A CONVERGENT 3-BLOCK SEMIPROXIMAL ALTERNATING DIRECTION METHOD OF MULTIPLIERS FOR CONIC PROGRAMMING WITH 4-TYPE CONSTRAINTS*

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

Abstract. In this paper, we consider conic programming problems whose constraints consist of linear equalities, linear inequalities, a nonpolyhedral cone, and a polyhedral cone. A convenient way for solving this class of problems is to apply the directly extended alternating direction method of multipliers (ADMM) to its dual problem, which has been observed to perform well in numerical computations but may diverge in theory. Ideally, one should find a convergent variant which is at least as efficient as the directly extended ADMM in practice. We achieve this goal by designing a convergent semiproximal ADMM (called sPADMM3c for convenience) for convex programming problems having three separable blocks in the objective function with the third part being linear. At each iteration, the proposed sPADMM3c takes one special block coordinate descent (BCD) cycle with the order $1 \to 3 \to 2 \to 3$, instead of the usual $1 \to 2 \to 3$ Gauss–Seidel BCD cycle used in the nonconvergent directly extended 3-block ADMM, for updating the variable blocks. Our numerical experiments demonstrate that the convergent method is at least 20% faster than the directly extended ADMM with unit step-length for the vast majority of about 550 large-scale doubly nonnegative semidefinite programming problems with linear equality and/or inequality constraints. This confirms that at least for conic convex programming, one can design a convergent and efficient ADMM with a special BCD cycle of updating the variable blocks.

1. Introduction. Our primary motivation in this paper is to design an efficient but simple first order method with guaranteed convergence to find a solution of moderate accuracy to the following conic programming with four types of constraints:

$$\text{(P)} \quad \max\left\{\langle -c, x\rangle \mid \mathcal{A}_E x = b_E,\ \mathcal{A}_I x \geq b_I,\ x \in \mathcal{K},\ x \in \mathcal{K}_p\right\}.$$

Here $\mathcal{A}_E$ and $\mathcal{A}_I$ are two linear maps defined from the finite-dimensional real Euclidean space $\mathcal{X}$ to $\Re^{m_E}$ and $\Re^{m_I}$, respectively, $(b_E, b_I) \in \Re^{m_E} \times \Re^{m_I}$ and $c \in \mathcal{X}$ are given data, $\mathcal{K} \subseteq \mathcal{X}$ is a pointed closed convex (nonpolyhedral) cone whose interior $\text{int}(\mathcal{K}) \neq \emptyset$, and $\mathcal{K}_p$ is a polyhedral convex cone in $\mathcal{X}$ such that $\mathcal{K} \cap \mathcal{K}_p$ is nonempty. Note that in theory the fourth block constraint in (P) can be incorporated into the first and the second block constraints. However, treating the polyhedral cone $\mathcal{K}_p$ separately instead of representing it in terms of equalities and inequalities is of considerable advantage in numerical computations. Problem (P) covers a wide range of

---

†Department of Mathematics and Risk Management Institute, National University of Singapore, 10 Lower Kent Ridge Road, Singapore 119076 (matsundf@nus.edu.sg). This author's research was 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@nus.edu.sg). 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.

interesting convex models. In particular, it includes the important class of doubly non-negative (DNN) semidefinite programming (SDP) with both equality and inequality constraints

$$\text{(SDP)} \quad \max\left\{\langle -C, X\rangle \mid \mathcal{A}_E X = b_E, \ \mathcal{A}_I X \geq b_I, \ X \in \mathcal{S}_+^n, \ X \in \mathcal{K}_p\right\},$$

where $\mathcal{S}_+^n$ is the cone of $n \times n$ symmetric and positive semidefinite matrices in the space of $n \times n$ symmetric matrices $\mathcal{S}^n$, $X \in \mathcal{K}_p$ means that every entry of the matrix $X \in \mathcal{S}^n$ is nonnegative (one can, of course, only require a subset of the entries of X to be nonnegative or nonpositive or free), and $C \in \mathcal{S}^n$ is a given symmetric matrix.

For a given linear map $\mathcal{A}$ from $\mathcal{X}$ to another finite-dimensional real Euclidean space $\mathcal{X}'$, we denote its adjoint by $\mathcal{A}^*$, and for any closed convex set $\mathcal{C} \subseteq \mathcal{X}$, we denote the metric projection operator onto $\mathcal{C}$ by $\Pi_{\mathcal{C}}(\cdot)$. If $\mathcal{C} \subseteq \mathcal{X}$ is a closed convex cone, we denote its dual cone by $\mathcal{C}^*$, i.e.,

$$\mathcal{C}^* := \{d \in \mathcal{X} \mid \langle d, x\rangle \geq 0 \quad \forall\, x \in \mathcal{C}\}.$$

In this paper, we will make extensive use of the Moreau decomposition theorem [26], which states that $x = \Pi_{\mathcal{C}}(x) - \Pi_{\mathcal{C}^*}(-x)$ for any nonempty closed convex cone $\mathcal{C} \subseteq \mathcal{X}$ and $x \in \mathcal{X}$.

The dual of (P) takes the form of

$$\min\left\{-\langle b_I, y_I\rangle - \langle b_E, y_E\rangle \mid s + \mathcal{A}_I^* y_I + z + \mathcal{A}_E^* y_E = c, \ s \in \mathcal{K}^*, \ z \in \mathcal{K}_p^*, \ y_I \geq 0\right\},$$

which can equivalently be written as the following convex programming with four separate blocks in the objective function and a coupling linear equation constraint:

$$\begin{aligned} \text{(D)} \quad \min \quad & \delta_{\mathcal{K}^*}(s) + \left(\delta_{\Re_+^{m_I}}(y_I) - \langle b_I, y_I\rangle\right) + \delta_{\mathcal{K}_p^*}(z) - \langle b_E, y_E\rangle \\ \text{s.t.} \quad & s + \mathcal{A}_I^* y_I + z + \mathcal{A}_E^* y_E = c, \end{aligned}$$

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.

Problem (D) belongs to a class of multiblock convex optimization problems whose objective function is the sum of q convex functions without overlapping variables:

$$(1.1) \qquad \min\left\{\sum_{i=1}^q \theta_i(w_i) \mid \sum_{i=1}^q \mathcal{B}_i^* w_i = c\right\},$$

where for each $i \in \{1, \ldots, q\}$, $\mathcal{W}_i$ is a finite-dimensional real Euclidean space equipped with an inner product $\langle \cdot, \cdot\rangle$ and its induced norm $\|\cdot\|$, $\theta_i : \mathcal{W}_i \mapsto (-\infty, +\infty]$ is a closed proper convex functions, $\mathcal{B}_i : \mathcal{X} \mapsto \mathcal{W}_i$ is a linear map, and $c \in \mathcal{X}$ is given. Note that one can write (D) in the form of (1.1) in a number of different ways. One natural choice is of course to write (D) in terms of (1.1) for $q = 4$ with $(w_1, w_2, w_3, w_4) \equiv (s, y_I, z, y_E)$. However, by noting that in (D) the objective function containing the y_E part is a linear term, we shall treat (D) as a special case of (1.1) for $q = 3$ with $(w_1, w_2, w_3) \equiv (s, y_I, (z, y_E))$. In the latter case, the third function θ_3 is partially linear: it is linear about y_E but nonlinear about z. This partial linear structure of θ_3 will be heavily exploited in our pursuit of designing a convergent but efficient first order method in this paper.

Let $\sigma > 0$ be given. The augmented Lagrange function for (1.1) is defined by

$$L_\sigma(w_1, \ldots, w_q; x) := \sum_{i=1}^q \theta_i(w_i) + \left\langle x, \sum_{i=1}^q \mathcal{B}_i^* w_i - c\right\rangle + \frac{\sigma}{2}\left\|\sum_{i=1}^q \mathcal{B}_i^* w_i - c\right\|^2$$

for $w_i \in \mathcal{W}_i$, $i = 1, \ldots, q$, and $x \in \mathcal{X}$. Choose any initial points $w_i^0 \in \mathrm{dom}(\theta_i)$, $i = 1, \ldots, q$, and $x^0 \in \mathcal{X}$. The classic augmented Lagrangian function method of Hestenes, Powell, and Rockafellar [22, 32, 34] consists of the following iterations:

$$(1.2) \qquad (w_1^{k+1}, \ldots, w_q^{k+1}) = \arg\min L_\sigma(w_1, \ldots, w_q; x^k),$$

$$(1.3) \qquad x^{k+1} = x^k + \tau\sigma\left(\sum_{i=1}^q \mathcal{B}_i^* w_i^{k+1} - c\right),$$

where $\tau > 0$, e.g., $\tau \in (0, 2)$, is a positive constant that controls the step-length in (1.3). To solve (1.2) exactly or approximately to high precision can be a challenging task in many situations. To deal with this challenge, one may try to replace (1.2) by considering the following q-block alternating direction method of multipliers (ADMM):

$$(1.4) \qquad \begin{aligned} w_1^{k+1} &= \arg\min L_\sigma(w_1, w_2^k \ldots, w_q^k; x^k), \\ &\ \ \vdots \\ w_i^{k+1} &= \arg\min L_\sigma(w_1^{k+1}, \ldots, w_{i-1}^{k+1}, w_i, w_{i+1}^k, \ldots, w_q^k; x^k), \\ &\ \ \vdots \\ w_q^{k+1} &= \arg\min L_\sigma(w_1^{k+1}, \ldots, w_{q-1}^{k+1}, w_q; x^k), \\ x^{k+1} &= x^k + \tau\sigma\left(\sum_{i=1}^q \mathcal{B}_i^* w_i^{k+1} - c\right). \end{aligned}$$

The above q-block ADMM is extended directly from the ADMM for solving the following 2-block convex optimization problem

$$(1.5) \qquad \min\left\{\theta_1(w_1) + \theta_2(w_2) \mid \mathcal{B}_1^* w_1 + \mathcal{B}_2^* w_2 = c\right\}.$$

For a chosen initial point $(w_1^0, w_2^0, x^0) \in \mathrm{dom}(\theta_1) \times \mathrm{dom}(\theta_2) \times \mathcal{X}$, the classic 2-block ADMM consists of the iterations

$$(1.6) \qquad w_1^{k+1} = \arg\min L_\sigma(w_1, w_2^k; x^k),$$

$$(1.7) \qquad w_2^{k+1} = \arg\min L_\sigma(w_1^{k+1}, w_2; x^k),$$

$$(1.8) \qquad x^{k+1} = x^k + \tau\sigma(\mathcal{B}_1^* w_1^{k+1} + \mathcal{B}_2^* w_2^{k+1} - c),$$

where $\tau > 0$ is a positive constant. The classic 2-block ADMM for solving (1.5) was first introduced by Glowinski and Marrocco [14] and Gabay and Mercier [12]. When $\mathcal{B}_1^* = \mathcal{I}$, the identity mapping, $\mathcal{B}_2^*$ is injective, and θ_1 is strongly convex, the convergence of the classic 2-block ADMM was proved first by Gabay and Mercier [12] for any $\tau \in (0, 2)$ if θ_2 is linear, and then by Glowinski [13] and Fortin and Glowinski [10] for any $\tau \in (0, (1 + \sqrt{5})/2)$ if θ_2 is a general nonlinear convex function. Gabay [11] has further shown that the classic 2-block ADMM for $\tau = 1$ is a special case of the Douglas–Rachford splitting method. Moreover, Eckstein and Bertsekas [6] have shown that the latter is actually an application of the proximal point algorithm on the dual problem by means of a specially constructed splitting operator. A variation of the classic 2-block ADMM is to adjust the penalty parameter σ at every iteration based on the previous iterations' progress with the goal of improving the convergence in practice. That is, one replaces (1.8) by

$$(1.9) \qquad x^{k+1} = x^k + \tau\sigma_k(\mathcal{B}_1^* w_1^{k+1} + \mathcal{B}_2^* w_2^{k+1} - c).$$

A scheme to adjust the penalty parameters σ_k was studied in [20, 41], and it often works well in practice. Due to its extreme simplicity and efficiency in several applications in mathematical imaging science, signal processing, etc., the classic 2-block ADMM has regained its popularity in recent years. For a tutorial on the classic 2-block ADMM, see the recent work by Eckstein and Yao [7].

