SIAM J. OPTIM.
Vol. 26, No. 2, pp. 1072–1100

AN EFFICIENT INEXACT ABCD METHOD FOR LEAST SQUARES SEMIDEFINITE PROGRAMMING*

DEFENG SUN†, KIM-CHUAN TOH‡, AND LIUQIN YANG‡

Abstract. We consider least squares semidefinite programming (LSSDP) where the primal matrix variable must satisfy given linear equality and inequality constraints, and must also lie in the intersection of the cone of symmetric positive semidefinite matrices and a simple polyhedral set. We propose an inexact accelerated block coordinate descent (ABCD) method for solving LSSDP via its dual, which can be reformulated as a convex composite minimization problem whose objective is the sum of a coupled quadratic function involving four blocks of variables and two separable nonsmooth functions involving only the first and second block, respectively. Our inexact ABCD method has the attractive $O(1/k^2)$ iteration complexity if the subproblems are solved progressively more accurately. The design of our ABCD method relies on recent advances in the symmetric Gauss–Seidel technique for solving a convex minimization problem whose objective is the sum of a multiblock quadratic function and a nonsmooth function involving only the first block. Extensive numerical experiments on various classes of over 600 large scale LSSDP problems demonstrate that our proposed ABCD method not only can solve the problems to high accuracy, but it is also far more efficient than (a) the well-known block coordinate descent method, (b) an enhanced version of the accelerated randomized block coordinate gradient method, and (c) the accelerated proximal gradient method.

Key words. least squares SDP, accelerated block coordinate descent, symmetric Gauss–Seidel

AMS subject classifications. 90C06, 90C22, 90C25, 65F10

DOI. 10.1137/15M1021799

1. Introduction.

Let $\mathcal{S}^n$ be the space of $n \times n$ real symmetric matrices endowed with the standard trace inner product $\langle \cdot, \cdot \rangle$ and its induced norm $\| \cdot \|$. We denote by $\mathcal{S}^n_+$ the cone of positive semidefinite matrices in $\mathcal{S}^n$. For any matrix $X \in \mathcal{S}^n$, we use $X \succ 0$ ($X \succeq 0$) to indicate that X is a symmetric positive definite (positive semidefinite) matrix.

Consider the following semidefinite programming (SDP) problem

$$\begin{array}{ll} \min & \langle C, X \rangle \\ \text{s.t.} & \mathcal{A}_E(X) = b_E,\ \mathcal{A}_I X - s = 0,\ X \in \mathcal{S}^n_+,\ X \in \mathcal{P}, s \in \mathcal{K}, \end{array} \tag{1}$$

where $b_E \in \Re^{m_E}$ and $C \in \mathcal{S}^n$ are given data, $\mathcal{A}_E : \mathcal{X} \to \Re^{m_E}$ and $\mathcal{A}_I : \mathcal{X} \to \Re^{m_I}$ are two given linear maps, $\mathcal{P}$ and $\mathcal{K}$ are two nonempty simple closed convex sets, e.g., $\mathcal{P} = \{W \in \mathcal{S}^n : \ L \le W \le U\}$ with $L, U \in \mathcal{S}^n$ being given matrices and $\mathcal{K} = \{w \in \Re^{m_I} : \ l \le w \le u\}$ with $l, u \in \Re^{m_I}$ being given vectors. When applying a proximal point algorithm (PPA) [26, 27] to solve (1), we need to solve the following

*Received by the editors May 18, 2015; accepted for publication (in revised form) January 26, 2016; published electronically May 3, 2016.
http://www.siam.org/journals/siopt/26-2/M102179.html

†Department of Mathematics and Risk Management Institute, National University of Singapore, 10 Lower Kent Ridge Road, Singapore (matsundf@nus.edu.sg). This author's research supported in part by the Ministry of Education, Singapore, Academic Research Fund under grant R-146-000-194-112.

‡Department of Mathematics, National University of Singapore, 10 Lower Kent Ridge Road, Singapore (mattohkc@nus.edu.sg, yangliuqin@u.nus.edu). The second author's research was supported in part by the Ministry of Education, Singapore, Academic Research Fund under grant R-146-000-194-112.

subproblem in each iteration for a given point (X^k, s^k) and a parameter $\sigma_k > 0$:

$$(X^{k+1}, s^{k+1}) = \arg\min_{X,s} \left\{ \begin{array}{l} \langle C,\, X\rangle + \frac{1}{2\sigma_k}(\|X - X^k\|^2 + \|s - s^k\|^2) \\ |\ \ \mathcal{A}_E(X) = b_E,\ \mathcal{A}_I X - s = 0,\ X \in \mathcal{S}^n_+,\ X \in \mathcal{P},\ s \in \mathcal{K} \end{array} \right\}. \tag{2}$$

This motivated us to study the following least squares SDP (LSSDP) which includes (2) as a particular case:

$$\begin{aligned} (\mathrm{P}) \quad \min \quad & \tfrac{1}{2}\|X - G\|^2 + \tfrac{1}{2}\|s - g\|^2 \\ \text{s.t.} \quad & \mathcal{A}_E(X) = b_E,\ \mathcal{A}_I X - s = 0,\ X \in \mathcal{S}^n_+,\ X \in \mathcal{P},\ s \in \mathcal{K}, \end{aligned}$$

where $G \in \mathcal{S}^n$, $g \in \Re^{m_I}$ are given data. In order for the PPA to be efficient for solving (1), it is of great importance for us to design an efficient algorithm to solve the above problem (P). Thus, the objective of this paper is to achieve this goal via solving the dual of (P).

The dual of (P) is given by

$$\begin{aligned} (\mathrm{D}) \quad \min F(Z, v, S, y_E, y_I) := & -\langle b_E,\, y_E\rangle + \delta_{\mathcal{S}^n_+}(S) + \delta^*_{\mathcal{P}}(-Z) + \delta^*_{\mathcal{K}}(-v) \\ & + \frac{1}{2}\|\mathcal{A}^*_E y_E + \mathcal{A}^*_I y_I + S + Z + G\|^2 \\ & + \frac{1}{2}\|g + v - y_I\|^2 - \frac{1}{2}\|G\|^2 - \frac{1}{2}\|g\|^2, \end{aligned}$$

where for any given set $\mathcal{C}$, $\delta_{\mathcal{C}}(\cdot)$ is the indicator function over $\mathcal{C}$ such that $\delta_{\mathcal{C}}(u) = 0$ if $u \in \mathcal{C}$ and ∞ otherwise, and $\delta^*_{\mathcal{C}}(\cdot)$ is the conjugate function of $\delta_{\mathcal{C}}$ defined by

$$\delta^*_{\mathcal{C}}(\cdot) = \sup_{W \in \mathcal{C}} \langle \cdot,\, W\rangle. \tag{3}$$

Problem (D) belongs to a general class of multiblock convex optimization problems of the form

$$\min\{\Psi(x) := \theta(x) + \zeta(x)\}, \tag{4}$$

where $x = (x_1, \ldots, x_q) \in \mathcal{X} := \mathcal{X}_1 \times \cdots \times \mathcal{X}_q$, and each $\mathcal{X}_i$ is a finite dimensional real Euclidean space equipped with an inner product $\langle \cdot, \cdot\rangle$ and its induced norm $\|\cdot\|$. Here $\theta(x) = \sum_{i=1}^q \theta_i(x_i)$, $\zeta : \mathcal{X} \to \mathcal{R}$, and $\theta_i : \mathcal{X}_i \to (-\infty, +\infty]$, $i = 1, \ldots, q$, are proper, lower semicontinuous convex functions. We assume that ζ is continuously differentiable on an open neighborhood containing $\mathrm{dom}(\theta) := \{x \in \mathcal{X} : \theta(x) < \infty\}$ and its gradient $\nabla\zeta$ is Lipschitz continuous. Note that one can write (D) in the form of (4) in a number of different ways. One natural choice is of course to express it in the form of (4) for $q = 4$ with $(x_1, x_2, x_3, x_4) \equiv ((Z, v), S, y_E, y_I)$. Another possibility is to express it in the form of (4) for $q = 2$ with $(x_1, x_2) \equiv ((Z, v), (S, y_E, y_I))$.

For the problem in (4), a well-known technique for solving it is the block coordinate descent (BCD) method, for example, see [11, 29, 33, 34] and references therein. Specifically, at iteration k, one may update the blocks successively in the Gauss–Seidel fashion (other rules can also be applied; see [34]):

$$\begin{aligned} x_1^{k+1} &= \arg\min\nolimits_{x_1} \Psi(x_1, x_2^k, \ldots, x_q^k), \\ &\vdots \\ x_i^{k+1} &= \arg\min\nolimits_{x_i} \Psi(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i, x_{i+1}^k, \ldots, x_q^k), \\ &\vdots \\ x_q^{k+1} &= \arg\min\nolimits_{x_q} \Psi(x_1^{k+1}, \ldots, x_{q-1}^{k+1}, x_q). \end{aligned} \tag{5}$$

When the subproblems in (5) are not easily solvable, a popular approach is to use a single step of the proximal gradient method, thus yielding the block coordinate gradient descent (BCGD) method [36, 2].

Problem (4) can also be solved by the accelerated proximal gradient (APG) method with iteration complexity of $O(1/k^2)$ such as in [18, 19, 20, 1, 35]. In the best case, BCD-type methods have an iteration complexity of $O(1/k)$ (see [28, 2]), and can hardly be accelerated to $O(1/k^2)$ as in the case for the APG method. Nevertheless, some researchers have tried to tackle this difficulty from different aspects. Beck and Tetruashvili [2] proposed an accelerated BCGD method for solving (4) by assuming that $\theta \equiv 0$, i.e., without the nonsmooth terms. Very recently, Chambolle and Pock [6] presented an accelerated BCD method for solving (4) by assuming that $\zeta(x)$ has the special form $\zeta(x) = \sum_{1\le i<j\le q} \|A_{ij}x_i + A_{ji}x_j\|^2$. In theory, this method can be applied to the problem (D) by choosing $x_1 = (Z, v)$ and $x_2 = (S, y_E, y_I)$. But the serious practical disadvantage is that the method in [6] does not take care of inexact solutions of the associated subproblems and hence it is not suitable for large scale problems since for (D) the subproblems must be solved numerically.

Besides BCD-type methods based on a deterministic updating order, there has been a wide interest in randomized BCD-type methods. Nesterov [21] presented a randomized BCD method with unconstrained and constrained versions in which the selection of the blocks is not done by a deterministic rule (such as the cyclic rule (5)), but rather via a predescribed distribution. Furthermore, an accelerated $O(1/k^2)$ variant was studied for the unconstrained version. To extend the accelerated version for the more generic problem (4), Fercoq and Richtárik [10] proposed an accelerated $O(1/k^2)$ randomized block coordinate gradient (ARBCG) method. For strongly convex functions, Lin, Lu, and Xiao [17] showed that a variant of this method can achieve a linear convergence rate. However, from our numerical experience, the ARBCG method usually can solve (D) only when each block is solved exactly. Even a numerically much enhanced version of the ARBCG (denoted as eARBCG) method with an efficient iterative method for solving the large linear systems (for which the theoretical convergence needs to be studied) is also typically 3–4 times slower than the accelerated block coordinate descent (ABCD) method with a special deterministic rule which we will propose later.

In this paper we aim to design an efficient *inexact* accelerated BCD-type method whose worst-case iteration complexity is $O(1/k^2)$ for (D). We achieve this goal by first proposing an inexact accelerated BCGD (ABCGD) method for the general convex programming problem (4) with $\theta_3 = \cdots = \theta_q = 0$, and then apply it to (D) to obtain an inexact ABCD method. Note that when ζ is a convex quadratic function, the ABCGD method and the ABCD method are identical. Our ABCD method is designed based on three components. First, we apply a Danskin-type theorem to eliminate the variable x_1 in (4). Then we adapt the inexact APG framework of Jiang, Sun, and Toh proposed in [14] to solve the resulting reduced problem. By choosing an appropriate proximal term and adapting the recently developed inexact symmetric Gauss–Seidel (sGS) decomposition technique [16] for solving a multiblock convex quadratic minimization problem (possibly with a single nonsmooth block), we show that each subproblem in the inexact APG method can be solved efficiently in a fashion almost like the sGS update.

As already mentioned, one can also apply the APG method directly to solve (D), or more generally (4). In this paper, we also adapt the APG method to directly solve (D) for the sake of numerical comparison. In addition, since the ARBCG method does not perform well for solving (D), again for the sake of numerical comparison,

we propose an enhanced version (called eARBCG) of an ARBCG method designed in [17] for solving (D). As one can see later from the extensive numerical experiments conducted to evaluate the performance of various methods, though the BCD, APG, and eARBCG methods are natural choices for solving (D), they are substantially less efficient than the ABCD method that we have designed. In particular, for solving (D), the ABCD method is at least ten times faster than the BCD method for the vast majority of the tested problems. It is quite surprising that a simple novel acceleration step with a special BCD cycle, as in the case of the ABCD method, can improve the performance of the standard Gauss–Seidel BCD method by such a dramatic margin.

The paper is organized as follows. In the next section, we introduce the key ingredients needed to design our proposed algorithm for solving (4), namely, a Danskin-type theorem for parametric optimization, the inexact sGS decomposition technique, and the inexact APG method. In section 3, we describe the integration of the three ingredients to design our inexact ABCGD method for solving (4). Section 4 presents some specializations of the introduced inexact ABCGD method to solve the dual LSSDP problem (D), as well as discussions on the numerical computations involved in solving the subproblems. In section 5, we describe the direct application of the APG method for solving (D). In addition, we also propose an eARBCG in [17] for solving (D). Extensive numerical experiments to evaluate the performance of the ABCD, APG, eARBCG, and BCD methods are presented in section 6. Finally, we conclude the paper in the last section.

Notation. For any given self-adjoint positive semidefinite operator $\mathcal{T}$ that maps a real Euclidean space $\mathcal{X}$ into itself, we use $\mathcal{T}^{1/2}$ to denote the unique self-adjoint positive semidefinite operator such that $\mathcal{T}^{1/2}\mathcal{T}^{1/2} = \mathcal{T}$ and define

$$\|x\|_{\mathcal{T}} := \sqrt{\langle x, \mathcal{T}x\rangle} = \|\mathcal{T}^{1/2}x\| \quad \forall\, x \in \mathcal{X}.$$

2. Preliminaries.

2.1. A Danskin-type theorem.

Here we shall present a Danskin-type theorem for parametric optimization problems.

Let $\mathcal{X}$ and $\mathcal{Y}$ be two finite dimensional real Euclidean spaces each equipped with an inner product $\langle\cdot,\cdot\rangle$ and its induced norm $\|\cdot\|$ and $\varphi : \mathcal{Y} \to (-\infty, +\infty]$ be a lower semicontinuous function and Ω be a nonempty open set in $\mathcal{X}$. Denote the effective domain of φ by $\mathrm{dom}(\varphi)$, which is assumed to be nonempty. Let $\phi(\cdot,\cdot) : \mathcal{Y} \times \mathcal{X} \to (-\infty, +\infty)$ be a continuous function. Define the function $g : \Omega \to [-\infty, +\infty)$ by

$$g(x) = \inf_{y\in\mathcal{Y}}\{\varphi(y) + \phi(y, x)\}, \quad x \in \Omega. \tag{6}$$

For any given $x \in \Omega$, let $\mathcal{M}(x)$ denote the solution set, possibly empty, to the optimization problem (6). The following theorem, largely due to Danskin [7], can be proven by essentially following the proof given in [9, Theorem 10.2.1].

THEOREM 2.1. *Suppose that $\varphi : \mathcal{Y} \to (-\infty, +\infty]$ is a lower semicontinuous function and $\phi(\cdot,\cdot) : \mathcal{Y} \times \mathcal{X} \to (-\infty, +\infty)$ is a continuous function. Assume that for every $y \in \mathrm{dom}(\varphi) \neq \emptyset$, $\phi(y,\cdot)$ is differentiable on Ω and $\nabla_x\phi(\cdot,\cdot)$ is continuous on $\mathrm{dom}(\varphi) \times \Omega$. Let $x \in \Omega$ be given. Suppose that there exists an open neighborhood $\mathcal{N} \subseteq \Omega$ of x such that for each $x' \in \mathcal{N}$, $\mathcal{M}(x')$ is nonempty and that the set $\cup_{x'\in\mathcal{N}}\mathcal{M}(x')$ is bounded. Then the following results hold:*

(i) *The function g is directionally differentiable at x and for any given $d \in \mathcal{X}$,*

$$g'(x;d) = \inf_{y \in \mathcal{M}(x)} \langle \nabla_x \phi(y,x), d \rangle.$$

(ii) *If $\mathcal{M}(x)$ reduces to a singleton, say $\mathcal{M}(x) = \{y(x)\}$, then g is Gâteaux differentiable at x with*

$$\nabla g(x) = \nabla_x \phi(y(x), x).$$

Danskin's Theorem 2.1 will lead to the following results when convexities on φ and ϕ are imposed.

