\Delta}^*$ be the solution of the linear system $\tilde{\mathcal{Q}}\tilde{\Delta} = \tilde{q}$ and define $e^i := \tilde{\Delta}^i - \tilde{\Delta}^*$ for every $i \ge 0$. For a given $\epsilon > 0$, it is well-known (see for example (5.36) of [13]) that

$$\langle e^i, \tilde{\mathcal{Q}}e^i\rangle \le 4\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2i}\langle e^0, \tilde{\mathcal{Q}}e^0\rangle \qquad \forall i \ge 0.$$

Using the latter inequality, it is easy to see that

$$\|e^i\| \le \epsilon\|e^0\| \qquad \forall i \ge \left\lceil \tfrac{1}{2}\sqrt{\kappa}\ln\tfrac{2\sqrt{\kappa}}{\epsilon}\right\rceil.$$

The latter observation with

$$\epsilon = \frac{\delta}{\delta + \|\tilde{\mathcal{Q}}\|} \tag{51}$$

and the fact that $\epsilon\|\tilde{\mathcal{Q}}\| = \delta\,[1-\epsilon]$, then imply that, for every i satisfying (50), we have

$$\begin{aligned}\left\|\tilde{\mathcal{Q}}\tilde{\Delta}^i - \tilde{q}\right\| &= \left\|\tilde{\mathcal{Q}}\tilde{\Delta}^i - \tilde{\mathcal{Q}}\tilde{\Delta}^*\right\| \le \left\|\tilde{\mathcal{Q}}\right\|\|e^i\| \le \epsilon\left\|\tilde{\mathcal{Q}}\right\|\|e^0\| = \delta\left[\|e^0\| - \epsilon\|e^0\|\right]\\ &\le \delta\left[\|e^0\| - \|e^i\|\right] \le \delta\|e^0 - e^i\| = \delta\|\tilde{\Delta}^0 - \tilde{\Delta}^i\| = \delta\|\tilde{\Delta}^i\|,\end{aligned}$$

where the last equality is due to the assumption that $\tilde{\Delta}^0 = 0$. □

B The Ramsey multiplicity problem

Given a graph G and $t \in \mathbb{N}$, let K_t denote the complete graph on t vertices, $k_t(G)$ denote the number of cliques of size t in G, and $\bar{G}$ denote the complement of G. Given $n \in \mathbb{N}$, define $k_t(n)$ as the minimum number of monochromatic copies of K_t in an improper 2-edge-coloring of K_n, i.e., $k_t(n) := \min\{k_t(G) + k_t(\bar{G}) : |G| = n\}$. If n is sufficiently large compared to t, it follows from Ramsey's theorem that $k_t(n) > 0$. Letting

$$c_t := \lim_{n\to\infty} \frac{k_t(n)}{\binom{n}{t}},$$

the problem of determining c_t is known as the Ramsey Multiplicity Problem (RMP). Moreover, a lower bound on c_t can be found by solving an SDP of the form

$$\begin{aligned} \min\ & -x \\ \text{s.t}\ & \mathcal{A}(X) + xe \leq b, \\ & X \in \mathcal{S}^n_+,\ x \in \mathbb{R}, \end{aligned}$$

where $b \in \mathbb{R}^m_+$, $e = (1, \ldots, 1)^T \in \mathbb{R}^m$ and $\mathcal{A} : \mathcal{S}^n \to \mathbb{R}^m$ is a linear operator (see [12] for details).

The SDP instances from this problem class used in Section 5 were made available to us by S. Nieß.