The multiple-block ADMM with larger τ often works very well in many cases. For example, Wen, Goldfarb, and Yin [43] used the 3-block ADMM with $\tau = 1.618$ to design an efficient software for solving some SDP problems of large sizes. However, it was shown very recently by Chen et al. [4] that in contrast to the classic 2-block ADMM, the directly extended 3-block ADMM with $\tau = 1$ may diverge.[1] This dashes any hope of using the directly extended q-block ($q \geq 3$) ADMM without modifications.[2] Actually, even before the announcement of [4], several researchers made serious attempts in correcting the possible nonconvergence of the multiple-block ADMM [16, 17, 18, 19, 21, 37]. A recent work by Wang et al. [42] is also along this line. Among the work on correcting the nonconvergence, the q-block ADMM with an additional Gaussian back substitution [18] distinguishes itself for its simplicity and generality. However, to the best of our knowledge, up to now the dilemma is that at least for convex conic programming, the modified versions, though with a convergence guarantee, often perform two to three times slower than the multiblock ADMM with no convergent guarantee.

In this paper we aim to resolve the dilemma just mentioned by focusing on the conic programming problem (P). Note that (1.4) performs the usual $1 \to 2 \to \cdots \to q$ Gauss–Seidel block coordinate descent (BCD) cycle in minimizing the variable blocks for solving (1.2) inexactly. In contrast, our convergent semiproximal ADMM for convex programming problems having three separate blocks in the objective function with the third part being linear (we call this method sPADMM3c, which will be presented in section 3.1) takes the special $1 \to 3 \to 2 \to 3$ BCD cycle at each iteration. This special cycle actually uses an essentially BCD cyclic rule in the terminology of Tseng [40] to minimize the variable blocks for solving (1.2) inexactly. Given the fact that all three component functions in the objective of a counterexample constructed in [4] to illustrate the nonconvergence of the directly extended 3-block ADMM are zero functions, it comes as a pleasant surprise that one only needs to update the third variable block twice to get a convergent 3-block ADMM provided that θ_3 is linear. At the moment, it is not clear to us if our corresponding ADMM is still convergent when θ_3 is nonlinear, though we conjecture that it is true.[3] In any case, for the conic programming problem (P), the requirement on the linearity of θ_3 is not restrictive at all, as we will see in the subsequent analysis. Our extensive numerical tests on the important class of SDP problems with linear equality and/or inequality constraints demonstrate that our convergent method is at least 20% faster than the directly extended ADMM with unit step-length for the vast majority of about 550 large-scale problems tested. This confirms that our convergent sPADMM can have both theoretical convergence guarantee and superior numerical efficiency over the

---

[1]The final version of [4] includes a nonconvergent example for the ADMM with any prefixed τ not smaller than 10^{-8} and the penalty parameter $\sigma = 1$.

[2]Hong and Luo [23] provided a proof on the convergence of the directly extended q-block ADMM under some restrictive assumptions including a global error bound condition with a sufficiently small step-length τ. Since in practical computations one always prefers a larger step-length for better numerical efficiency, a convergence result of this nature is mainly of theoretical importance.

[3]In a recent paper [24], Li, Sun, and Toh report that the conjecture is indeed true if θ_3 is a convex quadratic function.

directly extended ADMM. Our research conducted here opens up the possibility of designing an efficient and convergent ADMM with a suitably chosen essentially BCD cyclic rule rather than the usual Gauss–Seidel BCD cyclic rule, where the latter may lead to a nonconvergent ADMM, for multiblock convex optimization problems (1.1) with structures beyond those considered in (SDP).

The remaining parts of this paper are organized as follows. In the next section, for our subsequent developments we will present in detail the convergent properties of a semiproximal ADMM for solving 2-block convex optimization problems. In section 3, we will introduce a convergent 3-block sPADMM first for the special case where the third function θ_3 is a linear function and then show how this approach can be applied to the case where θ_3 is partially linear or fully nonlinear. We should emphasize that the linear or partial linear structure of θ_3 will not render the directly extended 3-block ADMM to become convergent as the three functions θ_1, θ_2, and θ_3 constructed in the counterexample in [4] are all linear (actually, zero functions). Extensions to the multiblock case are also briefly discussed in this section. In section 4, the applications of the convergent 3-block sPADMM to conic programming are discussed. Section 5 is devoted to the implementation and numerical experiments of using our convergent 3-block sPADMM for solving (SDP). We conclude our paper in the final section.

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

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

For a given convex function $\phi : \mathcal{X} \to (-\infty, \infty]$, ϕ^* denotes its Fenchel conjugate, i.e.,

$$\phi^*(x) := \sup\{\langle y, x \rangle - \phi(y)\}, \quad x \in \mathcal{X}.$$

2. Preliminaries. Throughout this paper, we assume that $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ are three finite-dimensional real Euclidean spaces each equipped with an inner product $\langle \cdot, \cdot \rangle$ and its induced norm $\| \cdot \|$. Let $f : \mathcal{Y} \mapsto (-\infty, +\infty]$, $g : \mathcal{Z} \mapsto (-\infty, +\infty]$ be given closed proper convex functions, $\mathcal{F} : \mathcal{X} \mapsto \mathcal{Y}$, $\mathcal{G} : \mathcal{X} \mapsto \mathcal{Z}$ be given linear maps, and $c \in \mathcal{X}$ be also given. Let $\mathcal{S}$ and $\mathcal{T}$ be two given self-adjoint positive semidefinite (not necessarily positive definite) linear operators on $\mathcal{Y}$ and $\mathcal{Z}$, respectively.

The purpose of this section is to discuss the convergent properties of a sPADMM for solving the 2-block convex optimization problem

$$(2.1) \qquad \min \{f(y) + g(z) \mid \mathcal{F}^*y + \mathcal{G}^*z = c\}$$

and its dual

$$(2.2) \qquad \max \{-\langle c, x \rangle - f^*(-\mathcal{F}x) - g^*(-\mathcal{G}x)\},$$

which can equivalently be written as

$$(2.3) \qquad \min \{\langle c, x \rangle + f^*(u) + g^*(v) \mid \mathcal{F}x + u = 0, \ \mathcal{G}x + v = 0\}.$$

These convergence results are the essential ingredients in proving the convergence of our 3-block sPADMM proposed in section 3.

Recall that the augmented Lagrange function for problem (2.1) is defined by

$$(2.4) \quad L_\sigma(y, z; x) = f(y) + g(z) + \langle x, \mathcal{F}^*y + \mathcal{G}^*z - c \rangle + \frac{\sigma}{2}\|\mathcal{F}^*y + \mathcal{G}^*z - c\|^2.$$

It is clear that in order for the classic 2-block ADMM scheme (1.6)–(1.8) applied to problem (2.1) to work, one needs to assume that both subproblems have a solution. The existence of solutions for the subproblems can be guaranteed if we assume that the objective functions in (1.6) and (1.7) are both coercive. However, conditions ensuring the boundedness of the two generated sequences $\{y^{k+1}\}$ and $\{z^{k+1}\}$ are very subtle, while the boundedness of the dual variable sequence $\{x^{k+1}\}$ is readily obtainable. More importantly, it is also desirable that both y^{k+1} and z^{k+1} can be computed relative easily if f and g have conducive structures. In this regard, the following sPADMM is preferred.

ALGORITHM sPADMM2. A GENERIC 2-BLOCK SEMIPROXIMAL ADMM FOR SOLVING (2.1).

Let $\sigma > 0$ and $\tau \in (0, \infty)$ be given parameters. Choose $y^0 \in \text{dom}(f)$, $z^0 \in \text{dom}(g)$, and $x^0 \in \mathcal{X}$. Perform the kth iteration as follows:
Step 1. Compute $y^{k+1} = \arg\min L_\sigma(y, z^k; x^k) + \frac{\sigma}{2}\|y - y^k\|_{\mathcal{S}}^2$.
Step 2. Compute $z^{k+1} = \arg\min L_\sigma(y^{k+1}, z; x^k) + \frac{\sigma}{2}\|z - z^k\|_{\mathcal{T}}^2$.
Step 3. Compute $x^{k+1} = x^k + \tau\sigma(\mathcal{F}^*y^{k+1} + \mathcal{G}^*z^{k+1} - c)$.

In the above 2-block sPADMM for solving problem (2.1), the choices of $\mathcal{S}$ and $\mathcal{T}$ are very much problem dependent. The general principle is that both $\mathcal{S}$ and $\mathcal{T}$ should be as small as possible, while y^{k+1} and z^{k+1} are still relatively easy to compute. The convergence analysis of the 2-block sPADMM can be conducted by following the proof given by Fortin and Glowinski [10] based on variational analysis. This has been done in [9]. Here we will only summarize what we need for later developments. For details, see Appendix B in [9].

Assumption 2.1. There exists $(\hat{y}, \hat{z}) \in \text{ri}(\text{dom} f \times \text{dom} g)$ such that $\mathcal{F}^*\hat{y} + \mathcal{G}^*\hat{z} = c$.

Under Assumption 2.1, it follows from [33, Corollary 28.2.2] and [33, Corollary 28.3.1] that $(\bar{y}, \bar{z}) \in \mathcal{Y} \times \mathcal{Z}$ is an optimal solution to problem (2.1) if and only if there exists a Lagrange multiplier $\bar{x} \in \mathcal{X}$ such that

$$(2.5) \qquad 0 \in \mathcal{F}\bar{x} + \partial f(\bar{y}), \quad 0 \in \mathcal{G}\bar{x} + \partial g(\bar{z}), \quad \mathcal{F}^*\bar{y} + \mathcal{G}^*\bar{z} - c = 0,$$

where ∂f and ∂g are the subdifferential mappings of f and g, respectively. Moreover, any $\bar{x} \in \mathcal{X}$ satisfying (2.5) is an optimal solution to the dual problem (2.3). Since both ∂f and ∂g are maximal monotone [35, Theorem 12.17], there exist two self-adjoint and positive semidefinite operators Σ_f and Σ_g such that for all $y, y' \in \text{dom}(f)$, $u \in \partial f(y)$, and $u' \in \partial f(y')$,

$$(2.6) \qquad \langle u - u', y - y' \rangle \geq \|y - y'\|_{\Sigma_f}^2,$$

and for all $z, z' \in \text{dom}(g)$, $v \in \partial g(z)$, and $v' \in \partial g(z')$,

$$(2.7) \qquad \langle v - v', z - z' \rangle \geq \|z - z'\|_{\Sigma_g}^2.$$

THEOREM 2.2. *Let Σ_f and Σ_g be the two self-adjoint and positive semidefinite operators defined by (2.6) and (2.7), respectively. Suppose that the solution set of problem (2.1) is nonempty and that Assumption 2.1 holds. Assume that $\mathcal{S}$ and $\mathcal{T}$ are chosen such that the sequence $\{(y^k, z^k, x^k)\}$ generated by Algorithm sPADMM2 is*

well defined. Let $(\bar{y}, \bar{z})$ be any optimal solution to problem (2.1) *and $\bar{x}$ be any optimal solution to problem* (2.3), *respectively. For $k = 0, 1, 2, \ldots$, denote*

$$y_e^k := y^k - \bar{y}, \quad z_e^k := z^k - \bar{z}, \quad \text{and} \quad x_e^k := x^k - \bar{x}.$$

Then, under the condition either (a) $\tau \in (0, (1 + \sqrt{5})/2)$ *or* (b) $\tau \geq (1 + \sqrt{5})/2$ *but $\sum_{k=0}^{\infty} (\|\mathcal{G}^*(z^{k+1} - z^k)\|^2 + \tau^{-1}\|\mathcal{F}^*y^{k+1} + \mathcal{G}^*z^{k+1} - c\|^2) < \infty$, the following results hold:*

(i) *The sequence $\{\|x_e^{k+1}\|^2 + \|z_e^{k+1}\|_{(\sigma^{-1}\Sigma_g + \mathcal{T} + \mathcal{G}\mathcal{G}^*)}^2 + \|y_e^{k+1}\|_{(\sigma^{-1}\Sigma_f + \mathcal{S} + \mathcal{F}\mathcal{F}^*)}^2\}$ is bounded.*

(ii) *If $(y^\infty, z^\infty, x^\infty)$ is an accumulation point of $\{(y^k, z^k, x^k)\}$, then (y^∞, z^∞) solves* (2.1) *and x^∞ solves* (2.3), *respectively, and it holds that*

$$\lim_{k \to \infty} \left(\|x_e^{k+1}\|^2 + \|z_e^{k+1}\|_{(\sigma^{-1}\Sigma_g + \mathcal{T} + \mathcal{G}\mathcal{G}^*)}^2 + \|y_e^{k+1}\|_{(\sigma^{-1}\Sigma_f + \mathcal{S} + \mathcal{F}\mathcal{F}^*)}^2 \right) = 0,$$

where in the definition of (y_e^k, z_e^k, x_e^k), $(\bar{y}, \bar{z}, \bar{x})$ is replaced by $(y^\infty, z^\infty, x^\infty)$.