PROPOSITION 2.2. *Suppose that $\varphi : \mathcal{Y} \to (-\infty, +\infty]$ is a proper closed convex function, Ω is an open convex set of $\mathcal{X}$, and $\phi(\cdot,\cdot) : \mathcal{Y} \times \mathcal{X} \to (-\infty, +\infty)$ is a closed convex function. Assume that for every $y \in \mathrm{dom}(\varphi)$, $\phi(y,\cdot)$ is differentiable on Ω and $\nabla_x \phi(\cdot,\cdot)$ is continuous on $\mathrm{dom}(\varphi) \times \Omega$ and that for every $x' \in \Omega$, $\mathcal{M}(x')$ is a singleton, denoted by $\{y(x')\}$. Then the following results hold:*

(i) *Let $x \in \Omega$ be given. If there exists an open neighborhood $\mathcal{N} \subseteq \Omega$ of x such that $y(\cdot)$ is bounded on any nonempty compact subset of $\mathcal{N}$, then the convex function g is differentiable on $\mathcal{N}$ with*

$$\nabla g(x') = \nabla_x \phi(y(x'), x') \quad \forall\, x' \in \mathcal{N}.$$

(ii) *Suppose that there exists a convex open neighborhood $\mathcal{N} \subseteq \Omega$ such that $y(\cdot)$ is bounded on any nonempty compact subset of $\mathcal{N}$. Assume that for any $y \in \mathrm{dom}(\varphi)$, $\nabla_x \phi(y,\cdot)$ is Lipschitz continuous on $\mathcal{N}$ and that there exists a self-joint positive semidefinite linear operator $\Sigma \succeq 0$ such that for all $x \in \mathcal{N}$ and $y \in \mathrm{dom}(\varphi)$,*

$$\Sigma \succeq \mathcal{H} \quad \forall\, \mathcal{H} \in \partial^2_{xx} \phi(y,x),$$

where for any given $y \in \mathrm{dom}(\varphi)$, $\partial^2_{xx}\phi(y,x)$ is the generalized Hessian of $\phi(y,\cdot)$ at x. Then

$$0 \le g(x') - g(x) - \langle \nabla g(x), x' - x \rangle \le \frac{1}{2} \langle x' - x, \Sigma(x' - x) \rangle \quad \forall\, x', x \in \mathcal{N}. \tag{7}$$

Moreover, if $\mathcal{N} = \Omega = \mathcal{X}$, then $\nabla g(\cdot)$ is Lipschitz continuous on $\mathcal{X}$ with the Lipschitz constant $||\Sigma||_2$ (the spectral norm of Σ) and for any $x \in \mathcal{X}$,

$$\Sigma \succeq \mathcal{G} \quad \forall\, \mathcal{G} \in \partial^2_{xx} g(x), \tag{8}$$

where $\partial^2_{xx} g(x)$ denotes the generalized Hessian of g at x.

(iii) *Assume that for every $x \in \Omega$, $\phi(\cdot,x)$ is continuously differentiable on an open neighborhood containing $\mathrm{dom}(\varphi)$. Suppose that there exist two constants $\alpha > 0$ and $L > 0$ such that for all $x', x \in \Omega$,*

$$\langle \nabla_y \phi(y',x) - \nabla_y \phi(y,x), y' - y \rangle \ge \alpha \|y' - y\|^2 \quad \forall\, y', y \in \mathrm{dom}(\varphi)$$

and

$$\|\nabla_y \phi(y,x') - \nabla_y \phi(y,x)\| \le L \|x' - x\| \quad \forall\, y \in \mathrm{dom}(\varphi).$$

Then $y(\cdot)$ is Lipschitz continuous on Ω such that for all $x', x \in \Omega$,

$$\|y(x') - y(x)\| \le (L/\alpha) \|x' - x\|.$$

Proof. Part (i). The convexity of g follows from Rockafellar [25, Theorem 1] and the rest of part (i) follows from the above Danskin theorem 2.1 and [24, Theorem 25.2].

Part (ii). For any $x', x \in \mathcal{N}$, we obtain from part (i) and the generalized mean value theorem (MVT) [13, Theorem 2.3] that

$$\begin{aligned} g(x') - g(x) &= \phi(y(x'), x') + \varphi(y(x')) - [\phi(y(x), x) + \varphi(y(x))] \\ &\le \phi(y(x), x') + \varphi(y(x)) - [\phi(y(x), x) + \varphi(y(x))] \\ &= \phi(y(x), x') - \phi(y(x), x) \\ &\le \langle \nabla_x \phi(y(x), x), x' - x\rangle + \frac{1}{2}\langle x' - x, \Sigma(x' - x)\rangle \quad \text{(by using the MVT)} \\ &= \langle \nabla g(x), x' - x\rangle + \frac{1}{2}\langle x' - x, \Sigma(x' - x)\rangle \quad \text{(by using Part (i))}, \end{aligned}$$

which, together with the convexity of g, implies that (7) holds.

Assume that $\mathcal{N} = \Omega = \mathcal{X}$. By using (7) and [19, Theorem 2.1.5], we can assert that ∇g is globally Lipschitz continuous with the Lipschitz constant $\|\Sigma\|_2$. So by Rademacher's theorem, the Hessian of g exists almost everywhere in $\mathcal{X}$. From (7), we can observe that for any $x \in \mathcal{X}$ such that the Hessian of g at x exists, it holds that

$$\nabla^2_{xx} g(x) \preceq \Sigma.$$

Thus, (8) follows from the definition of the generalized Hessian of g.

Part (iii). The conclusions of part (iii) follow from part (i), the maximal monotonicity of $\partial\varphi(\cdot)$, the assumptions, and the fact that for every $x' \in \Omega$,

$$0 \in \partial\varphi(y(x')) + \nabla_y \phi(y(x'), x').$$

We omit the details here. ∎

2.2. An inexact block sGS iteration. Let $s \ge 2$ be a given integer and $\mathcal{X} := \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_s$, where the $\mathcal{X}_i$'s are real finite dimensional Euclidean spaces. For any $x \in \mathcal{X}$, we write $x \equiv (x_1, x_2, \ldots, x_s)$ with $x_i \in \mathcal{X}_i$. Let $\mathcal{Q} : \mathcal{X} \to \mathcal{X}$ be a given self-adjoint positive semidefinite linear operator. Consider the following block decomposition

$$\mathcal{Q}x \equiv \begin{pmatrix} \mathcal{Q}_{11} & \mathcal{Q}_{12} & \cdots & \mathcal{Q}_{1s} \\ \mathcal{Q}_{12}^* & \mathcal{Q}_{22} & \cdots & \mathcal{Q}_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{Q}_{1s}^* & \mathcal{Q}_{2s}^* & \cdots & \mathcal{Q}_{ss} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_s \end{pmatrix},$$

$$\mathcal{U}x \equiv \begin{pmatrix} 0 & \mathcal{Q}_{12} & \cdots & \mathcal{Q}_{1s} \\ & \ddots & & \vdots \\ & & \ddots & \mathcal{Q}_{s-1,s} \\ & & & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_s \end{pmatrix},$$

where $\mathcal{Q}_{ii} : \mathcal{X}_i \to \mathcal{X}_i,\ i = 1, \ldots, s$, are self-adjoint positive semidefinite linear operators, $\mathcal{Q}_{ij} : \mathcal{X}_j \to \mathcal{X}_i,\ i = 1, \ldots, s-1,\ j > i$, are linear maps. Note that $\mathcal{Q} = \mathcal{U}^* + \mathcal{D} + \mathcal{U}$, where $\mathcal{D}x = (\mathcal{Q}_{11}x_1, \ldots, \mathcal{Q}_{ss}x_s)$.

Let $r \equiv (r_1, r_2, \ldots, r_s) \in \mathcal{X}$ be given. Define the convex quadratic function $h : \mathcal{X} \to \Re$ by

$$h(x) := \frac{1}{2}\langle x,\, \mathcal{Q}x\rangle - \langle r,\, x\rangle, \quad x \in \mathcal{X}.$$

Let $p : \mathcal{X}_1 \to (-\infty, +\infty]$ be a given lower semicontinuous proper convex function.

Here, we further assume that $\mathcal{Q}_{ii},\ i = 1, \ldots, s$, are positive definite. Define

$$x_{\le i} := (x_1, x_2, \ldots, x_i), \quad x_{\ge i} := (x_i, x_{i+1}, \ldots, x_s),\ i = 0, \ldots, s+1$$

with the convention that $x_{\le 0} = x_{\ge s+1} = \emptyset$.

Suppose that $\hat{\delta}_i, \delta_i^+ \in \mathcal{X}_i$, $i = 1, \ldots, s$, are given error vectors, with $\hat{\delta}_1 = 0$. Denote $\hat{\delta} \equiv (\hat{\delta}_1, \ldots, \hat{\delta}_s)$ and $\delta^+ \equiv (\delta_1^+, \ldots, \delta_s^+)$. Define the following operator and vector:

$$\mathrm{sGS}(\mathcal{Q}) := \mathcal{U}\mathcal{D}^{-1}\mathcal{U}^*, \tag{9}$$

$$\Delta(\hat{\delta}, \delta^+) := \delta^+ + \mathcal{U}\mathcal{D}^{-1}(\delta^+ - \hat{\delta}). \tag{10}$$

Let $y \in \mathcal{X}$ be given. Define

$$x^+ := \arg\min_x \left\{ p(x_1) + h(x) + \frac{1}{2}\|x - y\|^2_{\mathrm{sGS}(\mathcal{Q})} - \langle \Delta(\hat{\delta}, \delta^+),\, x\rangle \right\}. \tag{11}$$

The following proposition describing an equivalent BCD-type procedure for computing x^+ is the key ingredient for our subsequent algorithmic developments. The proposition is introduced by Li, Sun, and Toh [15] for the sake of making their Schur-complement-based alternating direction method of multipliers [16] more explicit.

PROPOSITION 2.3. *Assume that the self-adjoint linear operators $\mathcal{Q}_{ii}$, $i = 1, \ldots, s$, are positive definite. Let $y \in \mathcal{X}$ be given. For $i = s, \ldots, 2$, define $\hat{x}_i \in \mathcal{X}_i$ by*

$$\begin{aligned} \hat{x}_i &:= \arg\min_{x_i} \{\, p(y_1) + h(y_{\le i-1}, x_i, \hat{x}_{\ge i+1}) - \langle \hat{\delta}_i,\, x_i\rangle \} \\ &= \mathcal{Q}_{ii}^{-1}\left(r_i + \hat{\delta}_i - \sum_{j=1}^{i-1} \mathcal{Q}_{ji}^* y_j - \sum_{j=i+1}^{s} \mathcal{Q}_{ij}\hat{x}_j \right). \end{aligned} \tag{12}$$

Then the optimal solution x^+ defined by (11) *can be obtained exactly via*

$$\begin{cases} x_1^+ = \arg\min_{x_1} \{\, p(x_1) + h(x_1, \hat{x}_{\ge 2}) - \langle \delta_1^+,\, x_1\rangle \}, \\ x_i^+ = \arg\min_{x_i} \{\, p(x_1^+) + h(x_{\le i-1}^+, x_i, \hat{x}_{\ge i+1}) - \langle \delta_i^+,\, x_i\rangle \} \\ \quad = \mathcal{Q}_{ii}^{-1}\left(r_i + \delta_i^+ - \sum_{j=1}^{i-1} \mathcal{Q}_{ji}^* x_j^+ - \sum_{j=i+1}^{s} \mathcal{Q}_{ij}\hat{x}_j \right), \quad i = 2, \ldots, s. \end{cases} \tag{13}$$

Furthermore, $\mathcal{H} := \mathcal{Q} + \mathrm{sGS}(\mathcal{Q}) = (\mathcal{D} + \mathcal{U})\mathcal{D}^{-1}(\mathcal{D} + \mathcal{U}^*)$ is positive definite.

For later use, we also state the following proposition.

PROPOSITION 2.4. *Suppose that $\mathcal{H} := \mathcal{Q} + \mathrm{sGS}(\mathcal{Q}) = (\mathcal{D} + \mathcal{U})\mathcal{D}^{-1}(\mathcal{D} + \mathcal{U}^*)$ is positive definite. Let $\xi = \|\mathcal{H}^{-1/2}\Delta(\hat{\delta}, \delta^+)\|$. Then*

$$\xi = \|\mathcal{D}^{-1/2}(\delta^+ - \hat{\delta}) + \mathcal{D}^{1/2}(\mathcal{D} + \mathcal{U})^{-1}\hat{\delta}\| \le \|\mathcal{D}^{-1/2}(\delta^+ - \hat{\delta})\| + \|\mathcal{H}^{-1/2}\hat{\delta}\|. \tag{14}$$

Proof. The proof is straightforward by using the factorization of $\mathcal{H}$. ∎

Remark 1. In the computations in (12) and (13), we should interpret the solutions $\hat{x}_i$, x_i^+ as approximate solutions to the minimization problems without the terms involving $\hat{\delta}_i$ and δ_i^+. Once these approximate solutions have been computed, they would generate the error vectors $\hat{\delta}_i$ and δ_i^+. With these known error vectors, we know that the computed approximate solutions are actually the exact solutions to the minimization problems in (12) and (13).

2.3. An inexact APG method. For more generality, we consider the following minimization problem

$$\min\{F(x) := p(x) + f(x) \mid \ x \in \mathcal{X}\}, \tag{15}$$

where $\mathcal{X}$ is a finite dimensional real Euclidean space. The functions $f : \mathcal{X} \to \Re$, $p : \mathcal{X} \to (-\infty, \infty]$ are proper, lower semicontinuous convex functions (possibly nonsmooth). We assume that f is continuously differentiable on $\mathcal{X}$ and its gradient ∇f is Lipschitz continuous with modulus L on $\mathcal{X}$, i.e.,

$$\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\| \quad \forall\, x, y \in \mathcal{X}.$$

We also assume that problem (15) is solvable with an optimal solution $x^* \in \mathrm{dom}(p)$. The inexact APG algorithm proposed by Jiang, Sun, and Toh [14] for solving (15) is described as follows.

Algorithm 1. Input $y^1 = x^0 \in \mathrm{dom}(p)$, $t_1 = 1$. Set $k = 1$. Iterate the following steps.

Step 1. Find an approximate minimizer

$$x^k \approx \arg\min_{y\in\mathcal{X}} \Big\{ f(y^k) + \langle \nabla f(y^k),\, y - y^k\rangle + \frac{1}{2}\langle y - y^k,\, \mathcal{H}_k(y - y^k)\rangle + p(y) \Big\}, \tag{16}$$

where $\mathcal{H}_k$ is a self-adjoint positive definite linear operator that is chosen by the user.

Step 2. Compute $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$ and $y^{k+1} = x^k + \left(\frac{t_k - 1}{t_{k+1}}\right)(x^k - x^{k-1})$.

Given any positive definite linear operator $\mathcal{H}_k : \mathcal{X} \to \mathcal{X}$, define $q_k(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \Re$ by

$$q_k(x, y) = f(y) + \langle \nabla f(y),\, x - y\rangle + \frac{1}{2}\langle x - y,\, \mathcal{H}_k(x - y)\rangle.$$

Let $\{\epsilon_k\}$ be a given convergent sequence of nonnegative numbers such that

$$\sum_{k=1}^{\infty} \epsilon_k < \infty.$$

Suppose that for each k, we have an approximate minimizer

$$x^k \approx \arg\min\{p(x) + q_k(x, y^k) \mid \ x \in \mathcal{X}\} \tag{17}$$

that satisfies the following admissible conditions

$$f(x^k) \ \le \ q_k(x^k, y^k), \tag{18}$$

$$\nabla f(y^k) + \mathcal{H}_k(x^k - y^k) + \gamma^k = \Delta^k \quad \text{with } \|\mathcal{H}_k^{-1/2}\Delta^k\| \le \epsilon_k/(\sqrt{2}t_k), \tag{19}$$

where $\gamma^k \in \partial p(x^k)$. (Note that for x^k to be an approximate minimizer, we must have $x^k \in \mathrm{dom}(p)$.) Then the inexact APG algorithm described in Algorithm 1 has the following iteration complexity result.

THEOREM 2.5. *Suppose that the conditions* (18) *and* (19) *hold and* $\mathcal{H}_{k-1} \succeq \mathcal{H}_k \succ 0$ *for all* k. *Then*

$$0 \le F(x^k) - F(x^*) \le \frac{4}{(k+1)^2}\Big((\sqrt{\tau} + \bar{\epsilon}_k)^2 + 2\tilde{\epsilon}_k\Big), \tag{20}$$

where $\tau = \frac{1}{2}\langle x^0 - x^*,\, \mathcal{H}_1(x^0 - x^*)\rangle$, $\bar{\epsilon}_k = \sum_{j=1}^k \epsilon_j$, $\tilde{\epsilon}_k = \sum_{j=1}^k \epsilon_j^2$.

Proof. See [14, Theorem 2.1] for the proof. ∎

Note that Chambolle and Dossal [5] discussed the convergence of the iterates of an APG (called FISTA in [5]). However, the convergence of the iterates of the inexact versions of the APG including the proposed one here is still unknown.

3. An inexact ABCGD method. We consider the problem

$$\min\{\bar{F}(x_0, x) := \varphi(x_0) + p(x_1) + \phi(x_0, x) \mid\ x_0 \in \mathcal{X}_0,\ x \in \mathcal{X}\}, \tag{21}$$

where $x = (x_1, \ldots, x_s) \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_s (=: \mathcal{X})$, and $\varphi : \mathcal{X}_0 \to (-\infty, \infty]$, $p : \mathcal{X}_1 \to (-\infty, \infty]$, $\phi : \mathcal{X}_0 \times \mathcal{X} \to \Re$ are three closed proper convex functions and $\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_s$ are finite dimensional real Euclidean spaces. We assume that problem (21) is solvable with an optimal solution $(x_0^*, x^*) \in \mathrm{dom}(\varphi) \times \mathrm{dom}(p) \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_s$. Define the function $f : \mathcal{X} \to [-\infty, +\infty)$ by

$$f(x) = \inf_{x_0 \in \mathcal{X}_0} \{\varphi(x_0) + \phi(x_0, x)\}, \quad x = (x_1, \ldots, x_s) \in \mathcal{X}. \tag{22}$$

For any given $x \in \mathcal{X}$, let $\mathcal{M}(x)$ denote the solution set to the optimization problem in (22). Suppose that the assumptions in part (ii)[1] of Proposition 2.2 imposed on φ and ϕ hold for $\Omega = \mathcal{X}$. Then, by part (ii) of Proposition 2.2, we know that f is continuously differentiable and its gradient ∇f is globally Lipschitz continuous, and for all $x, y \in \mathcal{X}$,

$$f(x) - f(y) - \langle \nabla f(y), x - y\rangle \le \frac{1}{2}\langle x - y, \mathcal{L}(x - y)\rangle, \tag{23}$$

where $\mathcal{L} : \mathcal{X} \to \mathcal{X}$ is a self-adjoint positive semidefinite linear operator such that for any $x \in \mathcal{X}$,

$$\mathcal{L} \succeq \mathcal{H} \quad \forall\, \mathcal{H} \in \partial^2 f(x), \tag{24}$$

where $\partial^2 f(x)$ is the generalized Hessian of f at x. One natural choice, though not necessarily the best, for $\mathcal{L}$ is $\mathcal{L} = \Sigma$, where Σ is a self-adjoint positive semidefinite linear operator that satisfies

$$\Sigma \succeq \mathcal{H} \quad \forall\, \mathcal{H} \in \partial^2_{xx}\phi(x_0, x) \quad \forall\, x_0 \in \mathrm{dom}\varphi,\ x \in \mathcal{X}. \tag{25}$$

Here for any given $x_0 \in \mathrm{dom}\varphi$, $\partial^2_{xx}\phi(x_0, x)$ is the generalized Hessian of $\phi(x_0, \cdot)$ at x.

[1] For subsequent discussions, we do not need the assumptions in part (iii) of Proposition 2.2 though the results there have their own merits.

Now we consider an equivalent problem of (21):

$$\min\{F(x) := p(x_1) + f(x) \mid \ x \in \mathcal{X}\}. \tag{26}$$

Given, a self-adjoint positive semidefinite linear operator $\mathcal{Q} : \mathcal{X} \to \mathcal{X}$ such that

$$\mathcal{Q} \succeq \mathcal{L},\ \mathcal{Q}_{ii} \succ 0,\ i = 1, \ldots, s.$$

Define $h(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \Re$ by

$$\begin{aligned} h(x, y) &:= f(y) + \langle \nabla f(y),\, x - y\rangle + \frac{1}{2}\langle x - y,\, \mathcal{Q}(x - y)\rangle \\ &= f(y) + \langle \nabla_x \phi(x_0(y), y),\, x - y\rangle + \frac{1}{2}\langle x - y,\, \mathcal{Q}(x - y)\rangle, \quad x,\, y \in \mathcal{X}, \end{aligned} \tag{27}$$

where $x_0(y)$ is the unique solution to $\inf_{x_0 \in \mathcal{X}_0}\{\varphi(x_0) + \phi(x_0, y)\}$. From (23), we have

$$f(x) \le h(x, y) \quad \forall x,\ y \in \mathcal{X}. \tag{28}$$

We can now apply the inexact APG method described in Algorithm 1 to problem (26) to obtain the following inexact ABCGD algorithm for problem (21).

Algorithm 2. Input $y^1 = x^0 \in \mathrm{dom}(p) \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_s$, $t_1 = 1$. Let $\{\epsilon_k\}$ be a summable sequence of nonnegative numbers. Set $k = 1$. Iterate the following steps.

Step 1. Compute

$$x_0^k = \arg\min_{x_0} \Big\{\varphi(x_0) + \phi(x_0, \widetilde{x}^k)\Big\}. \tag{29}$$

Let $q_k(x, \widetilde{x}^k) = h(x, \widetilde{x}^k) + \frac{1}{2}\|x - \widetilde{x}^k\|^2_{\mathrm{sGS}(\mathcal{Q})}$. Using the inexact block sGS iteration, compute

$$\begin{aligned} x^k &\approx \arg\min_{x \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_s} \Big\{p(x_1) + q_k(x, \widetilde{x}^k)\Big\} \\ &= \arg\min_{x \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_s} \Big\{p(x_1) + f(\widetilde{x}^k) + \langle \nabla_x \phi(x_0^k, \widetilde{x}^k),\, x - \widetilde{x}^k\rangle \\ &\qquad + \frac{1}{2}\langle x - \widetilde{x}^k,\, \mathcal{Q}(x - \widetilde{x}^k)\rangle + \frac{1}{2}\|x - \widetilde{x}^k\|^2_{\mathrm{sGS}(\mathcal{Q})}\Big\} \end{aligned} \tag{30}$$

such that x^k satisfies the admissible conditions stated in (18) and (19) with $\mathcal{H}_k = \mathcal{H} := \mathcal{Q} + \mathrm{sGS}(\mathcal{Q})$, and the residual vector Δ^k satisfies $\|\mathcal{H}^{-1/2}\Delta^k\| \le \frac{\epsilon_k}{\sqrt{2}\, t_k}$.

Step 2. Compute $t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$ and $\widetilde{x}^{k+1} = x^k + \left(\frac{t_k - 1}{t_{k+1}}\right)(x^k - x^{k-1})$.

Remark 2. Note that by (28), the admissible condition (18) is automatically satisfied by the approximate solution x^k in (30). But in order for x^k to satisfy the admissible condition (19), we need to control the error incurred when solving (30) via the inexact block sGS iteration. Specifically, suppose $\hat{\delta}_i^k,\ \delta_i^k \in \mathcal{X}_i,\ i = 1, \ldots, s$, with $\hat{\delta}_1 = 0$, are the error vectors produced by the inexact block sGS iteration. Then the residual vector Δ^k in the admissible condition (19) is given by $\Delta^k = \Delta(\hat{\delta}^k, \delta^k)$ (see

(10)), and by Proposition 2.4,

$$\|\mathcal{H}^{-1/2}\Delta^k\| \le (2\|\mathcal{D}^{-1/2}\| + \|\mathcal{H}^{-1/2}\|)\max\{\|\delta^k\|, \|\hat{\delta}^k\|\}, \tag{31}$$

where $\mathcal{D} = \mathrm{diag}(\mathcal{Q}_{11}, \ldots, \mathcal{Q}_{ss})$. Thus in solving (30), we require that

$$\max\{\|\delta^k\|, \|\hat{\delta}^k\|\} \le \frac{\epsilon_k}{\beta\sqrt{2}t_k}, \tag{32}$$

where $\beta := 2\|\mathcal{D}^{-1/2}\| + \|\mathcal{H}^{-1/2}\|$, and hence $\|\mathcal{H}^{-1/2}\Delta^k\| \le \frac{\epsilon_k}{\sqrt{2}t_k}$.

The iteration complexity result for the inexact ABCGD algorithm described in Algorithm 2 follows from Theorem 2.5 without much difficulty.

THEOREM 3.1. *Denote $\mathcal{H} = \mathcal{Q}+\mathrm{sGS}(\mathcal{Q})$. Let $\{(x_0^k, x^k)\}$ be the sequence generated by Algorithm 2 and the condition (32) be satisfied. Then*

$$0 \le F(x^k) - F(x^*) \le \frac{4}{(k+1)^2}\Big((\sqrt{\tau} + \bar{\epsilon}_k)^2 + 2\tilde{\epsilon}_k\Big), \tag{33}$$

where $\tau = \frac{1}{2}\langle x^0 - x^*, \mathcal{H}(x^0 - x^*)\rangle$, $\bar{\epsilon}_k = \sum_{j=1}^k \epsilon_j$, $\tilde{\epsilon}_k = \sum_{j=1}^k \epsilon_j^2$, and F is defined in (26).

Proof. From (28), we have

$$F(x^k) \le p(x_1^k) + h(x^k, \widetilde{x}^k) + \frac{1}{2}\|x^k - \widetilde{x}^k\|^2_{\mathrm{sGS}(\mathcal{Q})}. \tag{34}$$

From the optimality condition for (30), we can obtain

$$\nabla_x \phi(x_0^k, \widetilde{x}^k) + \mathcal{H}(x^k - \widetilde{x}^k) + \gamma^k = \Delta(\hat{\delta}^k, \delta^k), \tag{35}$$

where $\gamma^k \in \partial p(x_1^k)$. By using (32) and Proposition 2.4, we know that

$$\begin{aligned}\|\mathcal{H}^{-1/2}\Delta(\hat{\delta}^k, \delta^k)\| &\le \|\mathcal{D}^{-1/2}(\delta^k - \hat{\delta}^k)\| + \|\mathcal{H}^{-1/2}\hat{\delta}^k\| \\ &\le (2\|\mathcal{D}^{-1/2}\| + \|\mathcal{H}^{-1/2}\|)\max\{\|\delta^k\|, \|\hat{\delta}^k\|\} \le \frac{\epsilon_k}{\sqrt{2}t_k}.\end{aligned}$$

Then (33) follows from Theorem 2.5. ∎

Remark 3. (a) Since we use the sGS iteration described in Proposition 2.3 to compute x^k in (30), we can see that Step 1 is actually one special BCGD cycle with the order $x_0 \to \hat{x}_s \to \hat{x}_{s-1} \to \cdots \to \hat{x}_2 \to x_1 \to x_2 \cdots \to x_s$.

(b) Assume that x_0^k has been computed, and $\hat{x}_s^k, \ldots, \hat{x}_2^k$ and x_1^k have been computed in the backward sweep of the sGS iteration for solving (30). One may try to estimate $x_2^k, \ldots, x_s^k$ by using $\hat{x}_2^k, \ldots, \hat{x}_s^k$, respectively. In this case the corresponding residual vector δ^k would be given by

$$\delta_i^k = \hat{\delta}_i^k + \sum_{j=1}^{i-1} \mathcal{Q}_{ji}^*(\hat{x}_i^k - \widetilde{x}_i^k), \quad i = 2, \ldots, s. \tag{36}$$

In practice, we may accept such an approximate solution $x_i^k = \hat{x}_i^k$ for $i = 2, \ldots, s$, provided a condition of the form $\|\delta_i^k\| \le c\|\hat{\delta}_i^k\|$ (for some constant $c > 1$ say $c = 10$) is satisfied for $i = 2, \ldots, s$. When such an approximation is admissible, then the linear systems involving $\mathcal{Q}_{ii}$ need only be solved once instead of twice for $i = 2, \ldots, s$. We should emphasize that such a saving can be significant when the linear systems are solved by a Krylov iterative method such as the preconditioned conjugate gradient (PCG) method. Of course, if the linear systems are solved by computing the Cholesky factorization (which is done only once at the start of the algorithm) of $\mathcal{Q}_{ii}$, then the savings would not be as substantial.

4. Two variants of the inexact ABCD method for (D). Now we can apply Algorithm 2 directly to (D) to obtain two variants of the inexact ABCD method. In the first variant, we apply Algorithm 2 to (D) by expressing it in the form of (21) with $s = 3$ and $(x_0, x_1, x_2, x_3) = ((Z, v), S, y_E, y_I)$. Specifically,

$$\begin{aligned}
\varphi((Z,v)) &= \delta^*_{\mathcal{P}}(-Z) + \delta^*_{\mathcal{K}}(-v), \qquad p(S) = \delta_{\mathcal{S}^n_+}(S),\\
\phi((Z,v),S,y_E,y_I) &= -\langle b_E,\, y_E\rangle + \frac{1}{2}\|\mathcal{A}_E^* y_E + \mathcal{A}_I^* y_I + S + Z + G\|^2\\
&\quad + \frac{1}{2}\|g + v - y_I\|^2 - \text{const},\\
f(S,y_E,y_I) &= \inf_{(Z,v)}\{\varphi((Z,v)) + \phi((Z,v),S,y_E,y_I)\}\\
&= -\langle b_E,\, y_E\rangle + \frac{1}{2}\|\Pi_{\mathcal{P}}(\mathcal{A}_E^* y_E + \mathcal{A}_I^* y_I + S + G)\|^2\\
&\quad + \frac{1}{2}\|\Pi_{\mathcal{K}}(g - y_I)\|^2 - \text{const},
\end{aligned}$$

where const $= \frac{1}{2}\|G\|^2 + \frac{1}{2}\|g\|^2$, and we take

$$\mathcal{Q} = \mathcal{L} = \nabla^2_{(S,y_E,y_I)}\phi((Z,v),S,y_E,y_I) = \begin{pmatrix} \mathcal{I}_{n^2} & \mathcal{A}_E^* & \mathcal{A}_I^* \\ \mathcal{A}_E & \mathcal{A}_E\mathcal{A}_E^* & \mathcal{A}_E\mathcal{A}_I^* \\ \mathcal{A}_I & \mathcal{A}_I\mathcal{A}_E^* & \mathcal{A}_I\mathcal{A}_I^* + \mathcal{I}_{m_I} \end{pmatrix}. \tag{37}$$

In the second variant, we express (D) in the form of (21) with $s = 1$ and $(x_0, x_1) = ((Z, v), (S, y_E, y_I))$. For the remaining parts of this paper, we assume that $\mathcal{A}_E : \mathcal{X} \to \Re^{m_E}$ is onto and the solution set of (D) is nonempty. The convergence of both variants follows from Theorem 3.1 for Algorithm 2 and will not be repeated here.

The detailed steps of the first variant of the inexact ABCD method are given as follows.

Remark 4. (a) To compute $\hat{y}_E^k$ in Algorithm ABCD-1, we need to solve the following linear system of equations:

$$(\mathcal{A}_E\mathcal{A}_E^*)\hat{y}_E^k \approx b_E - \mathcal{A}_E(\mathcal{A}_I^*\tilde{y}_I^k + \widetilde{S}^k + Z^k + G) \tag{38}$$

with the residual norm

$$\|\hat{\delta}_E^k\| = \|b_E - \mathcal{A}_E(\mathcal{A}_I^*\tilde{y}_I^k + \widetilde{S}^k + Z^k + G) - \mathcal{A}_E\mathcal{A}_E^*\hat{y}^k\| \le \epsilon_k/(\sqrt{2}t_k). \tag{39}$$

Algorithm ABCD-1: **An inexact ABCD method for (D).**

Select an initial point $(\widetilde{Z}^1, \tilde{v}^1, \widetilde{S}^1, \tilde{y}_E^1, \tilde{y}_I^1) = (Z^0, v^0, S^0, y_E^0, y_I^0)$ with $(-Z^0, -v^0) \in \mathrm{dom}(\delta^*_{\mathcal{P}}) \times \mathrm{dom}(\delta^*_{\mathcal{K}})$. Let $\{\epsilon_k\}$ be a summable sequence of nonnegative numbers, and $t_1 = 1$. Set $k = 1$. Iterate the following steps.