(iii) *If both $\sigma^{-1}\Sigma_f + \mathcal{S} + \mathcal{F}\mathcal{F}^*$ and $\sigma^{-1}\Sigma_g + \mathcal{T} + \mathcal{G}\mathcal{G}^*$ are positive definite, then the sequence $\{(y^k, z^k, x^k)\}$, which is automatically well defined, converges to a unique limit, say, $(y^\infty, z^\infty, x^\infty)$ with (y^∞, z^∞) solving* (2.1) *and x^∞ solving* (2.3), *respectively.*

(iv) *When the z-part disappears, the corresponding results in parts* (i)–(iii) *hold under the condition either $\tau \in (0, 2)$ or $\tau \geq 2$ but $\sum_{k=0}^{\infty} \|\mathcal{F}^*y^{k+1} - c\|^2 < \infty$.*

Remark 2.3. The conclusions of Theorem 2.2 for the case that $\tau \in (0, (1+\sqrt{5})/2)$ follow directly from the results given in [9, Theorem B.1]. For the case that $\tau \geq (1+\sqrt{5})/2$ but $\sum_{k=0}^{\infty}(\|\mathcal{G}^*(z^{k+1}-z^k)\|^2 + \tau^{-1}\|\mathcal{F}^*y^{k+1} + \mathcal{G}^*z^{k+1} - c\|^2) < \infty$, we can just mimic the proofs for part (c) in [9, Theorem B.1] for the case that $\tau \in (1, (1+\sqrt{5})/2)$ by using part (b) in [9, Theorem B.1] and the property on Fejér monotone sequences. Similarly, the conclusions for part (iv) can be derived correspondingly by using part (d) in [9, Theorem B.1]. In our numerical computations, we always start with a larger τ, e.g., $\tau = 1.95$, and reset it as $\tau := \max(\rho\tau, 1.618)$ for some $\rho \in (0, 1)$ if at the kth iteration

$$\|\mathcal{G}^*(z^{k+1} - z^k)\|^2 + \tau^{-1}\|\mathcal{F}^*y^{k+1} + \mathcal{G}^*z^{k+1} - c\|^2 > c_0 k^{-1.2}$$

for some constant $c_0 > 0$. Since τ can be reset for a finite number of times only, eventually either condition (a) or condition (b) in Theorem 2.2 is satisfied. Consequently, the conclusions of parts (i)–(iii) in Theorem 2.2 hold. When the z-part disappears, we can start with $\tau \geq 2$ and reset it accordingly by using a procedure similar to the above.

Remark 2.4. Independent of Fazel et al. [9], Deng and Yin [5] also analyzed the global convergence of Algorithm sPADMM2, though the focus of [5] is mainly on analyzing the rate of convergence for the following cases: (i) $\mathcal{S} \succeq 0$, $\mathcal{T} \succeq 0$, $\tau \in (0, 1]$; (ii) $\mathcal{S} \equiv 0$, $\mathcal{T} \succeq 0$, $\tau \in [1, \frac{1+\sqrt{5}}{2})$. The most interesting case used in this paper of taking $\mathcal{S} \succsim 0$, $\mathcal{T} \succsim 0$ and $\tau > 1$ (in particular, $\tau = 1.618$) is not covered by [5].

3. A convergent 3-block semiproximal ADMM. Assume that $\mathcal{W}$ is a finite-dimensional real Euclidean space. Let $h : \mathcal{W} \mapsto (-\infty, +\infty]$ be a given closed proper convex function and $\mathcal{H} : \mathcal{X} \mapsto \mathcal{W}$ be a given linear map. For the subsequent discussions, we let $\mathcal{T}_f$ and $\mathcal{T}_g$ be two given self-adjoint positive semidefinite (not necessarily positive definite) linear operators on $\mathcal{Y}$ and $\mathcal{Z}$, respectively.

Consider the following 3-block convex optimization problem:

$$(3.1) \qquad \min \left\{ f(y) + g(z) + h(w) \mid \mathcal{F}^* y + \mathcal{G}^* z + \mathcal{H}^* w = c \right\}.$$

The dual of (3.1) is

$$(3.2) \qquad \max \left\{ -\langle c, x \rangle - f^*(-\mathcal{F}x) - g^*(-\mathcal{G}x) - h^*(-\mathcal{H}x) \right\},$$

which can equivalently be written as

$$(3.3) \qquad \min \left\{ \langle c, x \rangle + f^*(u) + g^*(v) + h^*(s) \mid \mathcal{F}x + u = 0,\ \mathcal{G}x + v = 0,\ \mathcal{H}x + s = 0 \right\}.$$

By noting that the three variables u, v, and s are decoupled in the constraints of problem (3.3), one may attempt to apply the classic 2-block ADMM if $\mathcal{F}^*\mathcal{F} + \mathcal{G}^*\mathcal{G} + \mathcal{H}^*\mathcal{H}$ is positive definite or the 2-block sPADMM if it is only positive semidefinite (to avoid the potential nonconvergence of the former), to (3.3) with x and (u, v, s) as two separate block variables.[4] However, as far as we know from our numerical experiments, this approach is less efficient than working with the problem of the form (3.1) directly. A possible explanation for this phenomenon is that the sizes of $\mathcal{F}^*\mathcal{F} + \mathcal{G}^*\mathcal{G} + \mathcal{H}^*\mathcal{H}$ are often too large to admit an efficient Cholesky factorization and consequently one is forced to add a large semiproximal term to it to make the new operator more amenable for practical computations. At least, this is the case for conic programming (P).

In the next subsection, we will introduce our approach first for the case where h is a linear function and prove the convergence of our approach by relating it to a particularly designed 2-block sPADMM. For (D), this corresponds to the case where $m_I = 0$ or $\mathcal{K}_p = \mathcal{X}$. After that, we will extend our idea to deal with the case where h is only partially linear or fully nonlinear and the q-block case in section 3.2.

3.1. The case where h is linear. In this subsection, we are particularly interested in the case where h is a linear function of the form

$$(3.4) \qquad h(w) := -\langle b, w \rangle \quad \forall\, w \in \mathcal{W},$$

where $b \in \mathcal{W}$ is given. For simplicity, by removing the redundancy if necessary (although it may not be an easy task numerically), we assume that $\mathcal{H}\mathcal{H}^*$ is invertible, i.e., $\mathcal{H} : \mathcal{X} \to \mathcal{W}$ is surjective. Note that since $\mathcal{H} : \mathcal{X} \to \mathrm{Range}(\mathcal{H})$, the range space of $\mathcal{H}$, is always surjective, one may always redefine $\mathcal{W} := \mathrm{Range}(\mathcal{H})$ to make sure that $\mathcal{H}\mathcal{H}^* : \mathcal{W} \to \mathcal{W}$ is positive definite. However, it is a challenging task to compute the Cholesky decomposition of $\mathcal{H}\mathcal{H}^*$ by working in the range space of $\mathcal{H}$ if $\mathcal{W} \neq \mathrm{Range}(\mathcal{H})$.

For a given $\sigma > 0$, let $L_\sigma(y, z, w; x)$ be the augmented Lagrange function for (3.1), i.e., for any $(y, z, w, x) \in \mathcal{Y} \times \mathcal{Z} \times \mathcal{W} \times \mathcal{X}$,

$$L_\sigma(y, z, w; x) = f(y) + g(z) - \langle b, w \rangle + \langle x, \mathcal{F}^* y + \mathcal{G}^* z + \mathcal{H}^* w - c \rangle$$
$$(3.5) \qquad\qquad + \frac{\sigma}{2} \| \mathcal{F}^* y + \mathcal{G}^* z + \mathcal{H}^* w - c \|^2.$$

The following constraint qualification is needed for our subsequent discussions.

Assumption 3.1. There exists $(\hat{y}, \hat{z}, \hat{w}) \in \mathrm{ri}(\mathrm{dom}\, f \times \mathrm{dom}\, g) \times \mathcal{W}$ such that $\mathcal{F}^*\hat{y} + \mathcal{G}^*\hat{z} + \mathcal{H}^*\hat{w} = c$.

---

[4]This comment can be directly applied to the q-block convex optimization problem (1.1).

Similar to the discussion in section 2, under Assumption 3.1, it follows from [33, Corollary 28.2.2] and [33, Corollary 28.3.1] that $(\bar{y}, \bar{z}, \bar{w}) \in \mathcal{Y} \times \mathcal{Z} \times \mathcal{W}$ is an optimal solution to problem (3.1) if and only if there exists a Lagrange multiplier $\bar{x} \in \mathcal{X}$ such that

$$(3.6) \quad 0 \in \mathcal{F}\bar{x} + \partial f(\bar{y}), \quad 0 \in \mathcal{G}\bar{x} + \partial g(\bar{z}), \quad \mathcal{H}\bar{x} - b = 0, \quad \mathcal{F}^*\bar{y} + \mathcal{G}^*\bar{z} + \mathcal{H}^*\bar{w} - c = 0.$$

Moreover, any $\bar{x} \in \mathcal{X}$ satisfying (3.6) is an optimal solution to the dual problem (3.3).

We consider the following sPADMM for solving (3.1).

ALGORITHM sPADMM3c. A CONVERGENT 3-BLOCK SEMIPROXIMAL ADMM FOR SOLVING (3.1).

Let $\sigma > 0$ and $\tau \in (0, \infty)$ be given parameters. Choose $y^0 \in \text{dom}(f)$, $z^0 \in \text{dom}(g)$, and $x^0 \in \mathcal{X}$ such that $\mathcal{H}x^0 = b$. Set $w^0 := (\mathcal{H}\mathcal{H}^*)^{-1}\mathcal{H}(c - \mathcal{F}^*y^0 - \mathcal{G}^*z^0)$. Perform the kth iteration as follows:

Step 1. Compute $y^{k+1} = \arg\min L_\sigma(y, z^k, w^k; x^k) + \frac{\sigma}{2}\|y - y^k\|_{\mathcal{T}_f}^2$.

Step 2. Compute $w^{k+\frac{1}{2}} = \arg\min L_\sigma(y^{k+1}, z^k, w; x^k) = (\mathcal{H}\mathcal{H}^*)^{-1}\mathcal{H}(c - \mathcal{F}^*y^{k+1} - \mathcal{G}^*z^k)$ and $z^{k+1} = \arg\min L_\sigma(y^{k+1}, z, w^{k+\frac{1}{2}}; x^k) + \frac{\sigma}{2}\|z - z^k\|_{\mathcal{T}_g}^2$.

Step 3. Compute $w^{k+1} = \arg\min L_\sigma(y^{k+1}, z^{k+1}, w; x^k) = (\mathcal{H}\mathcal{H}^*)^{-1}\mathcal{H}(c - \mathcal{F}^*y^{k+1} - \mathcal{G}^*z^{k+1})$.

Step 4. Compute $x^{k+1} = x^k + \tau\sigma(\mathcal{F}^*y^{k+1} + \mathcal{G}^*z^{k+1} + \mathcal{H}^*w^{k+1} - c)$.