Step 1. Let $\widetilde{R}^k = \mathcal{A}_E^* \tilde{y}_E^k + \mathcal{A}_I^* \tilde{y}_I^k + \widetilde{S}^k + G$. Compute

$$(Z^k, v^k) = \arg\min_{Z,v} \big\{ F(Z, v, \widetilde{S}^k, \tilde{y}_E^k, \tilde{y}_I^k) \big\} = (\Pi_{\mathcal{P}}(\widetilde{R}^k) - \widetilde{R}^k,\ \Pi_{\mathcal{K}}(g - \tilde{y}_I^k) - (g - \tilde{y}_I^k)),$$

$$\hat{y}_E^k = \arg\min_{y_E} \big\{ F(Z^k, v^k, \widetilde{S}^k, y_E, \tilde{y}_I^k) - \langle \hat{\delta}_E^k,\, y_E \rangle \big\},$$

$$\hat{y}_I^k = \arg\min_{y_I} \big\{ F(Z^k, v^k, \widetilde{S}^k, \hat{y}_E^k, y_I) - \langle \hat{\delta}_I^k,\, y_I \rangle \big\},$$

$$S^k = \arg\min_S \big\{ F(Z^k, v^k, S, \hat{y}_E^k, \hat{y}_I^k) \big\} = \Pi_{\mathcal{S}_+^n} \Big(-(\mathcal{A}_E^* \hat{y}_E^k + \mathcal{A}_I^* \hat{y}_I^k + Z^k + G) \Big),$$

$$y_I^k = \arg\min_{y_I} \big\{ F(Z^k, v^k, S^k, \hat{y}_E^k, y_I) - \langle \delta_I^k,\, y_I \rangle \big\},$$

$$y_E^k = \arg\min_{y_E} \big\{ F(Z^k, v^k, S^k, y_E, y_I^k) - \langle \delta_E^k,\, y_E \rangle \big\},$$

where $\delta_E^k, \hat{\delta}_E^k \in \mathcal{R}^{m_E}$ and $\delta_I^k,\ \hat{\delta}_I^k \in \mathcal{R}^{m_I}$ are error vectors such that

$$\max\{\|\delta_E^k\|, \|\delta_I^k\|, \|\hat{\delta}_E^k\|, \|\hat{\delta}_I^k\|\} \le \epsilon_k / (\sqrt{2} t_k).$$

Step 2. Set $t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$ and $\beta_k = \frac{t_k - 1}{t_{k+1}}$. Compute

$$\widetilde{S}^{k+1} = S^k + \beta_k (S^k - S^{k-1}),\ \tilde{y}_E^{k+1} = y_E^k + \beta_k (y_E^k - y_E^{k-1}),\ \tilde{y}_I^{k+1} = y_I^k + \beta_k (y_I^k - y_I^{k-1}).$$

If the (sparse) Cholesky factorization of $\mathcal{A}_E \mathcal{A}_E^*$ can be computed (only once) at a moderate cost, then (38) can be solved exactly (i.e., $\hat{\delta}_E^k = 0$); otherwise (38) can be solved inexactly to satisfy (39) by an iterative method such as the PCG method. The same remark also applies to the computation of $\hat{y}_I^k, y_I^k, y_E^k$.

(b) From the presentation in Step 1 of Algorithm ABCD-1, it appears that we would need to solve the linear systems involving the matrices $\mathcal{A}_E \mathcal{A}_E^*$ and $\mathcal{A}_I \mathcal{A}_I^* + \mathcal{I}$ twice. In practice, one can often avoid solving the linear systems twice if $\hat{y}_E^k$ and $\hat{y}_I^k$ are already sufficiently accurate approximate solutions for the respective second linear systems. More specifically, suppose that we approximate y_I^k by $\hat{y}_I^k$. Then the residual vector for the second linear system would be given by

$$\delta_I^k = \hat{\delta}_I^k + \mathcal{A}_I (S^k - \widetilde{S}^k).$$

If the condition $\|\delta_I^k\| \le \epsilon_k / (\sqrt{2}\, t_k)$ is satisfied, then we need not solve the second linear system since $\hat{y}_I$ is already a sufficiently accurate solution for the second linear system. Similarly, if we use $\hat{y}_E^k$ to approximate y_E, then the corresponding residual vector would be given by

$$\delta_E^k = \hat{\delta}_E^k + \mathcal{A}_E (\mathcal{A}_I^* (y_I^k - \tilde{y}_I^k) + S^k - \widetilde{S}^k).$$

Again if the condition that $\|\delta_E^k\| \le \epsilon_k / (\sqrt{2} t_k)$ is satisfied, then we can take $y_E^k = \hat{y}_E^k$.

For the second variant of the inexact ABCD method, we apply Algorithm 2 to (D) by expressing it in the form of (21) with $s = 1$ and $(x_0, x_1) = ((Z, v), (S, y_E, y_I))$. In this case, we treat (S, y_E, y_I) as a single block and the corresponding subproblem

in the ABCD method neither admits an analytical solution nor is solvable via a linear system of equations. To solve the subproblem, we use a semismooth Newton-CG (SNCG) algorithm introduced in [38, 37] to solve it inexactly.

The detailed steps of the second variant of the inexact ABCD method are given as follows. We should mention that it is in fact an accelerated version of a majorized SNGG algorithm presented in [37].

Algorithm ABCD-2: **An inexact ABCD method with SNCG for (D).**

Select an initial point $(\widetilde{Z}^1, \tilde{v}^1, \widetilde{S}^1, \tilde{y}_E^1, \tilde{y}_I^1) = (Z^0, v^0, S^0, y_E^0, y_I^0)$ with $(-Z^0, -v^0) \in \mathrm{dom}(\delta_{\mathcal{P}}^*) \times \mathrm{dom}(\delta_{\mathcal{K}}^*)$. Let $\{\epsilon_k\}$ be a nonnegative summable sequence, $t_1 = 1$ and $\tau = 10^{-6}$. Set $k = 1$. Iterate the following steps.

Step 1. Let $\widetilde{R}^k = \mathcal{A}_E^* \tilde{y}_E^k + \mathcal{A}_I^* \tilde{y}_I^k + \widetilde{S}^k + G$. Compute

$$(Z^k, v^k) = \arg\min_{Z,v} \left\{ F(Z, v, \widetilde{S}^k, \tilde{y}_E^k, \tilde{y}_I^k) \right\} = (\Pi_{\mathcal{P}}(\widetilde{R}^k) - \widetilde{R}^k, \Pi_{\mathcal{K}}(g - \tilde{y}_I^k) - (g - \tilde{y}_I^k)),$$

$$(S^k, y_E^k, y_I^k) = \arg\min_{S, y_E, y_I} \left\{ F(Z^k, v^k, S, y_E, y_I) + \frac{\tau}{2}\|y_E - \tilde{y}_E^k\|^2 - \langle \delta_E^k,\, y_E \rangle - \langle \delta_I^k,\, y_I \rangle \right\},$$

where $\delta_E^k \in \mathcal{R}^{m_E}$, $\delta_I^k \in \mathcal{R}^{m_I}$ are error vectors such that

$$\max\{\|\delta_E^k\|, \|\delta_I^k\|\} \le \epsilon_k / (\sqrt{2} t_k).$$

Step 2. Set $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$, $\beta_k = \frac{t_k - 1}{t_{k+1}}$. Compute

$$\widetilde{S}^{k+1} = S^k + \beta_k (S^k - S^{k-1}),\ \tilde{y}_E^{k+1} = y_E^k + \beta_k (y_E^k - y_E^{k-1}),\ \tilde{y}_I^{k+1} = y_I^k + \beta_k (y_I^k - y_I^{k-1}).$$

In our numerical experiments, we always start with the first variant of the ABCD method, and then switch it to the second variant when the convergence speed of the first variant is deemed to be unsatisfactory. As discussed in [38, 37], each iteration of the SNCG algorithm can be quite expensive. In fact, ABCD-1 can achieve a high accuracy efficiently for most of the problems perhaps because it has $O(1/k^2)$ iteration complexity and there is no need to be switched to ABCD-2. However, for some difficult problems, ABCD-1 may stagnate. In this case, ABCD-2 can perform much better since it wisely makes use of the second-order information and it has fewer blocks.

4.1. An efficient iterative method for solving the linear systems. Observe that in both the ABCD method and the APG method to be presented later in the next section for solving (D), we need to solve the following linear systems

$$\mathcal{B} y_I = r, \tag{40}$$

where $\mathcal{B} = (\mathcal{A}_I \mathcal{A}_I^* + \alpha \mathcal{I})$, with $\alpha = 1$ and $\alpha = \frac{1}{3}$ for the ABCD method and APG method, respectively. For the case where the matrix $\mathcal{B}$ and its (sparse) Cholesky factorization (need only to be done once) can be computed at a moderate cost, (40) can be solved efficiently. However, if the Cholesky factorization of $\mathcal{B}$ is not available, then we need an alternative efficient method to deal with (40). In this paper, we solve (40) by a PCG method. In order to speed up the convergence of the CG method, we construct the following preconditioner $\widetilde{\mathcal{B}}$ based on a few leading eigenpairs of $\mathcal{B}$. Specifically, consider the following eigenvalue decomposition,

$$\mathcal{B} = P D P^T, \tag{41}$$

where $P \in \Re^{m_I \times m_I}$ is an orthogonal matrix whose columns are eigenvectors of $\mathcal{B}$, and D is the corresponding diagonal matrix of eigenvalues with the diagonal elements arranged in descending order: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{m_I}$. We choose the preconditioner to be

$$\widetilde{\mathcal{B}} = \sum_{i=1}^{k_I} \lambda_i P_i P_i^T + \lambda_{k_I} \sum_{i=k_I+1}^{m_I} P_i P_i^T, \tag{42}$$

where P_i is the ith column of P, k_I is a small integer such that $1 \leq k_I \ll m_I$, and $\lambda_{k_I} > 0$. Note that we only need to compute (which only needs to be done once) the first k_I eigenvalues of $\mathcal{B}$ and their corresponding eigenvectors. Then

$$\begin{aligned}\widetilde{\mathcal{B}}^{-1} &= \sum_{i=1}^{k_I} \lambda_i^{-1} P_i P_i^T + \lambda_{k_I}^{-1} \sum_{i=k_I+1}^{m_I} P_i P_i^T = \sum_{i=1}^{k_I} \lambda_i^{-1} P_i P_i^T + \lambda_{k_I}^{-1} \left(\mathcal{I} - \sum_{i=1}^{k_I} P_i P_i^T \right) \\ &= \lambda_{k_I}^{-1} \mathcal{I} - \sum_{i=1}^{k_I-1} (\lambda_{k_I}^{-1} - \lambda_i^{-1}) P_i P_i^T.\end{aligned}$$

From the expression of $\widetilde{\mathcal{B}}$, we can see that the overhead cost of applying the preconditioning step $\widetilde{\mathcal{B}}^{-1}v$ for a given v can be kept low compared to the cost of performing $\mathcal{B}v$.

In our numerical experiments, we solve (40) approximately by applying the PCG method with the preconditioner (42) whenever the sparse Cholesky factorization of $\mathcal{B}$ is too expensive to be computed. As we are solving a sequence of linear systems of the form (40), where the right-hand side vector changes moderately from iteration to iteration, we can warm start the PCG method with the previous solution y_I^{k-1} when solving the kth linear system. For the problems which we have tested in our numerical experiments, where the number of linear inequality constraints m_I in (P) can be very large, usually we only need less than ten PCG iterations on the average to solve (40) to the required accuracy. This confirms that the preconditioner (42) is quite effective for the problems under our consideration, and we have thus presented an efficient iterative method to solve the large scale linear systems here.

5. An APG method and an enhanced ARBCG method for solving (D). Instead of the ABCD method, one can also apply the APG and accelerated randomized BCGD methods to solve (D). The details are given in the next two subsections.

5.1. An APG method for solving (D). To apply the APG method, we note that (D) can equivalently be rewritten as follows:

$$\begin{aligned}(\widehat{\mathrm{D}}) \quad \min\ & \widehat{F}(y_E, y_I, Z) \\ &:= -\langle b_E,\, y_E\rangle + \delta^*_{\mathcal{P}}(-Z) + \frac{1}{2}\|\Pi_{\mathcal{S}^n_+}(\mathcal{A}_E^* y_E + \mathcal{A}_I^* y_I + Z + G)\|^2 \\ &\quad + \frac{1}{2}\|g - y_I\|^2 - \frac{1}{2}\|(g - y_I) - \Pi_{\mathcal{K}}(g - y_I)\|^2 - \frac{1}{2}\|G\|^2 - \frac{1}{2}\|g\|^2.\end{aligned}$$

In order to apply the APG method to solve $(\widehat{\mathrm{D}})$, we first derive a majorization of the objective function given the auxiliary iterate $(\tilde{y}_E^k, \tilde{y}_I^k, \widetilde{Z}^k)$. Let $\varphi_1(M, N, Z) = \frac{1}{2}\|\Pi_{\mathcal{S}^n_+}(M + N + Z + G)\|^2$, $\varphi_2(y_I) = \frac{1}{2}\|g - y_I\|^2 - \frac{1}{2}\|(g - y_I) - \Pi_{\mathcal{K}}(g - y_I)\|^2$. It

is known that $\nabla\varphi_i(\cdot)$, $i = 1, 2$, are Lipschitz continuous. Thus we have that for $M = \mathcal{A}_E^* y_E$, $N = \mathcal{A}_I^* y_I$, $\widetilde{M}^k = \mathcal{A}_E^* \tilde{y}_E^k$, $\widetilde{N}^k = \mathcal{A}_I^* \tilde{y}_I^k$,

$$\begin{aligned}\varphi_1(M, N, Z) \le{}& \varphi_1(\widetilde{M}^k, \widetilde{N}^k, \widetilde{Z}^k)\\ &+ \langle \Pi_{\mathcal{S}_+^n}(\widetilde{M}^k + \widetilde{N}^k + \widetilde{Z}^k + G),\, M + N + Z - \widetilde{M}^k - \widetilde{N}^k - \widetilde{Z}^k\rangle\\ &+ \frac{3}{2}\|M - \widetilde{M}^k\|^2 + \frac{3}{2}\|N - \widetilde{N}^k\|^2 + \frac{3}{2}\|Z - \widetilde{Z}^k\|^2,\\ \varphi_2(y_I) \le{}& \varphi_2(\tilde{y}_I^k) - \langle \Pi_{\mathcal{K}}(g - \tilde{y}_I^k),\, y_I - \tilde{y}_I^k\rangle + \frac{1}{2}\|y_I - \tilde{y}_I^k\|^2.\end{aligned}$$

From here, we get

$$\begin{aligned}&\widehat{F}(y_E, y_I, Z) - \widehat{F}(\tilde{y}_E^k, \tilde{y}_I^k, \widetilde{Z}^k)\\ &\quad\le \delta_{\mathcal{P}}^*(-Z) - \langle b_E,\, y_E - \tilde{y}_E^k\rangle + \langle X^k,\, \mathcal{A}_E^* y_E - \mathcal{A}_E^* \tilde{y}_E^k\rangle\\ &\qquad + \langle X^k,\, \mathcal{A}_I^* y_I - \mathcal{A}_I^* \tilde{y}_I^k\rangle + \langle X^k,\, Z - \widetilde{Z}^k\rangle - \langle s^k,\, y_I - \tilde{y}_I^k\rangle + \frac{3}{2}\|\mathcal{A}_E^* y_E - \mathcal{A}_E^* \tilde{y}_E^k\|^2\\ &\qquad + \frac{3}{2}\|\mathcal{A}_I^* y_I - \mathcal{A}_I^* \tilde{y}_I^k\|^2 + \frac{3}{2}\|Z - \widetilde{Z}^k\|^2 + \frac{1}{2}\|y_I - \tilde{y}_I^k\|^2\\ &\quad:= \phi^k(y_E, y_I, Z),\end{aligned}$$

where $X^k = \Pi_{\mathcal{S}_+^n}(\mathcal{A}_E^* \tilde{y}_E^k + \mathcal{A}_I^* \tilde{y}_I^k + \widetilde{Z}^k + G)$, $s^k = \Pi_{\mathcal{K}}(g - \tilde{y}_I^k)$.

At each iteration of the APG method [1, 35] applied to $(\widehat{\mathrm{D}})$, we need to solve the following minimization subproblem at the kth iteration:

$$\min_{y_E, y_I, Z} \Big\{ \phi^k(y_E, y_I, Z) \Big\}$$

whose optimal solution is given by

$$\text{(43)} \qquad \begin{cases} y_E^k = (\mathcal{A}_E \mathcal{A}_E^*)^{-1}\big(\frac{1}{3}(b_E - \mathcal{A}X^k) + \mathcal{A}_E \mathcal{A}_E^* \tilde{y}_E^k\big),\\ y_I^k = (\mathcal{A}_I \mathcal{A}_I^* + \frac{1}{3}\mathcal{I}_{m_I})^{-1}\big(\frac{1}{3}(s^k - \mathcal{A}X^k + \tilde{y}_I^k) + \mathcal{A}_I \mathcal{A}_I^* \tilde{y}_I^k\big),\\ Z^k = \Pi_{\mathcal{P}}(\frac{1}{3}X^k - \widetilde{Z}^k) - (\frac{1}{3}X^k - \widetilde{Z}^k). \end{cases}$$

With the computed iterate (y_E^k, y_I^k, Z^k), we update the auxiliary iterate

$$(\tilde{y}_E^{k+1}, \tilde{y}_I^{k+1}, \widetilde{Z}^{k+1})$$

similarly to Step 2 of Algorithm ABCD-1.

5.2. An enhanced ARBCG method for solving (D). Next we describe the eARBCG method for solving (D). Our algorithm is an application of Algorithm 3 in [17], which is a simplified block framework of the APPROX algorithm presented in [10] for the sake of numerical comparison, and the steps are given as follows.

Algorithm eARBCG: **An enhanced ARBCG method for (D).**

We use the notation in (4) with $q = 4$ and $x = ((Z, v), S, y_E, y_I)$. Select an initial point $\widetilde{x}^1 = x^1$. Set $\alpha_0 = 1/q$ and $k = 1$. Iterate the following steps.

Step 1. Compute $\alpha_k = \frac{1}{2}\Big(\sqrt{\alpha_{k-1}^4 + 4\alpha_{k-1}^2} - \alpha_{k-1}^2\Big)$.

Step 2. Compute $\widehat{x}^{k+1} = (1 - \alpha_k)x^k + \alpha_k\widetilde{x}^k$.

Step 3. Choose $i_k \in \{1, \ldots, q\}$ uniformly at random and compute

$$\widetilde{x}_{i_k}^{k+1} = \mathrm{argmin}\Big\{\langle \nabla_{i_k}\zeta(\widehat{x}^k),\, x_{i_k} - \widehat{x}_{i_k}^k\rangle + \frac{q\alpha_k}{2}\|x_{i_k} - \widetilde{x}_{i_k}^k\|_{\mathcal{T}_{i_k}}^2 + \theta_{i_k}(x_{i_k})\Big\},$$

where $\mathcal{T}_1$, $\mathcal{T}_2$ are identity operators, and $\mathcal{T}_3 = \mathcal{A}_E\mathcal{A}_E^*$ and $\mathcal{T}_4 = \mathcal{A}_I\mathcal{A}_I^* + \mathcal{I}$. Here $\theta_1(x_1) = \delta_{\mathcal{P}}^*(-Z) + \delta_{\mathcal{K}}^*(-v)$, $\theta_2(x_2) = \delta_{\mathcal{S}_+^n}(S)$, $\theta_3(x_3) = 0 = \theta_4(x_4)$, and ζ is the smooth part of F in (D). Set $\widetilde{x}_i^{k+1} = \widetilde{x}_i^k$ for all $i \neq i_k$ and

$$x_i^{k+1} = \begin{cases} \widehat{x}_i^k + q\alpha_k(\widetilde{x}_i^{k+1} - \widetilde{x}_i^k) & \text{if } i = i_k, \\ \widehat{x}_i^k & \text{if } i \neq i_k. \end{cases}$$

Note that in the original ARBCG algorithm in [10] and [17], each block needs to be solved exactly. Our enhancement to the algorithm is in using an efficient iterative method for solving the large linear systems inexactly with the same rule as introduced in section 4. Indeed the practical performance of the eARBCG is much better than the original version since it is numerically tough to get an exact solution for a large scale linear system. However, we should note that although the nonconvergence of the eARBCG method is never observed for the problems tested in our numerical experiments, the theoretical convergence of the eARBCG has yet to be established, which we leave as a future research topic.

6. Numerical experiments. In our numerical experiments, we test the algorithms designed in the last two sections to the LSSDP problem (P) by taking $G = -C, g = 0$ for the data arising from various classes of SDP problems of the form given in (1). Specifically, the LSSDP problem corresponds to the first subproblem (2) of the PPA for solving (1) by setting $k = 0$, $X^0 = 0$, $s^0 = 0$, and $\sigma_0 = 1$.

6.1. SDP problem sets. Now we describe the classes of SDP problems we considered in our numerical experiments.

(i) SDP problems coming from the relaxation of a binary integer nonconvex quadratic (BIQ) programming:

$$\min\left\{\frac{1}{2}x^TQx + \langle c,\, x\rangle \mid x \in \{0,1\}^{n-1}\right\}. \tag{44}$$

This problem has shown in [3] that under some mild assumptions, it can equivalently be reformulated as the following completely positive programming (CPP) problem:

$$\min\left\{\frac{1}{2}\langle Q,\, X_0\rangle + \langle c,\, x\rangle \mid \mathrm{diag}(X_0) = x, X = [X_0, x; x^T, 1] \in \mathcal{C}_{pp}^n\right\}, \tag{45}$$

where $\mathcal{C}_{pp}^n$ denotes the n-dimensional completely positive cone. It is well known that even though $\mathcal{C}_{pp}^n$ is convex, it is computationally intractable. To solve the CPP prob-

lem, one can relax $\mathcal{C}^n_{pp}$ to $\mathcal{S}^n_+ \cap \mathcal{N}$, and the relaxed problem has the form (1)

$$\begin{array}{ll} \min & \frac{1}{2}\langle Q,\, X_0\rangle + \langle c,\, x\rangle \\ \text{s.t.} & \mathrm{diag}(X_0) - x = 0,\ \alpha = 1, \quad X = \left[\begin{array}{cc} X_0 & x \\ x^T & \alpha \end{array}\right] \in \mathcal{S}^n_+, \quad X \in \mathcal{P}, \end{array} \tag{46}$$

where the polyhedral cone $\mathcal{P} = \{X \in \mathcal{S}^n \mid X \geq 0\}$. In our numerical experiments, the test data for Q and c are taken from Biq Mac Library maintained by Wiegele, which is available at http://biqmac.uni-klu.ac.at/biqmaclib.html.

(ii) SDP problems arising from the relaxation of maximum stable set problems. Given a graph G with edge set $\mathcal{E}$, the SDP relaxation $\theta_+(G)$ of the maximum stable set problem is given by

$$\theta_+(G) = \max\{\langle ee^T,\, X\rangle \mid \langle E_{ij},\, X\rangle = 0, (i,j) \in \mathcal{E},\, \langle I,\, X\rangle = 1,\, X \in \mathcal{S}^n_+,\, X \in \mathcal{P}\} \tag{47}$$

where $E_{ij} = e_i e_j^{\mathrm{T}} + e_j e_i^{\mathrm{T}}$ and e_i denotes the ith column of the identity matrix, $\mathcal{P} = \{X \in \mathcal{S}^n \mid X \geq 0\}$. In our numerical experiments, we test the graph instances G considered in [30], [31], and [32].

(iii) SDP relaxation for computing lower bounds for quadratic assignment problems (QAPs). Given matrices $A, B \in \mathcal{S}^n$, the QAP is given by

$$v^*_{\mathrm{QAP}} := \min\{\langle X, AXB\rangle \,:\, X \in \mathit{\Pi}\},$$

where $\mathit{\Pi}$ is the set of $n \times n$ permutation matrices.

For a matrix $X = [x_1, \ldots, x_n] \in \Re^{n\times n}$, we will identify it with the n^2-vector $x = [x_1; \ldots; x_n]$. For a matrix $Y \in R^{n^2\times n^2}$, we let Y^{ij} be the $n\times n$ block corresponding to $x_i x_j^T$ in the matrix xx^T. In [23], it is shown that v^*_{QAP} is bounded below by the following number generated from the SDP relaxation:

$$\begin{array}{rll} v := \min & \langle B \otimes A,\, Y\rangle & \\ \text{s.t.} & \sum_{i=1}^n Y^{ii} = I,\ \langle I,\, Y^{ij}\rangle = \delta_{ij} & \forall\, 1 \leq i \leq j \leq n, \\ & \langle E,\, Y^{ij}\rangle = 1 & \forall\, 1 \leq i \leq j \leq n, \quad Y \in \mathcal{S}^n_+,\ Y \in \mathcal{P}, \end{array} \tag{48}$$

where the sign "$\otimes$" stands for the Kronecker product, E is the matrix of ones, and $\delta_{ij} = 1$ if $i = j$, and 0 otherwise, $\mathcal{P} = \{X \in \mathcal{S}^{n^2} \mid X \geq 0\}$. In our numerical experiments, the test instances (A, B) are taken from the QAP Library [12].

(iv) SDP relaxations of clustering problems (RCPs) described in [22, (13)]:

$$\min\Big\{\langle -W,\, X\rangle \mid Xe = e, \langle I,\, X\rangle = K, X \in \mathcal{S}^n_+, X \in \mathcal{P}\Big\}, \tag{49}$$

where W is the so-called affinity matrix whose entries represent the pairwise similarities of the objects in the dataset, e is the vector of ones, and K is the number of clusters, $\mathcal{P} = \{X \in \mathcal{S}^n \mid X \geq 0\}$. All the data sets we tested are from the UCI Machine Learning Repository (available at http://archive.ics.uci.edu/ml/datasets.html). For some large data instances, we only select the first n rows. For example, the original data instance "`spambase`" has 4601 rows; we select the first 1500 rows to obtain the test problem "`spambase-large.2`" for which the number "`2`" means that there are $K = 2$ clusters.

(v) SDP problems arising from the SDP relaxation of frequency assignment problems (FAPs) [8]. Given a network represented by a graph G with edge set $\mathcal{E}$ and an edge-weight matrix W, a certain type of FAP on G can be relaxed into the following

SDP (see [4, (5)]):

$$\begin{array}{ll} \max & \langle(\frac{k-1}{2k})Lap(W) - \frac{1}{2}\mathrm{Diag}(We),\, X\rangle \\ \text{s.t.} & \mathrm{diag}(X) = e,\ \ X \in \mathcal{S}^n_+,\ \ X \in \mathcal{P}, \end{array} \tag{50}$$

where $Lap(W) := \mathrm{Diag}(We) - W$ is the Laplacian matrix, $e \in \Re^n$ is the vector of all ones, and $\mathcal{P} = \{X \in \mathcal{S}^n \mid L \le X \le U\}$,

$$L_{ij} = \begin{cases} -\frac{1}{k-1} & \forall (i,j) \in \mathcal{E}, \\ -\infty & \text{otherwise}, \end{cases} \quad U_{ij} = \begin{cases} -\frac{1}{k-1} & \forall (i,j) \in \mathcal{U}, \\ \infty & \text{otherwise} \end{cases}$$

with $k > 1$ being a given integer and $\mathcal{U}$ is a given subset of $\mathcal{E}$.

(vi) For the SDP problems described in (46) arising from relaxing the BIQ problems, in order to get tighter bounds, we may add in some valid inequalities to get the following problems:

$$\begin{array}{ll} \min & \frac{1}{2}\langle Q,\, Y\rangle + \langle c,\, x\rangle \\ \text{s.t.} & \mathrm{diag}(Y) - x = 0,\ \alpha = 1,\ X = \begin{bmatrix} Y & x \\ x^T & \alpha \end{bmatrix} \in \mathcal{S}^n_+,\ X \in \mathcal{P}, \\ & 0 \le -Y_{ij} + x_i \le 1,\ 0 \le -Y_{ij} + x_j \le 1,\ -1 \le Y_{ij} - x_i - x_j \le 0, \\ & \forall\ 1 \le i < j,\ j \le n-1, \end{array} \tag{51}$$

where $\mathcal{P} = \{X \in \mathcal{S}^n \mid X \ge 0\}$. For convenience, we call the problem in (51) an extended BIQ problem. Note that the last set of inequalities in (51) are obtained from the valid inequalities $0 \le x_i(1-x_j) \le 1$, $0 \le x_j(1-x_i) \le 1$, $0 \le (1-x_i)(1-x_j) \le 1$ when x_i, x_j are binary variables.

6.2. Numerical results. In this section, we compare the performance of the ABCD, APG, eARBCG, and BCD methods for solving the LSSDP (P). Note that the BCD method follows the template described in (5) with $q = 4$ and $x = ((Z, v), S, y_E, y_I)$. All our computational results are obtained by running MATLAB on a workstation (20-core, Intel Xeon E5-2670 v2 @ 2.5 GHz, 64 GB RAM).

In our numerical experiments, for problem (P), we assume that $\gamma := \max\{\|G\|, \|g\|\} \le 1$, otherwise we can solve an equivalent rescaled problem

$$(\overline{\mathrm{P}}) \quad \begin{array}{ll} \min & \frac{1}{2}\|X - \bar{G}\|^2 + \frac{1}{2}\|s - \bar{g}\|^2 \\ \text{s.t.} & \mathcal{A}_E(X) = \bar{b}_E,\ \mathcal{A}_I X - s = 0,\ X \in \mathcal{S}^n_+,\ X \in \bar{\mathcal{P}},\ s \in \bar{\mathcal{K}}, \end{array}$$

where $\bar{G} = \frac{G}{\gamma}$, $\bar{g} = \frac{g}{\gamma}$, $\bar{b}_E = \frac{b_E}{\gamma}$, $\bar{\mathcal{P}} = \{X \mid \gamma X \in \mathcal{P}\}$, $\bar{\mathcal{K}} = \{s \mid \gamma s \in \mathcal{K}\}$. Note that (X, s) is a solution to (P) if and only if $(\frac{X}{\gamma}, \frac{s}{\gamma})$ is a solution to $(\overline{\mathrm{P}})$.

Note that under a suitable Slater's condition, the KKT conditions for (P) and (D) are given as follows:

$$\begin{aligned} A_E X = b_E,\ \ \mathcal{A}_I X - s = 0,\ \ X - Y = 0, \\ X = \Pi_{\mathcal{S}^n_+}(\mathcal{A}_E^* y_E + \mathcal{A}_I^* y_I + Z + G), \\ Y = \Pi_{\mathcal{P}}(\mathcal{A}_E^* y_E + \mathcal{A}_I^* y_I + S + G), \\ s = \Pi_{\mathcal{K}}(g - y_I). \end{aligned} \tag{52}$$

Thus we measure the accuracy of an approximate optimal solution (Z, v, S, y_E, y_I) for

TABLE 1
Number of problems which are solved to the accuracy of 10^{-6} *in* η.

Problem set (No.) \ solver	ABCD	APG	eARBCG	BCD
θ_+ (64)	64	64	64	11
FAP (7)	7	7	7	7
QAP (95)	95	95	24	0
BIQ (165)	165	165	165	65
RCP (120)	120	120	120	108
exBIQ (165)	165	141	165	10
Total (616)	616	592	545	201

TABLE 2
Number of problems which are solved to the accuracy of 10^{-6}, 10^{-7}, *and* 10^{-8}, *in* η *by the ABCD method.*

Problem set (No.) \ ε	10^{-6}	10^{-7}	10^{-8}
θ_+ (64)	64	58	52
FAP (7)	7	7	7
QAP (95)	95	95	95
BIQ (165)	165	165	165
RCP (120)	120	120	118
exBIQ (165)	165	165	165
Total (616)	616	610	602

(**D**) by using the following relative residual:

$$\eta = \max\{\eta_1, \eta_2, \eta_3\}, \tag{53}$$

where $\eta_1 = \frac{\|b_E - \mathcal{A}_E X\|}{1+\|b_E\|}$, $\eta_2 = \frac{\|X-Y\|}{1+\|X\|}$, $\eta_3 = \frac{\|s-\mathcal{A}_I X\|}{1+\|s\|}$, $X = \Pi_{S^n_+}(\mathcal{A}_E^* y_E + \mathcal{A}_I^* y_I + Z + G)$, $Y = \Pi_{\mathcal{P}}(\mathcal{A}_E^* y_E + \mathcal{A}_I^* y_I + S + G)$, $s = \Pi_{\mathcal{K}}(g - y_I)$. Additionally, we compute the relative gap defined by

$$\eta_g = \frac{\frac{1}{2}\|X-G\|^2 + \frac{1}{2}\|s-g\|^2 - F(Z, v, S, y_E, y_I)}{1 + \frac{1}{2}\|X-G\|^2 + \frac{1}{2}\|s-g\|^2 + |F(Z, v, S, y_E, y_I)|}. \tag{54}$$

Let $\varepsilon > 0$ be a given accuracy tolerance. We terminate the ABCD, APG, eARBCG, and BCD methods when $\eta < \varepsilon$.

Table 1 shows the number of problems that have been successfully solved to the accuracy of 10^{-6} in η by each of the four solvers: ABCD, APG, eARBCG, and BCD, with the maximum number of iterations set at 25000.[2] As can be seen, only ABCD can solve all the problems to the desired accuracy of 10^{-6}. The performance of the BCD method is especially poor, and it can only solve 201 problems out of 616 to the desired accuracy.

Table 2 shows the number of problems that have been successfully solved to the accuracy of 10^{-6}, 10^{-7}, and 10^{-8} in η by the solver ABCD, with the maximum number of iterations set at 25000. As can be seen, ABCD can even solve almost all the problems to the high accuracy of 10^{-8}.

Table 3 compares the performance of the solvers ABCD, APG, and eARBCG on a subset of the 616 tested LSSDP problems using the tolerance $\varepsilon = 10^{-6}$. The full table for all the 616 problems is available at http://www.math.nus.edu.sg/∼mattohkc/publist.html. The first three columns of Table 3 give the problem name, the dimensions of the variables y_E (m_E) and y_I (m_I), the size of the matrix C (n_s) in (P),

[2]The maximum number of iterations for eARBCG is $25000q$, where q is the number of block variables, i.e., $q = 4$ for extended BIQ problems and $q = 3$ otherwise, since eARBCG only updates one block variable at each iteration.

TABLE 3

Performance of ABCD, APG, and eARBCG on θ_+, FAP, QAP, BIQ, RCP, and extended BIQ problems ($\varepsilon = 10^{-6}$).