Note that in Step 2 of Algorithm sPADMM3c, by direct calculations we should have

$$w^{k+\frac{1}{2}} = \arg\min L_\sigma(y^{k+1}, z^k, w; x^k) = (\mathcal{H}\mathcal{H}^*)^{-1}[\mathcal{H}(c - \mathcal{F}^*y^{k+1} - \mathcal{G}^*z^k) + \sigma^{-1}(b - \mathcal{H}x^k)].$$

However, by using Proposition 3.2, to be introduced later, we know that $b - \mathcal{H}x^k = 0$ for all k. Thus $w^{k+\frac{1}{2}} = (\mathcal{H}\mathcal{H}^*)^{-1}\mathcal{H}(c - \mathcal{F}^*y^{k+1} - \mathcal{G}^*z^k)$. In Step 3, w^{k+1} is computed in a similar way. When $\mathcal{T}_f = 0$ and $\mathcal{T}_g = 0$, i.e., the proximal terms $\|y - y^k\|_{\mathcal{T}_f}^2$ and $\|z - z^k\|_{\mathcal{T}_g}^2$ are absent, Algorithm sPADMM3c will become our convergent ADMM for solving (3.1) (ADMM3c). One reason for including $\mathcal{T}_f$ and $\mathcal{T}_g$ is to ensure that both y^{k+1} and z^{k+1} are well defined; see further discussions on this part in section 2. The difference between our ADMM3c and the directly extended 3-block ADMM (ADMM3d) is that we perform an extra intermediate step to compute $w^{k+\frac{1}{2}}$ before computing z^{k+1}, i.e., at the kth iteration we perform a particularly chosen essentially BCD cycle in updating the variable y, z, w in the terminology of Tseng [40]. Except for this extra step, ADMM3c is as simple as ADMM3d, which at each iteration performs a Gauss–Seidel BCD cycle in updating the variable y, z, w. Observe that in both ADMM3c and ADMM3d, we need to solve linear systems involving the fixed operator $\mathcal{H}\mathcal{H}^*$. For the case where the computation (which only needs to be done once) of $\mathcal{H}\mathcal{H}^*$ and its (sparse) Cholesky factorization can be done at a moderate cost, Step 3 of the above algorithm can be performed cheaply. Now, under the condition that the Cholesky factorization of $\mathcal{H}\mathcal{H}^*$ is available, the extra cost for computing $w^{k+\frac{1}{2}}$ is actually insignificant. The reward for doing the extra step in computing $w^{k+\frac{1}{2}}$ is that we are able to prove the convergence of our ADMM3c not only for $\tau = 1$ but also allow τ to take a larger step-length, e.g., $\tau = 1.618$, so as to achieve faster convergence than the directly extended ADMM3d. Note that if the z-part disappears, then Step 2 of Algorithm sPADMM3c disappears and our ADMM3c is identical to

the classic 2-block ADMM but with $\tau \in (0, 2)$ instead of $\tau \in (0, (1 + \sqrt{5})/2)$ due to our requirement that $\mathcal{H}x^0 = b$.

Next we will prove the convergence of Algorithm sPADMM3c for solving (3.1) by relating it to the generic 2-block sPADMM for solving a 2-block convex optimization problem discussed in section 2.

For problem (3.1), one can obtain w explicitly from the equality constraint $\mathcal{F}^*y + \mathcal{G}^*z + \mathcal{H}^*w = c$ as follows:

$$(3.7) \qquad w(y, z) = (\mathcal{H}\mathcal{H}^*)^{-1}\mathcal{H}(c - \mathcal{F}^*y - \mathcal{G}^*z), \quad (y, z) \in \mathcal{Y} \times \mathcal{Z}.$$

Substituting (3.7) into (3.1), we can recast (3.1) equivalently as

$$(3.8) \qquad \min \left\{ f(y) + g(z) + \langle \bar{b}, \mathcal{F}^*y + \mathcal{G}^*z - c \rangle \mid \mathcal{Q}(\mathcal{F}^*y + \mathcal{G}^*z - c) = 0 \right\},$$

where $\bar{b} := \mathcal{H}^*(\mathcal{H}\mathcal{H}^*)^{-1}b$,

$$\mathcal{Q} := \mathcal{I} - \mathcal{P}, \qquad \mathcal{P} := \mathcal{H}^*(\mathcal{H}\mathcal{H}^*)^{-1}\mathcal{H},$$

and $\mathcal{I} : \mathcal{X} \to \mathcal{X}$ is the identity map. It is easy to check that the two operators $\mathcal{Q}$ and $\mathcal{P}$ satisfy the following properties:

$$(3.9)$$
$$\mathcal{P}^* = \mathcal{P}, \quad \mathcal{Q}^* = \mathcal{Q}, \quad \mathcal{P}^*\mathcal{P} = \mathcal{P}, \quad \mathcal{Q}^*\mathcal{Q} = \mathcal{Q}, \quad \mathcal{P}\mathcal{H}^* = \mathcal{H}^*, \quad \mathcal{H}\mathcal{Q} = 0, \quad \mathcal{Q}\mathcal{H}^* = 0.$$

The dual of (3.8) is given by

$$(3.10) \qquad \begin{aligned} \min \quad & f^*(u) + g^*(v) + \langle c, \bar{b} + \mathcal{Q}\lambda \rangle \\ \text{s.t.} \quad & \mathcal{F}(\bar{b} + \mathcal{Q}\lambda) + u = 0, \quad \mathcal{G}(\bar{b} + \mathcal{Q}\lambda) + v = 0. \end{aligned}$$

Note that (3.10) is equivalent to (3.3) if we let $x = \bar{b} + \mathcal{Q}\lambda$.

Let $\sigma > 0$ be a positive constant. Define the augmented Lagrange function for (3.8) by

$$\begin{aligned} \widehat{L}_\sigma(y, z; \lambda) = {} & f(y) + g(z) + \langle \bar{b}, \mathcal{F}^*y + \mathcal{G}^*z - c \rangle \\ (3.11) \qquad & + \langle \lambda, \mathcal{Q}(\mathcal{F}^*y + \mathcal{G}^*z - c) \rangle + \frac{\sigma}{2}\|\mathcal{Q}(\mathcal{F}^*y + \mathcal{G}^*z - c)\|^2. \end{aligned}$$

Now we can apply the generic 2-block sPADMM discussed in section 2 to (3.8).

ALGORITHM SPADMM2S. A SPECIFIC 2-BLOCK SEMIPROXIMAL ADMM FOR SOLVING (3.8).

Let $\sigma > 0$ and $\tau \in (0, \infty)$ be given parameters. Choose $y^0 \in \mathrm{dom}(f)$, $z^0 \in \mathrm{dom}(g)$, and $\lambda^0 \in \mathrm{Range}(\mathcal{Q})$. Perform the kth iteration as follows:

Step 1. Compute $y^{k+1} = \arg\min \widehat{L}_\sigma(y, z^k; \lambda^k) + \frac{\sigma}{2}\|\mathcal{F}^*(y - y^k)\|_{\mathcal{P}}^2 + \frac{\sigma}{2}\|y - y^k\|_{\mathcal{T}_f}^2$.

Step 2. Compute $z^{k+1} = \arg\min \widehat{L}_\sigma(y^{k+1}, z; \lambda^k) + \frac{\sigma}{2}\|\mathcal{G}^*(z - z^k)\|_{\mathcal{P}}^2 + \frac{\sigma}{2}\|z - z^k\|_{\mathcal{T}_g}^2$.

Step 3. Compute $\lambda^{k+1} = \lambda^k + \tau\sigma\mathcal{Q}(\mathcal{F}^*y^{k+1} + \mathcal{G}^*z^{k+1} - c)$.

It is important to note that in Algorithm sPADMM2s for solving (3.8), we have two proximal terms in both Step 1 and Step 2 instead of one proximal term. In particular, the first proximal term is necessary as neither $\mathcal{Q}\mathcal{F}^*$ nor $\mathcal{Q}\mathcal{G}^*$ is injective.

Next, we establish the equivalence of Algorithm sPADMM3c for solving problem (3.1) and Algorithm sPADMM2s for solving problem (3.8).

PROPOSITION 3.2. *Let $\sigma > 0$ and $\tau \in (0, \infty)$ be given parameters. Choose $y^0 \in \mathrm{dom}(f)$, $z^0 \in \mathrm{dom}(g)$ and $\lambda^0 \in \mathrm{Range}(\mathcal{Q})$. Let $x^0 = \mathcal{H}^*(\mathcal{H}\mathcal{H}^*)^{-1}b + \lambda^0$. Then for any $k \geq 0$, we have the following results:*

(i) *The point (y^k, z^k) generated by Algorithm* sPADMM2s *for solving problem* (3.8) *is identical to the point (y^k, z^k) generated by Algorithm* sPADMM3c *for solving problem* (3.1).

(ii) *λ^k and x^k satisfy the relation*

$$x^k = \mathcal{H}^*(\mathcal{H}\mathcal{H}^*)^{-1}b + \lambda^k.$$

Proof. We prove this proposition by induction. First, note that since $\mathcal{H}\mathcal{Q} = 0$ and $\lambda^0 \in \mathrm{Range}(\mathcal{Q})$, we have $\mathcal{H}x^0 = b$.

Recall that $w^k = (\mathcal{H}\mathcal{H}^*)^{-1}\mathcal{H}(c - \mathcal{F}^*y^k - \mathcal{G}^*z^k)$. Note that by direct computations we have $\mathcal{H}^*w^k = \mathcal{P}\mathcal{H}^*w^k = \mathcal{P}(c - \mathcal{F}^*y^k - \mathcal{G}^*z^k)$. Assume that the conclusions of this proposition hold up to $k \geq 0$. Then, by using (3.9) and the fact that λ^k is in the range of $\mathcal{Q}$ and $\mathcal{H}x^k = b$, we can easily check for every $k = 0, 1, \ldots$ that

$$y^{k+1} = \arg\min \left\{ \widehat{L}_\sigma(y, z^k; \lambda^k) + \frac{\sigma}{2}\|\mathcal{F}^*(y - y^k)\|_{\mathcal{P}}^2 + \frac{\sigma}{2}\|y - y^k\|_{\mathcal{T}_f}^2 \right\}$$

$$= \arg\min \left\{ \widehat{L}_\sigma(y, z^k; \lambda^k) + \frac{\sigma}{2}\|\mathcal{P}(\mathcal{F}^*y + \mathcal{G}^*z^k + \mathcal{H}^*w^k - c)\|^2 + \frac{\sigma}{2}\|y - y^k\|_{\mathcal{T}_f}^2 \right\}$$

$$= \arg\min \left\{ \begin{array}{l} f(y) + g(z^k) + \langle \bar{b}, \mathcal{F}^*y + \mathcal{G}^*z^k - c\rangle + \langle \lambda^k, \mathcal{Q}(\mathcal{F}^*y + \mathcal{G}^*z^k - c)\rangle \\ + \frac{\sigma}{2}\|\mathcal{Q}(\mathcal{F}^*y + \mathcal{G}^*z^k - c)\|^2 + \frac{\sigma}{2}\|\mathcal{P}(\mathcal{F}^*y + \mathcal{G}^*z^k + \mathcal{H}^*w^k - c)\|^2 + \frac{\sigma}{2}\|y - y^k\|_{\mathcal{T}_f}^2 \end{array} \right\}$$

$$= \arg\min \left\{ \begin{array}{l} f(y) + g(z^k) + \langle x^k, \mathcal{F}^*y + \mathcal{G}^*z^k - c\rangle + \frac{\sigma}{2}\|\mathcal{Q}(\mathcal{F}^*y + \mathcal{G}^*z^k + \mathcal{H}^*w^k - c)\|^2 \\ + \frac{\sigma}{2}\|\mathcal{P}(\mathcal{F}^*y + \mathcal{G}^*z^k + \mathcal{H}^*w^k - c)\|^2 + \frac{\sigma}{2}\|y - y^k\|_{\mathcal{T}_f}^2 \end{array} \right\}$$

$$= \arg\min \left\{ \begin{array}{l} f(y) + g(z^k) - \langle b, w^k\rangle + \langle x^k, \mathcal{F}^*y + \mathcal{G}^*z^k + \mathcal{H}^*w^k - c\rangle + \frac{\sigma}{2}\|y - y^k\|_{\mathcal{T}_f}^2 \\ + \frac{\sigma}{2}\|\mathcal{Q}(\mathcal{F}^*y + \mathcal{G}^*z^k + \mathcal{H}^*w^k - c)\|^2 + \frac{\sigma}{2}\|\mathcal{P}(\mathcal{F}^*y + \mathcal{G}^*z^k + \mathcal{H}^*w^k - c)\|^2 \end{array} \right\}$$

$$= \arg\min \left\{ \begin{array}{l} f(y) + g(z^k) - \langle b, w^k\rangle + \langle x^k, \mathcal{F}^*y + \mathcal{G}^*z^k + \mathcal{H}^*w^k - c\rangle \\ + \frac{\sigma}{2}\|\mathcal{F}^*y + \mathcal{G}^*z^k + \mathcal{H}^*w^k - c\|^2 + \frac{\sigma}{2}\|y - y^k\|_{\mathcal{T}_f}^2 \end{array} \right\}$$

$$= \arg\min \left\{ L_\sigma(y, z^k, w^k; x^k) + \frac{\sigma}{2}\|y - y^k\|_{\mathcal{T}_f}^2 \right\}.$$