			Iteration			η			η_g			Time		
Problem	$m_E;m_I$	n_s	ABCD	APG	eARBCG	ABCD	APG	eARBCG	ABCD	APG	eARBCG	ABCD	APG	eARBCG
theta8	7905;0	400	0;1689	2583	7225	9.8-7	9.9-7	9.9-7	-1.3-7	-1.9-9	-3.2-7	3:24	5:20	11:02
theta82	23872;0	400	0;1405	2227	6302	9.9-7	9.9-7	9.7-7	-1.5-7	4.3-9	-1.4-7	2:49	4:52	9:16
theta83	39862;0	400	0;1672	2157	6242	9.9-7	9.9-7	9.9-7	1.6-8	5.8-9	-1.2-7	4:00	4:36	9:30
theta10	12470;0	500	0;1820	2629	7839	9.9-7	9.9-7	9.9-7	-9.2-8	1.9-10	-2.5-7	7:19	9:08	18:26
theta102	37467;0	500	0;1705	2266	6923	9.9-7	9.9-7	9.9-7	6.8-8	1.9-8	-1.5-7	7:04	8:19	16:25
theta103	62516;0	500	0;1641	2228	6729	9.9-7	9.9-7	9.8-7	5.9-8	1.8-8	-1.3-7	6:50	8:31	17:01
theta104	87245;0	500	25;1483	2235	6726	9.9-7	9.9-7	9.9-7	-1.8-7	1.6-8	-1.1-7	7:16	7:41	16:43
theta12	17979;0	600	0;1961	2644	8728	9.9-7	9.8-7	9.7-7	-1.4-7	5.8-8	-2.6-7	12:28	14:40	27:32
theta123	90020;0	600	0;1743	2366	7160	9.9-7	9.9-7	9.9-7	4.4-8	2.0-8	-1.3-7	11:27	13:37	27:35
theta162	127600;0	800	0;2149	2670	7508	9.9-7	9.8-7	9.9-7	4.0-8	2.3-8	-1.3-7	29:59	30:58	54:54
hamming-9-8	2305;0	512	0;382	615	6304	2.5-7	5.3-7	8.4-7	-1.3-6	-2.6-7	-3.0-7	57	1:23	13:02
hamming-10-2	23041;0	1024	89;254	817	5211	8.2-7	7.6-7	6.4-7	4.2-6	7.5-8	-5.4-8	4:39	6:28	56:51
hamming-9-5-6	53761;0	512	0;253	456	1629	8.0-7	1.9-7	3.6-7	5.6-7	-1.2-8	-5.1-8	27	37	2:40
brock400-1	20078;0	400	0;1586	2347	6553	9.9-7	9.9-7	9.6-7	-6.0-8	-3.4-9	-1.9-7	1:35	2:17	6:21
G43	9991;0	1000	0;3350	3949	15965	9.9-7	9.9-7	9.9-7	-5.2-7	-4.6-8	-2.8-7	26:40	30:07	2:11:03
G44	9991;0	1000	0;3351	4039	15965	9.9-7	9.8-7	9.9-7	-4.1-7	6.1-8	-2.9-7	28:02	31:43	2:34:37
G45	9991;0	1000	0;3220	3868	15678	9.9-7	9.9-7	9.9-7	-4.6-7	-1.1-7	-2.0-7	25:28	29:15	2:51:51
G46	9991;0	1000	0;3144	3647	15630	9.9-7	9.7-7	9.8-7	-4.7-7	1.9-7	-1.9-7	24:59	27:29	2:55:29
G47	9991;0	1000	0;3378	4025	15607	9.9-7	9.8-7	9.9-7	-5.5-7	7.8-9	-2.3-7	26:55	32:19	1:28:28
G51	5910;0	1000	0;11273	16058	41754	8.6-7	9.9-7	9.9-7	-5.2-7	-1.9-8	-9.1-7	1:39:35	2:18:14	4:03:33
G52	5917;0	1000	0;7955	18194	36594	9.9-7	9.8-7	9.9-7	-5.0-7	7.3-9	-9.3-7	1:09:04	3:03:03	3:43:49
G53	5915;0	1000	0;10887	16420	39485	9.6-7	9.5-7	9.8-7	-5.6-7	-1.6-8	-7.3-7	1:41:55	3:48:26	4:55:22
G54	5917;0	1000	0;10883	13798	41754	9.5-7	9.9-7	9.9-7	-4.9-7	1.5-8	-9.2-7	2:46:39	4:09:19	7:36:36
1dc.512	9728;0	512	0;3755	5543	17564	9.9-7	9.5-7	9.8-7	-3.6-7	-1.6-8	-3.9-7	6:21	17:53	25:12
1et.512	4033;0	512	0;3265	3181	11786	9.9-7	9.9-7	9.9-7	-3.8-7	-1.8-8	-5.0-7	5:33	11:14	16:14
1tc.512	3265;0	512	0;5590	10046	26847	9.9-7	9.8-7	9.8-7	-6.8-7	1.3-8	-4.0-7	9:39	33:42	37:02
2dc.512	54896;0	512	16;2871	3468	8987	9.5-7	9.7-7	9.8-7	-3.8-7	7.2-9	-2.4-7	5:24	12:16	12:47
1zc.512	6913;0	512	0;3370	2666	18104	9.9-7	9.1-7	9.4-7	-3.2-7	-8.9-7	-3.8-7	4:59	7:27	24:15
1dc.1024	24064;0	1024	0;3074	5976	16655	9.9-7	9.5-7	9.9-7	-2.7-7	3.4-8	-2.6-7	28:57	2:17:15	1:53:08
1et.1024	9601;0	1024	0;5119	7874	26394	9.9-7	9.9-7	9.9-7	-3.7-7	-7.3-8	4.4-7	46:55	2:55:02	3:42:44
1tc.1024	7937;0	1024	0;6282	10496	26839	9.9-7	9.9-7	9.9-7	-6.4-7	-9.6-10	-1.8-7	1:00:21	3:46:21	5:03:53
1zc.1024	16641;0	1024	0;2281	1724	13029	9.7-7	9.9-7	9.7-7	-1.3-6	-3.5-7	6.2-8	18:16	32:11	2:29:37
2dc.1024	169163;0	1024	18;2029	4910	11689	9.9-7	9.7-7	9.7-7	-2.9-7	2.1-9	-3.2-7	19:23	1:46:34	46:54
1dc.2048	58368;0	2048	0;2700	5447	17472	9.9-7	9.9-7	9.9-7	-8.5-8	-1.6-7	6.7-7	2:54:31	11:05:59	7:38:17
1et.2048	22529;0	2048	0;5267	14243	30322	9.9-7	9.3-7	9.7-7	-4.2-7	-1.9-8	9.7-7	5:13:18	21:40:08	14:08:56
1tc.2048	18945;0	2048	0;7737	15235	65261	9.8-7	9.8-7	9.4-7	-5.4-7	-5.1-8	1.4-6	7:34:43	22:17:52	31:38:14
2dc.2048	504452;0	2048	0;5392	6996	14000	9.9-7	9.8-7	9.9-7	8.8-9	1.6-9	-4.3-7	5:38:09	6:15:33	6:56:29

TABLE 3. (*cont.*)

			Iteration			η			η_g			Time		
Problem	$m_E;m_I$	n_s	ABCD	APG	eARBCG	ABCD	APG	eARBCG	ABCD	APG	eARBCG	ABCD	APG	eARBCG
1zc.2048	39425;0	2048	0;2928	1730	32486	9.2-7	9.4-7	8.7-7	-9.3-8	-6.0-7	6.6-7	4:16:36	1:38:33	21:51:00
fap08	120;0	120	0;47	111	223	8.4-7	9.5-7	9.7-7	-6.5-8	2.0-7	2.6-6	00	01	01
fap09	174;0	174	0;54	111	258	9.4-7	9.6-7	9.1-7	1.4-7	1.9-7	2.1-6	01	01	02
fap10	183;0	183	0;45	67	254	8.4-7	7.9-7	6.2-7	7.5-8	1.2-7	3.2-7	01	01	02
fap11	252;0	252	0;53	76	219	8.6-7	9.9-7	9.2-7	1.8-7	2.1-7	1.1-6	01	01	04
fap12	369;0	369	0;58	94	258	7.8-7	9.7-7	7.1-7	-9.2-8	4.2-7	1.0-6	03	07	15
fap25	2118;0	2118	0;46	95	218	9.2-7	8.1-7	9.0-7	-3.1-8	5.1-8	1.1-6	2:44	11:02	12:26
fap36	4110;0	4110	24;48	129	250	8.4-7	9.7-7	9.9-7	1.4-9	2.7-8	1.3-6	1:15:31	1:38:55	1:40:50
bur26a	1051;0	676	151;27	3110	43852	9.9-7	9.9-7	9.9-7	-5.7-6	-8.0-6	-5.6-6	6:14	17:12	1:33:31
bur26b	1051;0	676	150;27	3083	43600	9.6-7	9.9-7	9.9-7	-5.7-6	-7.9-6	-5.5-6	6:05	17:05	1:38:59
bur26c	1051;0	676	146;27	3245	45432	9.3-7	9.9-7	9.9-7	-6.9-6	-9.2-6	-6.3-6	6:18	18:47	2:08:59
bur26d	1051;0	676	172;27	3188	44790	9.6-7	9.9-7	9.9-7	-6.1-6	-8.7-6	-6.0-6	7:12	18:00	2:30:30
bur26e	1051;0	676	153;27	3178	44615	9.6-7	9.9-7	9.9-7	-6.2-6	-8.7-6	-6.0-6	8:01	18:23	2:49:49
bur26f	1051;0	676	127;27	3142	44272	9.9-7	9.9-7	9.9-7	-5.7-6	-8.6-6	-5.9-6	4:10	17:37	2:48:14
bur26g	1051;0	676	149;27	3012	42291	9.8-7	9.9-7	9.9-7	-6.3-6	-8.4-6	-5.9-6	7:18	17:23	2:06:01
bur26h	1051;0	676	126;27	3004	42050	9.7-7	9.9-7	9.9-7	-6.1-6	-8.3-6	-5.9-6	4:09	15:40	2:27:21
chr20a	628;0	400	242;40	5548	75000	9.4-7	9.9-7	1.3-6	-1.7-5	-2.7-5	-2.1-5	2:44	10:09	1:39:07
chr20b	628;0	400	203;33	4727	74892	9.9-7	9.9-7	9.9-7	-1.5-5	-2.5-5	-1.7-5	2:41	8:34	1:32:59
chr20c	628;0	400	223;48	6738	75000	9.7-7	9.9-7	1.6-6	-2.1-5	-3.4-5	-3.0-5	3:25	12:27	59:58
chr22a	757;0	484	390;42	4871	75000	9.4-7	9.9-7	1.0-6	-1.3-5	-2.1-5	-1.5-5	6:10	13:04	1:44:43
chr22b	757;0	484	224;40	4430	68291	9.9-7	9.9-7	9.9-7	-1.4-5	-2.0-5	-1.4-5	3:31	11:51	1:44:34
chr25a	973;0	625	234;36	4750	75000	9.9-7	9.9-7	1.0-6	-1.3-5	-2.3-5	-1.6-5	8:08	21:30	4:00:41
esc32a	1582;0	1024	155;48	4950	75000	9.7-7	9.7-7	1.2-6	-1.2-5	-1.9-5	-1.3-5	8:41	52:28	4:35:48
esc32b	1582;0	1024	152;39	6966	75000	9.0-7	9.9-7	1.2-6	-1.4-5	-2.0-5	-1.5-5	10:26	1:30:30	4:50:54
esc32c	1582;0	1024	177;77	10626	75000	9.7-7	9.9-7	3.0-6	-3.4-5	-4.6-5	-5.4-5	13:49	2:48:01	4:58:03
esc32d	1582;0	1024	184;71	12520	75000	9.8-7	9.9-7	2.7-6	-3.1-5	-4.1-5	-4.7-5	28:56	3:05:06	5:16:11
esc32e	1582;0	1024	272;38	3245	45376	9.2-7	9.9-7	9.9-7	-2.7-7	-5.5-6	-3.8-6	18:12	23:50	3:32:35
esc32f	1582;0	1024	272;38	3245	45376	9.2-7	9.9-7	9.9-7	-2.7-7	-5.5-6	-3.8-6	18:51	25:13	3:51:30
esc32g	1582;0	1024	113;46	4071	63055	9.5-7	9.9-7	9.9-7	-1.2-6	-9.5-6	-6.7-6	13:00	31:09	5:41:29
esc32h	1582;0	1024	145;66	9791	75000	9.5-7	9.9-7	2.7-6	-3.0-5	-4.4-5	-5.0-5	20:40	1:41:10	10:49:54
had20	628;0	400	153;41	5269	75000	9.6-7	9.9-7	1.2-6	-9.8-6	-1.4-5	-1.0-5	1:38	8:02	41:08
kra30a	1393;0	900	179;42	6177	75000	9.9-7	9.9-7	1.5-6	-3.8-6	-3.6-5	-2.9-5	5:55	57:39	3:51:14
kra30b	1393;0	900	240;42	6216	75000	9.9-7	9.9-7	1.5-6	-2.6-5	-3.7-5	-3.0-5	19:19	1:11:10	3:58:05
kra32	1582;0	1024	336;44	6190	75000	9.6-7	9.9-7	1.4-6	-1.9-5	-2.9-5	-2.4-5	40:34	1:47:07	6:24:11
lipa20a	628;0	400	125;40	2273	29318	9.3-7	9.9-7	9.9-7	-1.6-6	-3.6-6	-2.4-6	1:04	3:43	20:31
lipa20b	628;0	400	164;38	4268	66682	9.4-7	9.9-7	9.9-7	-5.9-6	-1.1-5	-7.2-6	1:24	3:30	47:49
lipa30a	1393;0	900	193;37	1980	22720	9.8-7	9.9-7	9.9-7	-8.3-7	-2.3-6	-1.5-6	9:28	10:01	1:30:56
lipa30b	1393;0	900	193;29	3441	50659	9.4-7	9.9-7	9.9-7	-4.4-6	-7.1-6	-4.8-6	11:30	17:46	3:52:30
lipa40a	2458;0	1600	83; 9	1646	18212	9.7-7	9.9-7	9.9-7	-6.2-7	-1.8-6	-1.2-6	21:13	37:53	8:40:03

TABLE 3. (*cont.*)

			Iteration			η			η_g			Time		
Problem	$m_E;m_I$	n_s	ABCD	APG	eARBCG	ABCD	APG	eARBCG	ABCD	APG	eARBCG	ABCD	APG	eARBCG
lipa40b	2458;0	1600	274;33	3460	51010	9.6-7	9.9-7	9.9-7	-3.7-6	-7.2-6	-4.8-6	1:11:40	2:20:07	14:32:56
nug20	628;0	400	159;43	6400	75000	9.8-7	9.9-7	1.5-6	-1.9-5	-2.7-5	-2.3-5	1:49	9:28	1:35:21
nug21	691;0	441	204;45	6713	75000	9.9-7	9.9-7	1.6-6	-2.4-5	-3.0-5	-2.6-5	5:24	13:12	1:58:15
nug22	757;0	484	409;48	7196	75000	9.7-7	9.9-7	1.8-6	-2.5-5	-3.6-5	-3.2-5	8:03	17:04	2:19:02
nug24	898;0	576	181;43	6400	75000	9.6-7	9.9-7	1.5-6	-1.9-5	-2.7-5	-2.2-5	6:25	23:30	1:45:44
nug25	973;0	625	240;42	6250	75000	9.8-7	9.9-7	1.5-6	-1.8-5	-2.5-5	-2.0-5	3:39	30:34	2:06:53
nug27	1132;0	729	287;44	5217	75000	9.9-7	9.9-7	1.1-6	-1.9-5	-2.7-5	-2.0-5	5:42	36:24	2:57:17
nug28	1216;0	784	277;44	5048	75000	9.8-7	9.9-7	1.1-6	-1.8-5	-2.5-5	-1.8-5	7:23	38:44	3:42:27
nug30	1393;0	900	210;41	6156	75000	9.6-7	9.9-7	1.4-6	-1.6-5	-2.2-5	-1.8-5	9:58	1:11:51	7:20:31
rou20	628;0	400	156;36	5308	75000	9.9-7	9.9-7	1.2-6	-1.3-5	-1.6-5	-1.2-5	1:21	8:50	1:39:12
scr20	628;0	400	198;39	5555	75000	9.8-7	9.9-7	1.3-6	-1.7-5	-2.9-5	-2.2-5	1:49	5:03	1:32:58
ste36a	1996;0	1296	211;37	5494	75000	9.8-7	9.9-7	1.2-6	-1.7-5	-2.5-5	-1.9-5	40:38	1:11:46	8:28:01
ste36b	1996;0	1296	231;48	6918	75000	9.6-7	9.9-7	1.7-6	-2.9-5	-3.9-5	-3.5-5	1:05:38	1:57:46	9:39:13
ste36c	1996;0	1296	225;39	5779	75000	9.9-7	9.9-7	1.3-6	-1.8-5	-2.7-5	-2.1-5	50:54	2:46:30	14:10:06
tai20a	628;0	400	164;31	5072	75000	9.8-7	9.9-7	1.1-6	-1.1-5	-1.5-5	-1.1-5	1:23	8:37	1:35:41
tai20b	628;0	400	180;45	7271	75000	9.9-7	9.9-7	1.8-6	-2.8-5	-3.9-5	-3.6-5	3:26	5:52	1:27:02
tai25a	973;0	625	346;31	6297	75000	9.9-7	9.9-7	1.5-6	-2.4-6	-2.8-5	-2.4-5	7:51	13:35	1:21:54
tai25b	973;0	625	141;39	6176	75000	9.9-7	9.9-7	1.4-6	-1.8-5	-2.6-5	-2.2-5	8:02	13:42	1:23:59
tai30a	1393;0	900	258;31	4212	65722	9.9-7	9.9-7	9.9-7	-7.0-6	-1.1-5	-7.5-6	17:39	19:37	2:43:13
tai30b	1393;0	900	106;35	4489	71044	9.9-7	9.9-7	9.9-7	-9.5-6	-2.1-5	-1.5-5	5:07	21:28	2:56:30
tai35a	1888;0	1225	246;30	3953	59536	9.7-7	9.9-7	9.9-7	-5.9-6	-8.9-6	-6.0-6	13:08	40:30	5:06:41
tai35b	1888;0	1225	128;43	6624	75000	9.9-7	9.9-7	1.6-6	-1.7-5	-2.4-5	-2.1-5	19:14	1:09:33	6:24:39
tai40a	2458;0	1600	325;31	3973	60071	9.8-7	9.9-7	9.9-7	-5.0-6	-8.6-6	-5.7-6	32:43	1:24:16	9:43:53
tai40b	2458;0	1600	126;33	4316	66912	9.2-7	9.9-7	9.9-7	-1.1-5	-1.5-5	-1.0-5	29:47	2:43:34	12:22:13
tho30	1393;0	900	159;43	6624	75000	9.9-7	9.9-7	1.6-6	-1.7-5	-2.3-5	-2.0-5	12:49	1:16:52	3:51:02
tho40	2458;0	1600	194;43	6242	75000	9.8-7	9.9-7	1.5-6	-1.6-5	-2.1-5	-1.8-5	1:41:24	3:51:16	24:09:34
be250.1	251;0	251	0;707	2448	2547	9.9-7	9.9-7	9.1-7	4.5-7	-1.2-6	-2.6-6	24	1:05	39
be250.2	251;0	251	0;693	2571	2525	9.8-7	9.9-7	9.5-7	6.5-7	-5.1-6	-2.7-6	15	57	41
be250.3	251;0	251	0;613	2346	2525	9.9-7	9.9-7	9.7-7	-4.4-7	2.5-7	-2.8-6	22	1:11	41
be250.4	251;0	251	0;694	2482	2547	9.9-7	9.9-7	8.5-7	2.7-7	-4.5-6	-2.7-6	23	1:14	41
be250.5	251;0	251	0;598	878	2598	9.9-7	9.9-7	9.4-7	-2.8-7	-2.0-6	1.6-6	26	28	42
be250.6	251;0	251	0;599	893	2547	9.9-7	9.9-7	8.9-7	-5.7-7	-5.7-6	-2.5-6	14	24	40
be250.7	251;0	251	0;621	2959	2525	9.9-7	9.9-7	9.6-7	-4.8-7	3.4-6	-2.9-6	21	1:26	41
be250.8	251;0	251	0;649	995	2547	9.9-7	9.9-7	8.5-7	-8.1-8	8.5-7	-2.7-6	26	29	40
be250.9	251;0	251	0;743	3326	2546	9.9-7	9.9-7	9.9-7	4.5-7	-9.4-7	-8.3-8	24	1:30	41
be250.10	251;0	251	0;637	2716	2547	9.9-7	9.9-7	8.9-7	-1.4-7	4.6-6	-2.7-6	15	1:10	40
bqp500-1	501;0	501	0;676	2378	3365	9.9-7	9.7-7	9.3-7	1.2-6	9.0-6	9.0-7	2:59	8:19	7:23
bqp500-2	501;0	501	0;741	2562	4448	9.2-7	9.8-7	9.9-7	1.0-6	9.9-6	-5.3-6	3:15	9:08	9:54
bqp500-3	501;0	501	0;674	2550	4430	9.9-7	9.9-7	9.9-7	1.4-6	1.0-5	-5.4-6	3:01	9:01	9:53

TABLE 3. (*cont.*)

			Iteration			η			η_g			Time		
Problem	$m_E;m_I$	n_s	ABCD	APG	eARBCG	ABCD	APG	eARBCG	ABCD	APG	eARBCG	ABCD	APG	eARBCG
bqp500-4	501;0	501	0;648	2574	4503	9.9-7	9.9-7	9.9-7	1.2-6	1.1-5	-5.5-6	2:53	9:00	10:09
bqp500-5	501;0	501	0;760	2507	4000	9.9-7	9.9-7	9.9-7	8.7-7	9.7-6	-5.2-6	3:35	8:48	8:41
bqp500-6	501;0	501	0;733	2450	3472	9.9-7	9.9-7	9.6-7	8.4-7	9.3-6	-3.3-6	3:17	8:26	7:19
bqp500-7	501;0	501	0;709	2483	3621	9.9-7	9.9-7	9.9-7	9.9-7	9.8-6	-5.0-6	2:56	8:05	7:34
bqp500-8	501;0	501	0;713	2449	3472	9.8-7	9.9-7	9.3-7	1.4-6	9.5-6	-3.6-6	2:28	7:43	7:19
bqp500-9	501;0	501	0;674	2452	3431	9.9-7	9.8-7	9.8-7	7.7-7	9.6-6	-3.3-6	3:02	8:32	7:12
bqp500-10	501;0	501	0;692	2570	4477	9.9-7	9.9-7	9.9-7	1.1-6	1.0-5	-5.4-6	2:47	8:22	8:16
gka1e	201;0	201	0;675	958	2999	9.9-7	9.9-7	9.7-7	7.2-7	4.8-6	2.8-6	18	19	31
gka2e	201;0	201	0;719	1155	2632	9.9-7	9.9-7	9.9-7	9.1-7	5.9-6	9.2-7	19	20	27
gka3e	201;0	201	0;753	2995	3730	9.9-7	9.9-7	9.6-7	5.0-7	-2.4-6	2.8-6	22	1:01	37
gka4e	201;0	201	0;735	2445	2036	9.9-7	9.9-7	9.9-7	-2.5-7	-1.7-6	-8.0-7	21	47	21
gka5e	201;0	201	0;712	2931	2598	9.9-7	9.9-7	9.8-7	4.6-8	-3.5-6	-3.4-6	19	57	27
gka1f	501;0	501	0;630	2233	2731	9.8-7	9.9-7	9.8-7	4.4-7	-7.4-6	-4.2-6	2:53	7:55	5:54
gka2f	501;0	501	0;684	2298	3338	9.7-7	9.8-7	9.9-7	8.4-7	9.1-6	3.2-6	3:05	8:11	7:11
gka3f	501;0	501	0;801	2579	4430	9.9-7	9.9-7	9.9-7	9.3-7	1.0-5	-5.4-6	3:29	8:40	9:39
gka4f	501;0	501	0;751	2798	5158	9.9-7	9.9-7	9.8-7	8.3-7	1.1-5	4.2-6	2:36	9:15	11:20
gka5f	501;0	501	0;749	2120	5158	9.7-7	9.9-7	9.9-7	1.0-6	-9.6-6	4.2-6	3:28	7:20	9:43
soybean-large.2	308;0	307	0;1745	6307	13795	9.8-7	9.8-7	9.7-7	-7.2-7	-9.4-10	-4.9-8	1:17	3:07	5:03
soybean-large.3	308;0	307	0;1764	4676	14096	9.1-7	9.9-7	7.3-7	-1.8-7	-3.8-8	-1.2-8	45	2:17	5:22
soybean-large.4	308;0	307	0;1262	4417	6965	8.5-7	9.9-7	8.9-7	1.2-7	7.2-9	-1.9-8	34	2:16	2:31
soybean-large.5	308;0	307	0;1017	4492	6219	9.4-7	9.9-7	9.8-7	-2.9-8	2.4-9	-2.2-8	43	2:11	2:19
soybean-large.6	308;0	307	0;601	2623	7468	9.8-7	9.9-7	9.8-7	-8.3-9	-1.6-9	-3.7-11	26	1:19	2:45
soybean-large.7	308;0	307	0;740	2103	4065	9.8-7	9.9-7	8.7-7	5.8-9	-7.0-10	1.0-9	32	1:03	1:26
soybean-large.8	308;0	307	0;706	2501	4351	9.9-7	9.9-7	9.9-7	-1.3-8	2.9-9	-8.4-9	34	1:18	1:34
soybean-large.9	308;0	307	0;728	2289	4560	9.4-7	9.9-7	7.8-7	3.1-8	-2.4-9	2.7-8	27	1:20	1:43
soybean-large.10	308;0	307	0;525	2227	4560	9.5-7	9.9-7	9.3-7	5.5-8	-2.5-9	3.0-8	15	1:15	1:39
soybean-large.11	308;0	307	0;484	3806	2261	9.7-7	9.9-7	8.3-7	1.7-8	5.4-9	-1.3-8	17	1:57	46
spambase-large.2	1501;0	1500	0;62	217	263	5.4-7	8.2-7	2.8-8	-7.0-8	-4.0-10	4.0-10	4:02	8:40	6:47
spambase-large.3	1501;0	1500	0;62	217	263	5.4-7	8.2-7	2.8-8	-7.0-8	-3.7-10	4.2-10	3:07	7:36	6:47
spambase-large.4	1501;0	1500	0;62	217	263	5.4-7	8.2-7	2.8-8	-7.0-8	-3.5-10	4.4-10	3:47	8:39	6:47
spambase-large.5	1501;0	1500	0;62	217	263	5.4-7	8.2-7	2.8-8	-7.0-8	-3.2-10	4.5-10	3:53	9:30	6:38
spambase-large.6	1501;0	1500	0;62	217	263	5.4-7	8.2-7	2.9-8	-7.0-8	-3.0-10	4.7-10	3:41	9:34	6:22
spambase-large.7	1501;0	1500	0;62	217	263	5.4-7	8.2-7	2.9-8	-7.0-8	-2.7-10	4.9-10	3:47	10:21	6:36
spambase-large.8	1501;0	1500	0;62	217	263	5.4-7	8.2-7	2.9-8	-7.0-8	-2.5-10	5.0-10	3:56	9:55	6:40
spambase-large.9	1501;0	1500	0;62	217	263	5.4-7	8.2-7	2.9-8	-7.0-8	-2.2-10	5.2-10	3:53	9:14	6:19
spambase-large.10	1501;0	1500	0;62	217	263	5.4-7	8.2-7	2.9-8	-7.0-8	-2.0-10	5.4-10	4:12	8:59	5:36
spambase-large.11	1501;0	1500	0;62	217	263	5.4-7	8.2-7	2.9-8	-7.0-8	-1.7-10	5.5-10	4:08	8:04	7:12
abalone-large.2	1001;0	1000	0;869	10956	80482	9.9-7	9.8-7	9.9-7	1.1-8	3.4-9	-1.3-8	11:45	2:26:39	8:46:18
abalone-large.3	1001;0	1000	0;695	5033	41764	9.9-7	9.9-7	9.9-7	-2.2-8	-4.2-9	-4.0-9	10:44	1:10:46	4:53:35

TABLE 3. (*cont.*)

			Iteration			η			η_g			Time		
Problem	$m_E;m_I$	n_s	ABCD	APG	eARBCG	ABCD	APG	eARBCG	ABCD	APG	eARBCG	ABCD	APG	eARBCG
abalone-large.4	1001;0	1000	0;716	3089	18374	8.2-7	9.9-7	6.9-7	5.9-8	-1.3-9	-2.6-10	10:49	41:57	2:24:14
abalone-large.5	1001;0	1000	0;590	2872	10594	9.6-7	9.9-7	6.0-7	4.6-8	-6.5-10	-1.9-9	9:28	39:26	1:33:36
abalone-large.6	1001;0	1000	0;561	3104	7697	9.7-7	9.9-7	9.3-7	1.0-8	-1.9-9	-1.5-9	9:01	43:42	1:07:38
abalone-large.7	1001;0	1000	0;522	2949	7470	9.7-7	9.9-7	9.9-7	-5.5-9	8.8-10	2.6-9	8:43	40:16	1:07:55
abalone-large.8	1001;0	1000	0;490	2920	5533	9.7-7	9.9-7	9.5-7	-1.9-8	1.8-9	4.0-9	7:53	36:26	51:42
abalone-large.9	1001;0	1000	0;507	3117	3509	9.9-7	9.9-7	8.7-7	-1.9-10	-2.5-9	6.4-9	8:47	39:25	31:59
abalone-large.10	1001;0	1000	0;503	3078	3202	9.9-7	9.9-7	8.6-7	-6.6-9	3.2-9	5.7-10	9:21	35:18	29:41
abalone-large.11	1001;0	1000	0;531	2578	3206	9.5-7	9.9-7	9.3-7	3.0-8	2.0-9	6.0-10	8:26	32:41	28:18
segment-large.2	1001;0	1000	0;52	121	210	8.8-7	9.7-7	7.2-7	-2.0-7	6.6-9	1.6-8	1:09	1:38	1:57
segment-large.3	1001;0	1000	0;52	121	210	8.8-7	9.7-7	7.3-7	-2.0-7	6.8-9	1.6-8	1:13	1:44	1:55
segment-large.4	1001;0	1000	0;52	121	210	8.8-7	9.7-7	7.3-7	-2.0-7	7.1-9	1.7-8	1:04	1:35	1:56
segment-large.5	1001;0	1000	0;52	121	210	8.8-7	9.7-7	7.3-7	-2.0-7	7.4-9	1.7-8	1:08	1:31	1:54
segment-large.6	1001;0	1000	0;52	121	210	8.8-7	9.7-7	7.4-7	-2.0-7	7.6-9	1.7-8	1:04	1:34	1:55
segment-large.7	1001;0	1000	0;52	121	210	8.8-7	9.7-7	7.4-7	-2.0-7	7.9-9	1.7-8	1:05	1:32	1:46
segment-large.8	1001;0	1000	0;52	121	210	8.8-7	9.7-7	7.5-7	-2.0-7	8.2-9	1.7-8	55	1:33	1:46
segment-large.9	1001;0	1000	0;52	121	210	8.8-7	9.7-7	7.5-7	-2.0-7	8.4-9	1.7-8	48	1:30	1:42
segment-large.10	1001;0	1000	0;52	121	210	8.8-7	9.7-7	7.6-7	-2.0-7	8.7-9	1.8-8	46	1:23	1:37
segment-large.11	1001;0	1000	0;52	121	210	8.8-7	9.7-7	7.7-7	-2.0-7	8.9-9	1.8-8	43	1:40	1:27
housing.2	507;0	506	0;15	47	54	1.8-7	8.2-7	4.7-7	3.4-9	-3.5-8	4.1-9	03	07	04
housing.3	507;0	506	0;15	47	54	1.8-7	8.2-7	4.7-7	3.4-9	-3.5-8	4.1-9	03	05	04
housing.4	507;0	506	0;15	47	54	1.7-7	8.1-7	4.7-7	3.4-9	-3.5-8	4.2-9	03	06	04
housing.5	507;0	506	0;15	47	54	1.7-7	8.1-7	4.7-7	3.4-9	-3.5-8	4.2-9	03	08	04
housing.6	507;0	506	0;15	47	54	1.7-7	8.1-7	4.7-7	3.4-9	-3.5-8	4.2-9	03	08	04
housing.7	507;0	506	0;15	47	54	1.7-7	8.1-7	4.7-7	3.4-9	-3.5-8	4.2-9	02	08	04
housing.8	507;0	506	0;15	47	54	1.6-7	8.1-7	4.7-7	3.4-9	-3.5-8	4.2-9	03	07	06
housing.9	507;0	506	0;15	47	54	1.6-7	8.1-7	4.7-7	3.4-9	-3.5-8	4.2-9	04	08	06
housing.10	507;0	506	0;15	47	54	1.6-7	8.1-7	4.7-7	3.4-9	-3.5-8	4.2-9	04	08	06
housing.11	507;0	506	0;15	47	54	1.6-7	8.1-7	4.7-7	3.4-9	-3.5-8	4.2-9	03	08	07
ex-be250.1	251;93375	251	0;719	9100	3005	9.9-7	9.9-7	9.9-7	-4.1-8	-6.1-9	-1.7-8	1:16	14:51	2:38
ex-be250.2	251;93375	251	0;705	9409	2656	9.9-7	9.9-7	9.8-7	-7.8-8	1.4-9	-3.0-8	1:25	14:22	2:09
ex-be250.3	251;93375	251	0;706	9195	2733	9.9-7	9.9-7	9.9-7	-3.4-8	-2.2-9	2.1-9	1:10	10:56	2:29
ex-be250.4	251;93375	251	0;705	12298	2976	9.9-7	9.9-7	9.7-7	-5.0-8	-7.3-9	-8.8-9	1:12	17:06	2:29
ex-be250.5	251;93375	251	0;705	10136	2894	9.8-7	9.9-7	9.9-7	-8.6-9	-1.4-8	-3.5-8	1:13	12:38	2:16
ex-be250.6	251;93375	251	0;685	10050	2701	9.9-7	9.9-7	9.9-7	-5.7-8	4.0-9	-2.7-8	1:10	16:06	2:10
ex-be250.7	251;93375	251	0;711	10887	2892	9.9-7	9.9-7	9.9-7	-2.9-8	-5.2-9	-1.2-8	1:12	16:42	2:19
ex-be250.8	251;93375	251	0;712	9672	2733	9.9-7	9.9-7	9.8-7	-4.6-8	6.7-9	-2.8-8	1:13	12:50	2:17
ex-be250.9	251;93375	251	0;711	6718	2645	9.9-7	9.9-7	9.8-7	-6.8-8	5.9-9	-5.1-9	1:18	10:21	2:05
ex-be250.10	251;93375	251	0;717	11436	2843	9.9-7	9.9-7	9.7-7	-6.9-8	-1.9-9	-3.4-9	1:16	14:40	2:27
ex-bqp500-1	501;374250	501	0;1439	15676	16436	9.9-7	9.9-7	8.8-7	-7.8-7	3.1-8	-4.1-7	14:34	2:00:25	1:42:18