Similarly, for every $k = 0, 1, \ldots$, we have

$$z^{k+1} = \arg\min \left\{ \widehat{L}_\sigma(y^{k+1}, z; \lambda^k) + \frac{\sigma}{2}\|\mathcal{G}^*(z - z^k)\|_{\mathcal{P}}^2 + \frac{\sigma}{2}\|z - z^k\|_{\mathcal{T}_g}^2 \right\}$$

$$= \arg\min \left\{ \widehat{L}_\sigma(y^{k+1}, z; \lambda^k) + \frac{\sigma}{2}\|\mathcal{P}(\mathcal{F}^*y^{k+1} + \mathcal{G}^*z + \mathcal{H}^*w^{k+\frac{1}{2}} - c)\|^2 + \frac{\sigma}{2}\|z - z^k\|_{\mathcal{T}_g}^2 \right\}$$

$$= \arg\min \left\{ L_\sigma(y^{k+1}, z, w^{k+\frac{1}{2}}; x^k) + \frac{\sigma}{2}\|z - z^k\|_{\mathcal{T}_g}^2 \right\},$$

where $w^{k+\frac{1}{2}} = \arg\min L_\sigma(y^{k+1}, z^k, w; x^k) = (\mathcal{H}\mathcal{H}^*)^{-1}\mathcal{H}(c - \mathcal{F}^*y^{k+1} - \mathcal{G}^*z^k)$ and

$$\lambda^{k+1} = \lambda^k + \tau\sigma\mathcal{Q}(\mathcal{F}^*y^{k+1} + \mathcal{G}^*z^{k+1} - c)$$

$$= x^k - \mathcal{H}^*(\mathcal{H}\mathcal{H}^*)^{-1}b + \tau\sigma(\mathcal{F}^*y^{k+1} + \mathcal{G}^*z^{k+1} + \mathcal{H}^*w^{k+1} - c)$$

$$= x^{k+1} - \mathcal{H}^*(\mathcal{H}\mathcal{H}^*)^{-1}b,$$

where $w^{k+1} = \arg\min L_\sigma(y^{k+1}, z^{k+1}, w; x^k) = (\mathcal{H}\mathcal{H}^*)^{-1}\mathcal{H}(c - \mathcal{F}^*y^{k+1} - \mathcal{G}^*z^{k+1})$. This completes our proof. $\square$

Now we are ready to establish the convergence results for Algorithm sPADMM3c for solving (3.1).

THEOREM 3.3. *Let Σ_f and Σ_g be the two self-adjoint and positive semidefinite operators defined by* (2.6) *and* (2.7), *respectively. Suppose that the solution set of problem* (3.1) *is nonempty and that Assumption* 3.1 *holds. Assume that $\mathcal{T}_f$ and $\mathcal{T}_g$ are chosen such that the sequence $\{(y^k, z^k, w^k, x^k)\}$ generated by Algorithm* sPADMM3c *is well defined. Let $(\bar{y}, \bar{z}, \bar{w})$ be any optimal solution to* (3.1) *and $\bar{x}$ be any optimal solution to* (3.3), *respectively. For $k = 0, 1, 2, \ldots$, denote*

$$y_e^k := y^k - \bar{y}, \quad z_e^k := z^k - \bar{z}, \quad w_e^k := w^k - \bar{w}, \quad \text{and} \quad x_e^k := x^k - \bar{x}.$$

Then, under the condition either (a) $\tau \in (0, (1 + \sqrt{5})/2)$ *or* (b) $\tau \geq (1 + \sqrt{5})/2$ *but $\sum_{k=0}^\infty (\|\mathcal{G}^*(z^{k+1} - z^k) + \mathcal{H}^*(w^{k+1} - w^{k+\frac{1}{2}})\|^2 + \tau^{-1}\|\mathcal{F}^*y^{k+1} + \mathcal{G}^*z^{k+1} + \mathcal{H}^*w^{k+1} - c\|^2) < \infty$, the following results hold:*

(i) *The sequence $\{\|x_e^{k+1}\|^2 + \|z_e^{k+1}\|_{(\sigma^{-1}\Sigma_g + \mathcal{T}_g + \mathcal{G}\mathcal{G}^*)}^2 + \|y_e^{k+1}\|_{(\sigma^{-1}\Sigma_f + \mathcal{T}_f + \mathcal{F}\mathcal{F}^*)}^2\}$ is bounded.*

(ii) *If $(y^\infty, z^\infty, w^\infty, x^\infty)$ is an accumulation point of $\{(y^k, z^k, w^k, x^k)\}$, then $(y^\infty, z^\infty, w^\infty)$ solves* (3.1) *and x^∞ solves* (3.3), *respectively, and it holds that*

$$\lim_{k\to\infty} \left(\|x_e^{k+1}\|^2 + \|z_e^{k+1}\|_{(\sigma^{-1}\Sigma_g + \mathcal{T}_g + \mathcal{G}\mathcal{G}^*)}^2 + \|y_e^{k+1}\|_{(\sigma^{-1}\Sigma_f + \mathcal{T}_f + \mathcal{F}\mathcal{F}^*)}^2\right) = 0,$$

where in the definition of $(y_e^k, z_e^k, w_e^k, x_e^k)$, the point $(\bar{y}, \bar{z}, \bar{w}, \bar{x})$ is replaced by $(y^\infty, z^\infty, w^\infty, x^\infty)$.

(iii) *If both $\sigma^{-1}\Sigma_f + \mathcal{T}_f + \mathcal{F}\mathcal{F}^*$ and $\sigma^{-1}\Sigma_g + \mathcal{T}_g + \mathcal{G}\mathcal{G}^*$ are positive definite, then the sequence $\{(y^k, z^k, w^k, x^k)\}$, which is automatically well defined, converges to a unique limit, say, $(y^\infty, z^\infty, w^\infty, x^\infty)$ with $(y^\infty, z^\infty, w^\infty)$ solving* (3.1) *and x^∞ solving* (3.3), *respectively.*

(iv) *When the the z-part disappears, the corresponding results in parts* (i)–(iii) *hold for any $\tau \in (0, 2)$ or $\tau \geq 2$ but $\sum_{k=0}^\infty \|\mathcal{F}^*y^{k+1} + \mathcal{H}^*w^{k+1} - c\|^2 < \infty$.*

Proof. By combing Theorem 2.2 with Proposition 3.2 and using the relation (3.7), we can readily obtain the conclusions of this theorem. $\square$

Remark 3.4. The main idea for proving the convergence of Algorithm sPADMM3c is via showing that Algorithm sPADMM3c is equivalent to Algorithm sPADMM2s, which is obtained by applying Algorithm sPADMM2 to the problem (3.8) using two special semiproximal terms $\mathcal{S} = \mathcal{F}\mathcal{P}\mathcal{F}^* + \mathcal{T}_f$ and $\mathcal{T} = \mathcal{G}\mathcal{P}\mathcal{G}^* + \mathcal{T}_g$ in Step 1 and Step 2, respectively. This simple discovery of the equivalence of Algorithm sPADMM3c for solving (3.1) and Algorithm sPADMM2s for solving its equivalent problem (3.8) is significant since on the one hand, it settles the convergence of Algorithm sPADMM3c by using known convergence results for Algorithm sPADMM2s, and on the other hand, it allows one to take advantage of the extremely simple structure of Algorithm sPADMM3c in searching for an efficient implementation for solving convex conic programming. Note that one cannot even prove the convergence of Algorithm ADMM3c (without the two semiproximal terms $\mathcal{T}_f$ and $\mathcal{T}_g$) by directly applying the classic 2-block ADMM to (3.8). Actually, since neither $\mathcal{Q}\mathcal{F}^*$ nor $\mathcal{Q}\mathcal{G}^*$ is injective, one cannot use the classic 2-block ADMM to solve (3.8) at all unless additional conditions on f and g are imposed.

3.2. Extensions. In this subsection, we first consider the 3-block convex optimization problem (3.1), i.e.,

$$\min \left\{ f(y) + g(z) + h(w) \mid \mathcal{F}^* y + \mathcal{G}^* z + \mathcal{H}^* w = c \right\}$$

for the case where at least one of the three functions f, g, and h is partially linear. Without loss of generality, we assume that $h : \mathcal{W} \equiv \mathcal{W}_I \times \mathcal{W}_E \to (-\infty, \infty]$ is of the partial linear structure

$$h(w) = \theta(w_I) - \langle b, w \rangle = \theta(w_I) - \langle b_I, w_I \rangle - \langle b_E, w_E \rangle \quad \forall\, w \equiv (w_I, w_E) \in \mathcal{W}_I \times \mathcal{W}_E,$$

where $b \equiv (b_I, b_E) \in \mathcal{W}_I \times \mathcal{W}_E$ is a given vector and $\theta : \mathcal{W}_I \to (-\infty, \infty]$ is a closed proper nonlinear convex function. Decompose $\mathcal{H} \equiv \mathcal{H}_I \times \mathcal{H}_E$ such that for any $x \in \mathcal{X}$,

$$\begin{pmatrix} \mathcal{H}_I x \\ \mathcal{H}_E x \end{pmatrix} \equiv \mathcal{H} x = \mathcal{H}_I x \times \mathcal{H}_E x \in \mathcal{W}_I \times \mathcal{W}_E.$$

Again, by removing redundancy in $\mathcal{H}$ if necessary, we assume that $\mathcal{H}_E \mathcal{H}_E^*$ is invertible, i.e., $\mathcal{H}_E$ is surjective. We also assume that the Cholesky factorization of $\mathcal{H}_E \mathcal{H}_E^*$ can be computed at a moderate cost. In order to apply our proposed convergent 3-block semiproximal ADMM to solve

$$(3.12) \qquad \min \left\{ f(y) + g(z) + h(w) \mid \mathcal{F}^* y + \mathcal{G}^* z + \mathcal{H}_I^* w_I + \mathcal{H}_E^* w_E = c \right\},$$

we need to convert it into the form of problem (3.1) with h being linear. For this purpose, we define $\mathcal{H}_3 : \mathcal{W} \to \mathcal{X} \times \mathcal{W}_I$ to be the following linear map:

$$\mathcal{H}_3^* w := \begin{pmatrix} \mathcal{H}_I^* w_I + \mathcal{H}_E^* w_E \\ -\mathcal{D}_I^* w_I \end{pmatrix} \quad \forall\, w \in \mathcal{W},$$

where $\mathcal{D}_I : \mathcal{W}_I \to \mathcal{W}_I$ is a particularly chosen nonsingular linear operator, e.g., the identity operator $\mathcal{I} : \mathcal{W}_I \to \mathcal{W}_I$, and $\mathcal{D}_I^*$ is the adjoint of $\mathcal{D}_I$.

We consider the following two cases.

Case 1. The inverse of $\mathcal{H}_3 \mathcal{H}_3^*$ can be computed at a moderate cost.

By introducing a slack variable $\eta \in \mathcal{W}_I$, we can rewrite (3.12) as

$$\min \left\{ f(y) + g(z) + \theta(w_I) - \langle b, w \rangle \mid \mathcal{F}^* y + \mathcal{G}^* z + \mathcal{H}_I^* w_I + \mathcal{H}_E^* w_E = c,\ \mathcal{D}_I^*(\eta - w_I) = 0 \right\},$$

which can then be cast into the form of problem (3.1) as follows:

$$(3.13) \quad \min \left\{ \left(f(y) + \theta(\eta) \right) + g(z) - \langle b, w \rangle \ \middle|\ \begin{pmatrix} \mathcal{F}^* y \\ \mathcal{D}_I^* \eta \end{pmatrix} + \begin{pmatrix} \mathcal{G}^* z \\ 0 \end{pmatrix} + \mathcal{H}_3^* w = \begin{pmatrix} c \\ 0 \end{pmatrix} \right\}.$$

The convergent 3-block sPADMM discussed in section 3.1 for solving problem (3.1) can then be applied to problem (3.13) in a straightforward way.

Case 2. The inverse of $\mathcal{H}_3 \mathcal{H}_3^*$ cannot be computed at a moderate cost.

Let $\mathcal{D} : \mathcal{X} \to \mathcal{X}$ be a given nonsingular linear operator and $\mathcal{D}^*$ be its adjoint. We assume that $\mathcal{D}$ is chosen in such a way that the inverse of $\mathcal{I} + \mathcal{D}\mathcal{D}^*$ can be computed explicitly with low costs (e.g., $\mathcal{D} = 5\mathcal{I}$). By introducing a slack variable $s \in \mathcal{X}$, we can rewrite (3.12) as

$$\min \left\{ f(y) + g(z) + \theta(w_I) - \langle b, w \rangle \mid \mathcal{F}^* y + \mathcal{G}^* z + s + \mathcal{H}_E^* w_E = c,\ \mathcal{D}^*(\mathcal{H}_I^* w_I - s) = 0 \right\},$$

which can then be recast in the form of (3.1) as follows:

$$
\begin{aligned}
\text{(3.14)} \qquad \min \quad & \big(f(y) + \tilde{\theta}(w_I)\big) + g(z) - \langle b_E, w_E \rangle \\
\text{s.t.} \quad & \begin{pmatrix} \mathcal{F}^* y \\ \mathcal{D}^* \mathcal{H}_I^* w_I \end{pmatrix} + \begin{pmatrix} \mathcal{G}^* z \\ 0 \end{pmatrix} + \mathcal{B}^*(s, w_E) = \begin{pmatrix} c \\ 0 \end{pmatrix},
\end{aligned}
$$

where the convex function $\tilde{\theta}(\cdot) \equiv \theta(\cdot) - \langle b_I, \cdot \rangle$ and the linear map $\mathcal{B} : \mathcal{X} \times \mathcal{W}_E \to \mathcal{X} \times \mathcal{X}$ is defined by

$$
\mathcal{B}^*(s, w_E) := \begin{pmatrix} s + \mathcal{H}_E^* w_E \\ -\mathcal{D}^* s \end{pmatrix} \quad \forall\, (s, w_E) \in \mathcal{X} \times \mathcal{W}_E.
$$

As in Case 1, we can apply the convergent 3-block sPADMM discussed in section 3.1 to problem (3.14) in a straightforward way as now the inverse of $\mathcal{B}\mathcal{B}^*$ can be computed based on $(\mathcal{I} + \mathcal{D}\mathcal{D}^*)^{-1}$ and the inverse of $\mathcal{H}_E(\mathcal{I} - (\mathcal{I} + \mathcal{D}\mathcal{D}^*)^{-1})\mathcal{H}_E^*$. In our numerical experiments in section 5, we choose $\mathcal{D} = \alpha \mathcal{I}$ for some $\alpha \in [3, 6]$, and α is dynamically adjusted according to the progress of the algorithm.

Though not the focus of this paper, here we will also briefly explain how to extend our convergent semiproximal ADMM to deal with the general q-block convex optimization problem (1.1):

$$
\min \left\{ \sum_{i=1}^q \theta_i(w_i) \,\middle|\, \sum_{i=1}^q \mathcal{B}_i^* w_i = c \right\}.
$$

For any $i \geq 3$, let $\mathcal{D}_i : \mathcal{X} \to \mathcal{X}$ be a given nonsingular linear operator and $\mathcal{D}_i^*$ be its adjoint. By introducing slack variables $s_i \in \mathcal{X}$, $i = 3, \ldots, q$, we can then rewrite (1.1) equivalently as

$$
\text{(3.15)}
$$
$$
\min \left\{ \sum_{i=1}^q \theta_i(w_i) \,\middle|\, \mathcal{B}_1^* w_1 + \mathcal{B}_2^* w_2 + s_3 + \cdots + s_q = c,\ \mathcal{D}_i^*(\mathcal{B}_i^* w_i - s_i) = 0,\ i = 3, \ldots, q \right\}.
$$

Let $l := \lfloor q/2 \rfloor$ be the largest integer that is smaller than or equal to $q/2$. Define $h(s_3, \ldots, s_q) \equiv 0$,

$$
f(w_1, w_3 \ldots, w_{l+1}) = \theta_1(w_1) + \sum_{i=3}^{l+1} \theta_i(w_i), \quad g(w_2, w_{l+2}, \ldots, w_q) = \theta(w_2) + \sum_{i=l+2}^{q} \theta_i(w_i),
$$

$$
\mathcal{F}^*(w_1, w_3, \ldots, w_{l+1}) = \begin{pmatrix} \mathcal{B}_1^* w_1 \\ \mathcal{D}_3^* \mathcal{B}_3^* w_3 \\ \vdots \\ \mathcal{D}_{l+1}^* \mathcal{B}_{l+1}^* w_{l+1} \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \mathcal{G}^*(w_2, w_{l+2}, \ldots, w_q) = \begin{pmatrix} \mathcal{B}_2^* w_2 \\ 0 \\ \vdots \\ 0 \\ \mathcal{D}_{l+2}^* \mathcal{B}_{l+2}^* w_{l+2} \\ \vdots \\ \mathcal{D}_q^* \mathcal{B}_q^* w_q \end{pmatrix}
$$

for any $w_i \in \mathcal{W}_i$, $i = 1, \ldots, q$, and $s_j \in \mathcal{X}$, $j = 3, \ldots, q$. Then it is easy to see that (3.15) is in the form of (3.1) with

$$
\mathcal{H}^*(s_3, \ldots, s_q) = \begin{pmatrix} s_3 + \cdots + s_q \\ -\mathcal{D}_3^* s_3 \\ \vdots \\ -\mathcal{D}_q^* s_q \end{pmatrix}.
$$

Note that we have

$$\mathcal{H}\mathcal{H}^* = \begin{pmatrix} \mathcal{D}_3\mathcal{D}_3^* & & \\ & \ddots & \\ & & \mathcal{D}_q\mathcal{D}_q^* \end{pmatrix} + \begin{pmatrix} \mathcal{I} \\ \vdots \\ \mathcal{I} \end{pmatrix} \begin{pmatrix} \mathcal{I} \\ \vdots \\ \mathcal{I} \end{pmatrix}^*.$$

Thus, if for each $i \in \{3, \ldots, q\}$, $\mathcal{D}_i$ is chosen such that $\mathcal{D}_i\mathcal{D}_i^*$ can be inverted easily, e.g., $\mathcal{D}_i = \alpha\mathcal{I}$ for some $\alpha > 0$, then we can compute the inverse of $\mathcal{H}\mathcal{H}^*$ analytically via the Sherman–Morrison–Woodbury formula if $\mathcal{I} + \sum_{i=3}^{q}(\mathcal{D}_i\mathcal{D}_i^*)^{-1}$ is also easy to invert. In this way, our convergent 3-block semiproximal ADMM discussed in section 3.1 can then be applied to problem (3.15) directly.

4. Applications to conic programming. In this section, we show how to apply our convergent 3-block sPADMM to solve conic programming (P) and its dual (D). Here we always assume that $\mathcal{A}_E\mathcal{A}_E^*$ is invertible and its Cholesky factorization can be computed at a moderate cost.

4.1. The case where $m_I = 0$. In this subsection, we show how our convergent ADMM3c can be used to solve conic programming (P) without the inequality constraints $\mathcal{A}_I x_I \geq b_I$. In this case, the conic programming (P) reduces to

$$(4.1) \qquad\qquad \max\left\{\langle -c, x\rangle \mid \mathcal{A}_E x = b_E, \ x \in \mathcal{K}, \ x \in \mathcal{K}_p\right\}$$

with its dual taking the form of

$$(4.2) \qquad\qquad \min\left\{\delta_{\mathcal{K}^*}(s) + \delta_{\mathcal{K}_p^*}(z) - \langle b_E, y_E\rangle \mid s + z + \mathcal{A}_E^* y_E = c\right\}.$$

For problem (4.2), instead of using the constraint qualification imposed in Assumption 3.1, we will use the following more familiar one in the conic programming field.

Assumption 4.1. (a) For problem (4.1), there exists a feasible solution $\hat{x} \in \mathcal{K}$ such that

$$\mathcal{A}_E\hat{x} = b_E, \ \hat{x} \in \text{int}(\mathcal{K}), \ \hat{x} \in \mathcal{K}_p.$$

(b) For problem (4.2), there exists a feasible solution $(\hat{s}, \hat{z}, \hat{y}_E) \in \mathcal{K}^* \times \mathcal{X} \times \Re^{m_E}$ such that

$$\hat{s} + \hat{z} + \mathcal{A}_E^* \hat{y}_E = c, \ \hat{s} \in \text{int}(\mathcal{K}^*), \ \hat{z} \in \mathcal{K}_p^*.$$

It is known from convex analysis (e.g., [1, Corollary 5.3.6]) that under Assumption 4.1, the strong duality for (4.1) and (4.2) holds and the following Karush–Kuhn–Tucker (KKT) condition has nonempty solutions:

$$(4.3) \qquad \begin{cases} \mathcal{A}_E x - b_E = 0, \quad s + z + \mathcal{A}_E^* y_E - c = 0, \\ \langle x, s\rangle = 0, \ x \in \mathcal{K}, \ s \in \mathcal{K}^*, \ \langle x, z\rangle = 0, \ x \in \mathcal{K}_p, \ z \in \mathcal{K}_p^*. \end{cases}$$

Let $\sigma > 0$ be given. The augmented Lagrange function for (4.2) is defined by

$$L_\sigma(s, z, y_E; x) := \delta_{\mathcal{K}^*}(s) + \delta_{\mathcal{K}_p^*}(z) + \langle -b_E, y_E\rangle + \langle x, s + z + \mathcal{A}_E^* y_E - c\rangle$$

$$(4.4) \qquad\qquad + \frac{\sigma}{2}\|s + z + \mathcal{A}_E^* y_E - c\|^2,$$

where $(s, z, y_E, x) \in \mathcal{X} \times \mathcal{X} \times \Re^{m_E} \times \mathcal{X}$.

We can apply our convergent 3-block ADMM (without the proximal terms) to problem (4.2) to obtain the following algorithm.

ALGORITHM CONIC-ADMM3C: A CONVERGENT 3-BLOCK ADMM FOR SOLVING (4.2).

Given parameters $\sigma > 0$ and $\tau \in (0, \infty)$. Choose $s^0 \in \mathcal{K}^*$, $z^0 \in \mathcal{K}_p^*$, $x^0 \in \mathcal{X}$ such that $\mathcal{A}_E x^0 = b_E$. Set $y_E^0 = (\mathcal{A}_E \mathcal{A}_E^*)^{-1} \mathcal{A}_E (c - s^0 - z^0)$. Perform the kth iteration as follows:

Step 1. Compute $s^{k+1} = \arg\min L_\sigma(s, z^k, y_E^k; x^k) = \Pi_{\mathcal{K}^*} \left(c - z^k - \mathcal{A}_E^* y_E^k - \sigma^{-1} x^k \right)$.

Step 2. Compute $y_E^{k+\frac{1}{2}} = \arg\min L_\sigma(s^{k+1}, z^k, y_E; x^k) = (\mathcal{A}_E \mathcal{A}_E^*)^{-1} \mathcal{A}_E (c - s^{k+1} - z^k)$ and

$$z^{k+1} = \arg\min L_\sigma(s^{k+1}, z, y_E^{k+\frac{1}{2}}; x^k) = \Pi_{\mathcal{K}_p^*} \left(c - s^{k+1} - \mathcal{A}_E^* y_E^{k+\frac{1}{2}} - \sigma^{-1} x^k \right).$$

Step 3. Compute $y_E^{k+1} = \arg\min L_\sigma(s^{k+1}, z^{k+1}, y_E; x^k) = (\mathcal{A}_E \mathcal{A}_E^*)^{-1} \mathcal{A}_E (c - s^{k+1} - z^{k+1})$.

Step 4. Compute $x^{k+1} = x^k + \tau\sigma(s^{k+1} + z^{k+1} + \mathcal{A}_E^* y_E^{k+1} - c)$.

The following convergence results for Algorithm Conic-ADMM3c for solving problem (4.2) can be derived directly from Theorem 3.3.

THEOREM 4.2. *Suppose that Assumption* 4.1 *holds and that $\mathcal{A}_E$ is surjective. Then the sequence $\{(s^k, z^k, y_E^k, x^k)\}$ generated by Algorithm* Conic-ADMM3c *is well defined. Furthermore, under the condition that either* (a) $\tau \in (0, (1 + \sqrt{5})/2)$ *or* (b) $\tau \geq (1 + \sqrt{5})/2$ *but $\sum_{k=0}^\infty (\|(z^{k+1} - z^k) + \mathcal{A}_E^*(y_E^{k+1} - y_E^{k+\frac{1}{2}})\|^2 + \tau^{-1}\|s^{k+1} + z^{k+1} + \mathcal{A}_E^* y_E^{k+1} - c\|^2) < \infty$, it holds that*
   (i) *the sequence $\{(s^k, z^k, y_E^k, x^k)\}$ converges to a unique limit, say, $(s^\infty, z^\infty, y_E^\infty, x^\infty)$ satisfying the KKT condition* (4.3);
   (ii) *when $\mathcal{K}_p = \mathcal{X}$, i.e., the z-part disappears, the corresponding result in part* (i) *holds for any $\tau \in (0, 2)$ or $\tau \geq 2$ but $\sum_{k=0}^\infty \|s^{k+1} + \mathcal{A}_E^* y_E^{k+1} - c\|^2 < \infty$.*

4.2. The case where $m_I > 0$. Here we consider conic programming problem (P) with inequality constraints, i.e.,

$$\text{(4.5)} \qquad \max\left\{ \langle -c, x \rangle \mid \mathcal{A}_E x = b_E,\ \mathcal{A}_I x \geq b_I,\ x \in \mathcal{K},\ x \in \mathcal{K}_p \right\}.$$

If m_I, the number of inequality constraints $\mathcal{A}_I x \geq b_I$, is relatively small, we can introduce a slack variable to convert (4.5) into the form of problem (4.1) with three blocks of constraints and then apply Algorithm Conic-ADMM3c introduced in section 4.1 to solve it. We omit the details here.