TABLE 3. (*cont.*)

			Iteration			η			η_g			Time		
Problem	$m_E; m_I$	n_s	ABCD	APG	eARBCG	ABCD	APG	eARBCG	ABCD	APG	eARBCG	ABCD	APG	eARBCG
ex-bqp500-2	501;374250	501	0;1545	14458	17465	9.9-7	9.9-7	9.6-7	-8.2-7	4.4-8	-3.7-7	16:01	58:40	1:56:40
ex-bqp500-3	501;374250	501	0;1795	15400	19460	9.9-7	9.9-7	9.5-7	-9.0-7	4.4-8	-2.5-7	19:43	1:12:55	1:51:14
ex-bqp500-4	501;374250	501	0;1738	20092	19899	9.9-7	9.9-7	9.9-7	-9.1-7	4.8-8	-4.6-7	12:39	2:25:38	2:07:29
ex-bqp500-5	501;374250	501	0;1777	15230	19517	9.9-7	9.9-7	9.3-7	-9.0-7	2.3-8	-2.2-7	17:38	1:01:55	2:10:26
ex-bqp500-6	501;374250	501	0;1902	16149	17590	9.5-7	9.9-7	9.0-7	-7.0-7	5.9-8	-5.0-7	19:49	1:20:43	1:36:51
ex-bqp500-7	501;374250	501	0;1835	15655	17481	9.9-7	9.9-7	9.9-7	-8.1-7	-6.2-8	-4.5-7	20:18	1:57:19	1:51:46
ex-bqp500-8	501;374250	501	0;1798	16958	22473	9.9-7	9.9-7	9.7-7	-8.8-7	-8.2-9	-1.9-7	18:12	1:09:32	2:31:37
ex-bqp500-9	501;374250	501	0;1977	15086	19588	9.9-7	9.9-7	9.9-7	-6.6-7	5.0-8	-2.6-7	20:19	1:14:18	2:04:43
ex-bqp500-10	501;374250	501	0;1447	16325	19686	9.7-7	9.9-7	9.8-7	-7.6-7	-6.2-8	-2.9-7	15:30	2:01:24	2:06:46
ex-gka1e	201;59700	201	0;750	7944	3909	9.9-7	9.9-7	9.9-7	-2.6-8	7.6-9	-4.1-9	35	7:18	1:38
ex-gka2e	201;59700	201	0;1651	25000	9885	9.9-7	1.0-5	9.6-7	5.9-7	1.0-7	-3.7-7	1:17	19:30	3:53
ex-gka3e	201;59700	201	0;1577	15270	13640	9.9-7	9.9-7	9.2-7	-3.2-7	-9.9-9	-3.3-7	1:07	14:41	5:02
ex-gka4e	201;59700	201	0;1460	25000	19976	9.9-7	1.7-5	9.9-7	-6.0-7	3.3-9	-5.2-7	1:20	22:57	7:34
ex-gka5e	201;59700	201	0;2038	25000	20715	9.9-7	1.9-5	9.9-7	-9.0-7	8.4-8	-6.1-7	1:16	22:12	7:42
ex-gka1f	501;374250	501	0;643	5599	2864	9.9-7	9.9-7	9.7-7	-8.4-8	-3.8-9	-1.4-9	7:15	50:41	18:28
ex-gka2f	501;374250	501	0;1049	8569	8939	9.9-7	9.9-7	9.8-7	-4.1-7	3.7-8	-1.8-7	6:18	1:08:19	55:07
ex-gka3f	501;374250	501	0;1798	21403	26462	9.9-7	9.9-7	9.2-7	-9.4-7	-5.6-8	-5.9-7	13:14	2:12:26	2:53:59
ex-gka4f	501;374250	501	0;2169	25000	37127	9.9-7	2.2-6	9.9-7	-7.2-7	-2.2-8	-6.9-7	21:49	2:27:06	3:11:07
ex-gka5f	501;374250	501	0;2259	25000	37456	9.8-7	1.6-6	9.9-7	-1.2-6	-5.5-8	-1.1-6	24:17	2:26:09	4:00:00

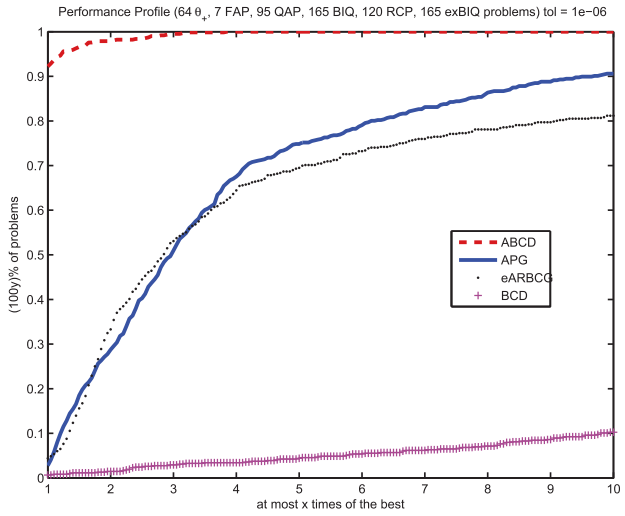

FIG. 1. *Performance profiles of ABCD, APG, eARBCG, and BCD on* $[1, 10]$.

respectively. The number of iterations,[3] the relative residual η and relative gap η_g, as well as computation times (in the format hours:minutes:seconds) are listed in the last twelve columns. As can be seen, ABCD is much faster than APG and eARBCG for most of the problems.

Figure 1 shows the performance profiles of the ABCD, APG, eARBCG, and BCD methods for all the 616 tested problems. We recall that a point (x, y) is in the performance profile curve of a method if and only if it can solve exactly $(100y)\%$ of all the tested problems at most x times slower than any other methods. It can be seen that ABCD outperforms the other 3 methods by a significant margin. Furthermore, the ABCD method is more than ten times faster than the BCD method for a vast majority of the problems. It is quite surprising that a simple novel acceleration step with a special BCD cycle can improve the performance of the standard Gauss–Seidel BCD method by such a dramatic margin.

Figure 2 shows the tolerance profiles of the ABCD method for all the 616 tested problems. Note that a point (x, y) is in the tolerance profile curve if and only if it can solve exactly $(100y)\%$ of all the tested problems at most x times slower than the time taken to reach the tolerance of 10^{-6}.

7. Conclusions. We have designed an inexact ABCGD method for solving a multiblock convex minimization problem whose objective is the sum of a coupled smooth function with Lipschitz continuous gradient and a separable (possibly nonsmooth) function involving only the first two blocks. An important class of problems with the specified structure is the dual of LSSDP where the primal matrix variable must satisfy given linear equality and inequality constraints, and must also lie in the intersection of the cone of positive semidefinite matrices and a simple polyhedral set.

Our inexact ABCGD method has $O(1/k^2)$ iteration complexity if the subproblems are solved progressively more accurately. The design of our ABCGD method relied on recent advances in the sGS technique for solving a convex minimization problem whose objective is the sum of a multiblock quadratic function and a nonsmooth function

[3] Note that the two numbers of iterations in ABCD are for ABCD-2 and ABCD-1, respectively.

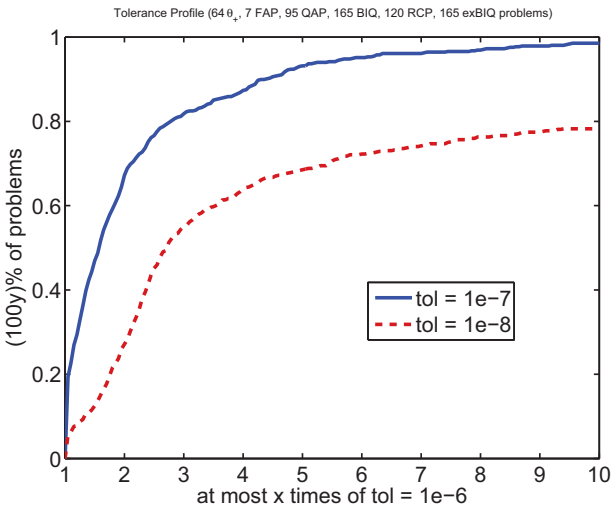

FIG. 2. *Tolerance profiles of ABCD on* $[1, 10]$.

involving only the first block. Extensive numerical experiments on various classes of over 600 large scale LSSDP problems demonstrate that our ABCGD method, which reduces to the ABCD method for LSSDP problems, not only can solve the problems to high accuracy, but it is also far more efficient than (a) the well known BCD method, (b) the eARBCG method, and (c) the APG method.

Acknowledgment. We would like to thank Ms. Ying Cui from National University of Singapore for her comments on refining the Danskin-type theorem presented in a previous version of this paper.

REFERENCES

[1] A. BECK AND M. TEBOULLE, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imag. Sci., 2 (2009), pp. 183–202.

[2] A. BECK AND L. TETRUASHVILI, *On the convergence of block coordinate descent type methods*, SIAM J. Optim., 23 (2013), pp. 2037–2060.

[3] S. BURER, *On the copositive representation of binary and continuous nonconvex quadratic programs*, Math. Program., 120 (2009), pp. 479–495.

[4] S. BURER, R. D. MONTEIRO, AND Y. ZHANG, *A computational study of a gradient-based log-barrier algorithm for a class of large-scale SDPs*, Math. Program., 95 (2003), pp. 359–379.

[5] A. CHAMBOLLE AND C. DOSSAL, *On the convergence of the iterates of the "fast iterative shrinkage/thresholding algorithm"*, J. Optim. Theory Appl., 166 (2015), pp. 968–982.

[6] A. CHAMBOLLE AND T. POCK, *A remark on accelerated block coordinate descent for computing the proximity operators of a sum of convex functions*, SMAI J. Comput. Math., 1 (2015), pp. 29–54.

[7] J. M. DANSKIN, *The theory of max-min, with applications*, SIAM J. Appl. Math., 14 (1966), pp. 641–664.

[8] A. EISENBLÄTTER, M. GRÖTSCHEL, AND A. M. KOSTER, *Frequency planning and ramifications of coloring*, Discuss. Math. Graph Theory, 22 (2002), pp. 51–88.

[9] F. FACCHINEI AND J.-S. PANG, *Finite-Dimensional Variational Inequalities and Complementarity Problems*, Vol. 1, Springer Ser. Oper. Res., Springer, New York, 2003.

[10] O. FERCOQ AND P. RICHTÁRIK, *Accelerated, parallel and proximal coordinate descent*, SIAM J. Optim., 25 (2015), pp. 1997–2023.

[11] L. GRIPPO AND M. SCIANDRONE, *On the convergence of the block nonlinear Gauss–Seidel method under convex constraints*, Oper. Res. Lett., 26 (2000), pp. 127–136.

[12] P. HAHN AND M. ANJOS, *QAPLIB – A Quadratic Assignment Problem Library*, http://www.seas.upenn.edu/qaplib.

[13] J.-B. Hiriart-Urruty, J.-J. Strodiot, and V. H. Nguyen, *Generalized Hessian matrix and second-order optimality conditions for problems with $C^{1,1}$ data*, Appl. Math. Optim., 11 (1984), pp. 43–56.
[14] K. Jiang, D. Sun, and K.-C. Toh, *An inexact accelerated proximal gradient method for large scale linearly constrained convex SDP*, SIAM J. Optim., 22 (2012), pp. 1042–1064.
[15] X. Li, D. F. Sun, and K.-C. Toh, *QSDPNAL: A two-phase proximal augmented Lagrangian method for convex quadratic semidefinite programming*, preprint, arXiv:1512.08872, 2015.
[16] X. Li, D. F. Sun, and K.-C. Toh, *A Schur complement based semi-proximal ADMM for convex quadratic conic programming and extensions*, Math. Program., 155 (2016), pp. 333–373.
[17] Q. Lin, Z. Lu, and L. Xiao, *An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization*, SIAM J. Optim., 25 (2015), pp. 2244–2273.
[18] Y. Nesterov, *A method of solving a convex programming problem with convergence rate $O(1/k^2)$*, Sov. Math. Dokl., 27 (1983), pp. 372–376.
[19] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Appl. Optim. 87, Springer Science+Business Media, New York, 2004.
[20] Y. Nesterov, *Smooth minimization of non-smooth functions*, Math. Program., 103 (2005), pp. 127–152.
[21] Y. Nesterov, *Efficiency of coordinate descent methods on huge-scale optimization problems*, SIAM J. Optim., 22 (2012), pp. 341–362.
[22] J. Peng and Y. Wei, *Approximating k-means-type clustering via semidefinite programming*, SIAM J. Optim., 18 (2007), pp. 186–205.
[23] J. Povh and F. Rendl, *Copositive and semidefinite relaxations of the quadratic assignment problem*, Discrete Optim., 6 (2009), pp. 231–241.
[24] R. T. Rockafellar, *Convex Analysis*, Princeton Math. 28, Princeton University Press, Princeton, NJ, 1970.
[25] R. T. Rockafellar, *Conjugate Duality and Optimization*, CBMS-NSF Regional Conf. Ser. in Appl. Math. 16, SIAM, Philadelphia, 1974.
[26] R. T. Rockafellar, *Augmented Lagrangians and applications of the proximal point algorithm in convex programming*, Math. Oper. Res., 1 (1976), pp. 97–116.
[27] R. T. Rockafellar, *Monotone operators and the proximal point algorithm*, SIAM J. Control Optim., 14 (1976), pp. 877–898.
[28] A. Saha and A. Tewari, *On the nonasymptotic convergence of cyclic coordinate descent methods*, SIAM J. Optim., 23 (2013), pp. 576–601.
[29] S. Sardy, A. G. Bruce, and P. Tseng, *Block coordinate relaxation methods for nonparametric wavelet denoising*, J. Comput. Graph. Statist., 9 (2000), pp. 361–379.
[30] N. Sloane, *Challenge problems: Independent sets in graphs*, https://oeis.org/A265032/a265032.html (2005).
[31] K.-C. Toh, *Solving large scale semidefinite programs via an iterative solver on the augmented systems*, SIAM J. Optim., 14 (2004), pp. 670–698.
[32] M. Trick, V. Chvatal, B. Cook, D. Johnson, C. McGeoch, and R. Tarjan, *The second dimacs implementation challenge — NP hard problems: Maximum clique, graph coloring, and satisfiability*, in DIMACS, http://dimacs.rutgers.edu/Challenges/ (1992).
[33] P. Tseng, *Dual coordinate ascent methods for non-strictly convex minimization*, Math. Program., 59 (1993), pp. 231–247.
[34] P. Tseng, *Convergence of a block coordinate descent method for nondifferentiable minimization*, J. Optim. Theory Appl., 109 (2001), pp. 475–494.
[35] P. Tseng, *On Accelerated Proximal Gradient Methods for Convex-Concave Optimization*, manuscript, 2008.
[36] P. Tseng and S. Yun, *A coordinate gradient descent method for nonsmooth separable minimization*, Math. Program., 117 (2009), pp. 387–423.
[37] L.-Q. Yang, D. F. Sun, and K.-C. Toh, *SDPNAL+: A majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints*, Math. Program. Comput., 7 (2015), pp. 331–366.
[38] X.-Y. Zhao, D. Sun, and K.-C. Toh, *A Newton-CG augmented Lagrangian method for semidefinite programming*, SIAM J. Optim., 20 (2010), pp. 1737–1765.