Next, we consider the case where m_I is large. The dual of (4.5) is given by

$$\text{(4.6)} \quad \min\left\{ \delta_{\mathcal{K}^*}(s) + \delta_{\Re_+^{m_I}}(y_I) + \delta_{\mathcal{K}_p^*}(z) - \langle b, y \rangle \mid s + \mathcal{A}_I^* y_I + z + \mathcal{A}_E^* y_E = c \right\}.$$

Let $\mathcal{D} : \mathcal{X} \to \mathcal{X}$ be a given nonsingular linear operator and $\mathcal{D}^*$ be its adjoint. In this case, we can rewrite (4.6) equivalently as

(4.7)
$$\min\left\{ \delta_{\mathcal{K}^*}(s) + \delta_{\Re_+^{m_I}}(y_I) + \delta_{\mathcal{K}_p^*}(u) - \langle b, y \rangle \mid s + \mathcal{A}_I^* y_I + z + \mathcal{A}_E^* y_E = c,\ \mathcal{D}^*(u - z) = 0 \right\}.$$

Define $\mathcal{B} : \mathcal{X} \times \Re^{m_E} \to \mathcal{X} \times \mathcal{X}$ to be the linear map whose adjoint $\mathcal{B}^*$ satisfies

$$\mathcal{B}^*(z, y_E) = \begin{pmatrix} z + \mathcal{A}_E^* y_E \\ -\mathcal{D}^* z \end{pmatrix}.$$

Then problem (4.7) can be reformulated as

$$(4.8) \quad \min \left\{ f(s, u) + g(y_I) + h(z, y_E) \mid \begin{pmatrix} s \\ \mathcal{D}^* u \end{pmatrix} + \begin{pmatrix} \mathcal{A}_I^* y_I \\ 0 \end{pmatrix} + \mathcal{B}^*(z, y_E) = \begin{pmatrix} c \\ 0 \end{pmatrix} \right\},$$

where

$$f(s, u) := \delta_{\mathcal{K}^*}(s) + \delta_{\mathcal{K}_p^*}(u), \quad g(y_I) := \delta_{\Re_+^{m_I}}(y_I) - \langle b_I, y_I \rangle, \quad h(z, y_E) := -\langle b_E, y_E \rangle$$

for any $(s, u) \in \mathcal{X} \times \mathcal{X}$, $y_I \in \Re^{m_I}$ and $(z, y_E) \in \mathcal{X} \times \Re^{m_E}$. Note that since

$$\mathcal{B}\mathcal{B}^* = \begin{pmatrix} \mathcal{I} + \mathcal{D}\mathcal{D}^* & \mathcal{A}_E^* \\ \mathcal{A}_E & \mathcal{A}_E \mathcal{A}_E^* \end{pmatrix}$$

and the inverse of $\mathcal{A}_E \mathcal{A}_E^*$ is assumed to be computable at a moderate cost, the inverse of $\mathcal{B}\mathcal{B}^*$ can also be computed based on $(\mathcal{I} + \mathcal{D}\mathcal{D}^*)^{-1}$ and the inverse of $\mathcal{A}_E(\mathcal{I} - (\mathcal{I} + \mathcal{D}\mathcal{D}^*)^{-1})\mathcal{A}_E^*$. For example, if $\mathcal{D}$ is a simple nonsingular matrix (e.g., $\mathcal{D} = \alpha \mathcal{I}$ for some $\alpha > 0$), then the inverse of $\mathcal{B}\mathcal{B}^*$ can be computed at a low cost once the inverse of $\mathcal{A}_E \mathcal{A}_E^*$ is available. Let $\rho_{\max}$ be the largest eigenvalue of the self-adjoint positive semidefinite operator $A_I \mathcal{A}_I^*$. Then we can apply our convergent Algorithm sPADMM3c given in section 3.1 directly to problem (4.8) by defining

$$\mathcal{T}_f \equiv 0 \quad \text{and} \quad \mathcal{T}_g \equiv \rho_{\max} \mathcal{I} - \mathcal{A}_I \mathcal{A}_I^*$$

to obtain a convergent 3-block sPADMM, denoted by Conic-sPADMM3c, for solving conic programming (P). The motivation for choosing the specific positive semidefinite linear operator $\mathcal{T}_g$ above is to make the computation of y_I^{k+1} simple. As before, the convergence analysis for Algorithm Conic-sPADMM3c can be analyzed similarly as in Theorem 4.2. For simplicity, we omit the details here.

5. Numerical experiments for SDP. In the last section, we have shown how our proposed convergent 3-block sPADMM for solving the convex optimization problem (3.1) can be used to solve (P) with/without the inequality constraints. In this section, we use (SDP) problems as our test examples. We separate our test examples into two groups. The first group is for DNN-SDP without the inequality constraints $\mathcal{A}_I x_I \geq b_I$. In the second group, we consider SDP problems arising from relaxation of binary integer quadratic (BIQ) programming problems with a large number of inequality constraints $\mathcal{A}_I x_I \geq b_I$.

5.1. Numerical results for DNN-SDP. The DNN-SDP takes the form of

$$(5.1) \qquad (\text{DNN-SDP}) \quad \max \left\{ \langle -C, X \rangle \mid \mathcal{A}_E X = b_E, \ X \in \mathcal{S}_+^n, \ X \in \mathcal{K}_p \right\},$$

whose dual can be written as

$$(5.2) \qquad \min \left\{ \delta_{\mathcal{S}_+^n}(S) + \delta_{\mathcal{K}_p^*}(Z) - \langle b_E, y_E \rangle \mid S + Z + \mathcal{A}_E^* y_E = C \right\}.$$

Obviously, our proposed Algorithm Conic-ADMM3c given in section 4.1 for solving problem (4.2) can be applied to problem (5.2) directly.

5.1.1. DNN-SDP problem sets. In our numerical experiments, we test the following classes of DNN-SDP problems:

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

$$(5.3) \qquad \min\left\{\frac{1}{2}x^T Q x + \langle c, x\rangle \mid x \in \{0,1\}^{n-1}\right\}.$$

It has been shown in [2] that under some mild assumptions, this problem can equivalently be reformulated as the following completely positive programming (CPP) problem:

$$(5.4) \quad \min\left\{\frac{1}{2}\langle Q, Y\rangle + \langle c, x\rangle \mid \operatorname{diag}(Y) = x, X = [Y, x; x^T, 1] \in \mathcal{C}_{pp}^n\right\},$$

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 problem, one would typically relax $\mathcal{C}_{pp}^n$ to $\mathcal{S}_+^n \cap \mathcal{K}_\mathcal{P}$, and the relaxed problem has the form of (SDP):

$$
\begin{aligned}
\min \quad & \tfrac{1}{2}\langle Q, Y\rangle + \langle c, x\rangle \\
(5.5)\\
\text{s.t.} \quad & \operatorname{diag}(Y) - x = 0, \quad \alpha = 1, \quad X = \begin{bmatrix} Y & x \\ x^T & \alpha \end{bmatrix} \in \mathcal{S}_+^n, \quad X \in \mathcal{K}_\mathcal{P},
\end{aligned}
$$

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

(ii) DNN-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

$$
\begin{aligned}
(5.6) \quad \theta_+(G) = \max\{&\langle ee^T, X\rangle \mid \langle E_{ij}, X\rangle = 0, (i,j) \in \mathcal{E}, \langle I, X\rangle = 1, \\
& X \in \mathcal{S}_+^n, X \in \mathcal{K}_\mathcal{P}\},
\end{aligned}
$$

where $E_{ij} = e_i e_j^\mathsf{T} + e_j e_i^\mathsf{T}$ and e_i denotes the ith column of the $n \times n$ identity matrix, and $\mathcal{K}_\mathcal{P} = \{X \in \mathcal{S}^n \mid X \geq 0\}$. In our numerical experiments, we test the graph instances G considered in [36], [38], and [39].

(iii) DNN-SDP problems arising from computing lower bounds for quadratic assignment problems (QAPs). Let Π be the set of $n \times n$ permutation matrices. Given matrices $A, B \in \mathcal{S}^n$, the QAP is given by

$$(5.7) \qquad \bar{v}_{\mathrm{QAP}} := \min\{\langle X, AXB\rangle : X \in \Pi\}.$$

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 $n^2 \times n^2$ matrix xx^T. It is shown in [30] that $\bar{v}_{\mathrm{QAP}}$ is bounded below by the following number generated from the SDP relaxation of (5.7):

$$
\begin{aligned}
v := \min \quad & \langle B \otimes A, Y\rangle \\
\text{s.t.} \quad & \sum_{i=1}^n Y^{ii} = I, \quad \langle I, Y^{ij}\rangle = \delta_{ij} \quad \forall 1 \leq i \leq j \leq n, \\
(5.8)\\
& \langle E, Y^{ij}\rangle = 1 \quad \forall 1 \leq i \leq j \leq n, \\
& Y \in \mathcal{S}_+^{n^2}, Y \in \mathcal{K}_\mathcal{P},
\end{aligned}
$$

where E is the matrix of ones, and $\delta_{ij} = 1$ if $i = j$, and 0 otherwise, $\mathcal{K}_{\mathcal{P}} = \{X \in \mathcal{S}^{n^2} \mid X \geq 0\}$. In our numerical experiments, the test instances (A, B) are taken from the QAP library [15].

(iv) DNN-SDP relaxation of clustering problems (RCPs) described in [29, eq. (13)]:

$$(5.9) \qquad \min\left\{\langle W, I - X\rangle \mid Xe = e, \langle I, X\rangle = K, X \in \mathcal{S}_+^n, X \in \mathcal{K}_{\mathcal{P}}\right\},$$

where W is the so-called affinity matrix whose entries represent the pairwise similarities of the objects in the data set, e is the vector of ones, and K is the number of clusters, $\mathcal{K}_{\mathcal{P}} = \{X \in \mathcal{S}^n \mid X \geq 0\}$. All the data sets we test are from the UCI Machine Learning Repository (available at http://archive.ics.uci.edu/ml/datasets.html). For some large-size data sets, we only select the first n rows. For example, the original data set `spambase` has 4601 rows and 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) DNN-SDP problems arising from semidefinite relaxation of frequency assignment problems (FAPs) [8]. Given a network represented by a graph G and an edge-weight matrix W, a certain type of FAP on G can be relaxed into the following SDP (see [3, eq. (5)]):

$$
(5.10) \qquad
\begin{aligned}
\max \quad & \langle (\tfrac{k-1}{2k})L(G, W) - \tfrac{1}{2}\mathrm{Diag}(We), X\rangle \\
\text{s.t.} \quad & \mathrm{diag}(X) = e, \quad X \in \mathcal{S}_+^n, \\
& \langle -E^{ij}, X\rangle = 2/(k-1) \quad \forall(i, j) \in U \subseteq E, \\
& \langle -E^{ij}, X\rangle \leq 2/(k-1) \quad \forall(i, j) \in E \setminus U,
\end{aligned}
$$

where $k > 1$ is an integer, $L(G, W); = \mathrm{Diag}(We) - W$ is the Laplacian matrix, $E^{ij} = e_i e_j^T + e_j e_i^T$ with $e_i \in \Re^n$ the vector with all zeros except in the ith position, and $e \in \Re^n$ is the vector of ones. Denote

$$
M_{ij} = \begin{cases}
-\frac{1}{k-1} & \forall(i, j) \in E, \\
0 & \text{otherwise.}
\end{cases}
$$

Then (5.10) is equivalent to

$$
(5.11) \qquad
\begin{aligned}
\max \quad & \langle (\tfrac{k-1}{2k})L(G, W) - \tfrac{1}{2}\mathrm{Diag}(We), X\rangle \\
\text{s.t.} \quad & \mathrm{diag}(X) = e, \ X \in \mathcal{S}_+^n, \ X - M \in \mathcal{K}_{\mathcal{P}},
\end{aligned}
$$

where $\mathcal{K}_{\mathcal{P}} = \{X \in \mathcal{S}^n \mid X_{ij} = 0, \forall(i, j) \in U; X_{ij} \geq 0, \forall(i, j) \in E \setminus U\}$. We should mention that we can easily extend our algorithm to handle the following slightly more general DNN-SDP:

$$\max\left\{\langle -C, X\rangle \mid \mathcal{A}_E X = b_E, \ X \in \mathcal{S}_+^n, \ X - M \in \mathcal{K}_p\right\},$$

where $M \in \mathcal{S}^n$ is a given matrix. Thus (5.11) can also be solved by our proposed algorithm.

5.1.2. Numerical results. For large-scale DNN-SDP problems, there exist two other competitive codes in the literature that are based on alternating direction algorithms: a directly extended ADMM solver (called SDPAD in [43]) and a two-easy-block-decomposition hybrid proximal extragradient method solver (called 2EBD-HPE in [25], but we will just call it 2EBD for convenience).

Here we compare our algorithm ADMM3c with SDPAD (release-beta2, released in December 2012) and 2EBD[5] (v0.2, released on May 31, 2013) for solving DNN-SDP. We also include a convergent alternating direction method with a Gaussian back substitution proposed in [18] (we call the method ADMM3g here and use the parameter $\alpha = 0.999$ in the Gaussian back substitution step[6]). We have implemented both ADMM3c and ADMM3g in MATLAB. The computational results for all the DNN-SDP problems are obtained on a Linux server (6-core, Intel Xeon X5650 @ 2.67 GHz, 32 G RAM).

We measure the accuracy of an approximate optimal solution (X, y_E, S, Z) for (5.1) and (5.2) by using the following relative residual:

$$(5.12) \qquad \eta = \max\{\eta_P, \eta_D, \eta_\mathcal{K}, \eta_\mathcal{P}, \eta_{\mathcal{K}^*}, \eta_{\mathcal{P}^*}, \eta_{C_1}, \eta_{C_2}\},$$

where

$$(5.13)$$

$$\eta_P = \frac{\|\mathcal{A}_E X - b_E\|}{1 + \|b_E\|}, \quad \eta_D = \frac{\|\mathcal{A}_E^* y_E + S + Z - C\|}{1 + \|C\|}, \quad \eta_\mathcal{K} = \frac{\|\Pi_{\mathcal{S}_+^n}(-X)\|}{1 + \|X\|}, \quad \eta_\mathcal{P} = \frac{\|X - \Pi_{\mathcal{K}_\mathcal{P}}(X)\|}{1 + \|X\|},$$

$$\eta_{\mathcal{K}^*} = \frac{\|\Pi_{\mathcal{S}_+^n}(-S)\|}{1 + \|S\|}, \quad \eta_{\mathcal{P}^*} = \frac{\|Z - \Pi_{\mathcal{K}_\mathcal{P}^*}(Z)\|}{1 + \|Z\|}, \quad \eta_{C_1} = \frac{|\langle X, S \rangle|}{1 + \|X\| + \|S\|}, \quad \eta_{C_2} = \frac{|\langle X, Z \rangle|}{1 + \|X\| + \|Z\|}.$$

Additionally, we compute the relative gap by

$$(5.14) \qquad \eta_g = \frac{\langle C, X \rangle - \langle b_E, y_E \rangle}{1 + |\langle C, X \rangle| + |\langle b_E, y_E \rangle|}.$$

We terminate the solvers ADMM3c, ADMM3g, and SDPAD when $\eta < 10^{-6}$. Note that, as mentioned in the introduction, the direct extension of ADMM to the case of a multiblock problem is not necessarily convergent [4]. Hence SDPAD, which is essentially an implementation of ADMM3d with $\tau = 1.618$ for solving DNN-SDP problems, does not have convergence guarantee. For the implementation of 2EBD, we need to explain in more details.

The method 2EBD in [25] is designed to solve a conic optimization problem of the form

$$(5.15) \qquad \min\Big\{\langle C, X \rangle \mid \mathcal{A}_1(X) - b_1 \in \mathcal{C}_1, \ \mathcal{A}_2(X) - b_2 \in \mathcal{C}_2\Big\},$$

where $b_1 \in \mathcal{W}_1$, $b_2 \in \mathcal{W}_2$, $C \in \mathcal{X}$ are given data, $\mathcal{A}_1 : \mathcal{X} \to \mathcal{W}_1$, $\mathcal{A}_2 : \mathcal{X} \to \mathcal{W}_2$ are given linear maps, and $\mathcal{C}_1 \in \mathcal{W}_1$, $\mathcal{C}_2 \in \mathcal{W}_2$ are nonempty closed convex cones. The dual of (5.15) is given by

$$(5.16) \quad \max\Big\{\langle b_1, w_1 \rangle + \langle b_2, w_2 \rangle \mid \mathcal{A}_1^*(w_1) + \mathcal{A}_2^*(w_2) = C, \ w_1 \in \mathcal{C}_1^*, \ w_2 \in \mathcal{C}_2^*\Big\}.$$

Note that the application of 2EBD to DNN-SDP strongly depends on the possibility of splitting its constraints into two-easy blocks such that the projection problems

---

[5]www2.isye.gatech.edu/~cod3/CamiloOrtiz/Software_files/2EBD-HPE_v0.2/2EBD-HPE_v0.2.zip

[6]We avoid taking $\alpha = 1$ as it leads to slow convergence for quite a number of tested examples.

$\min\left\{\frac{1}{2}\|X-X_0\|^2 \mid \mathcal{A}_i(X)-b_i \in \mathcal{C}_i\right\}, i = 1, 2$, can be computed easily for any given X_0. The users need to input the algorithms for computing $\min\left\{\frac{1}{2}\|X - X_0\|^2 \mid \mathcal{A}_i(X) - b_i \in \mathcal{C}_i\right\}, i = 1, 2$. For BIQ, θ_+, and FAP problems, their constraints can naturally be split into two-easy blocks [25]. For general DNN-SDP problems; however, it may be difficult to split their constraints naturally into two-easy blocks, although one can always reformulate them as SDP problems in the standard form. In our numerical experiments, we use this approach to test QAP and RCP problems as these problems do not appear to have obvious two-easy blocks structures.

For testing 2EDB on DNN-SDP problems, in order to agree with our stopping criterion, given a solution (X, w_1, w_2, S, Z) for (5.15) and (5.16), where S, Z are the dual variables corresponding to $X \in \mathcal{S}_+^n$ and $X \in \mathcal{K}_\mathcal{P}$, respectively, we measure the relative residual as follows:

$$(5.17) \qquad \hat{\eta} = \max\{\eta_P, \eta_D, \eta_\mathcal{K}, \eta_\mathcal{P}, \eta_{\mathcal{K}^*}, \eta_{\mathcal{P}^*}, \eta_{C_1}, \eta_{C_2}\},$$

where $\eta_P = \frac{\|(\Pi_{\mathcal{C}_1^*}(b_1 - \mathcal{A}_1 X), \Pi_{\mathcal{C}_2^*}(b_2 - \mathcal{A}_2 X))\|}{1 + \|(b_1, b_2)\|}$, $\eta_D = \frac{\|\mathcal{A}_1^*(w_1) + \mathcal{A}_2^*(w_2) - C\|}{1 + \|C\|}$, and $\eta_\mathcal{K}, \eta_\mathcal{P}, \eta_{\mathcal{K}^*}$, $\eta_{\mathcal{P}^*}, \eta_{C_1}, \eta_{C_2}$ are defined as in (5.13). We terminate the solver 2EBD when $\hat{\eta} < 10^{-6}$. And we measure the relative gap as

$$(5.18) \qquad \hat{\eta}_g = \frac{\langle C, X \rangle - (\langle b_1, w_1 \rangle + \langle b_2, w_2 \rangle)}{1 + |\langle C, X \rangle| + |\langle b_1, w_1 \rangle + \langle b_2, w_2 \rangle|}.$$

We should mention in the implementations of all the solvers, ADMM3c, SDPAD, ADMM3g, and 2EDB, the penalty parameter σ is dynamically adjusted according to the progress of the algorithms. In addition, all the algorithms also adopt some kind of restart strategies to ameliorate slow convergence. The exact details on the adjustment strategies are too tedious to be presented here but it suffices to mention that the key idea is to adjust σ so as to balance the progress of primal feasibilities $(\eta_P, \eta_\mathcal{K}, \eta_\mathcal{P})$ and dual feasibilities $(\eta_D, \eta_{\mathcal{K}^*}, \eta_{\mathcal{P}^*})$. In our numerical experiments, we use the same adjustment strategy for both the solvers ADMM3c and ADMM3g to solve all the tested problems, i.e., we do not change the strategy to maximize the performance of different classes of tested problems. The solver SDPAD also uses a common adjustment strategy, though different from that of ADMM3c and ADMM3g. But 2EDB uses different parameter settings for the adjustment strategy for different classes of tested problems.

Table 1 shows the number of problems that have been successfully solved to the accuracy of 10^{-6} in η or $\hat{\eta}$ by each of the four solvers ADMM3c, SDPAD, ADMM3g, and 2EBD, with the maximum number of iterations set at 25,000. We can see that ADMM3c solved the most instances to the required accuracy, with SDPAD in the second place, followed by ADMM3g and 2EDB in the third and fourth places, respectively. It is actually quite remarkable that all four solvers are able to solve these large-scale SDP problems to such a good accuracy despite the fact that they are all first order methods.

Table 2 reports detailed numerical results for ADMM3c, SDPAD, 2EBD, and ADMM3g in solving some very-large-scale DNN-SDP problems. The detailed results for all the 414 problems tested can be found at http://www.math.nus.edu.sg/~mattohkc/publist.html. Note that we did not list the numerical results for the directly extended ADMM with $\tau = 1$ here as it almost always takes 20% to 50% more time than the one with $\tau = 1.618$, i.e., SDPAD. From the detailed numerical results, one can observe that ADMM3c is generally the fastest in terms of the computing time, especially when the problem size is large.

TABLE 1
Numbers of problems which are solved to the accuracy of 10^{-6} in η or $\hat{\eta}$.

Problem set (No.)\solver	ADMM3c	SDPAD	2EBD	ADMM3g
θ_+ (58)	58	58	56	54
FAP (7)	7	7	7	7
QAP (95)	39	30	16	28
BIQ (134)	134	134	134	130
RCP (120)	120	114	109	113
Total (414)	358	343	322	332
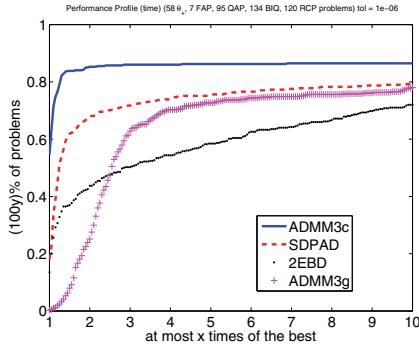
FIG. 1. *Performance profiles (time) of* ADMM3c, SDPAD, ADMM3g *and* 2EBD.
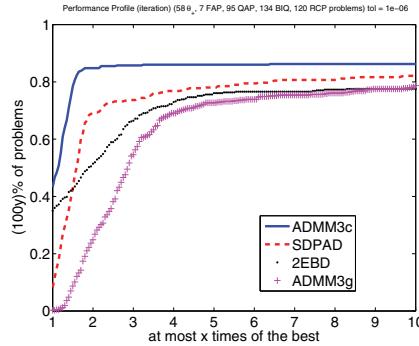
FIG. 2. *Performance profiles (iteration) of* ADMM3c, SDPAD, ADMM3g *and* 2EBD.

Figure 1 shows the performance profiles in terms of computing time for ADMM3c, SDPAD, 2EBD, and ADMM3g for all the tested problems including those problems not listed in Table 2. We recall that a point (x, y) is in the performance profiles curve of a method if and only if it can solve $(100y)\%$ of all the tested problems no slower than x times of any other methods. It can be seen that ADMM3c outperforms the three other solvers by a significant margin.

Figure 2 shows the performance profiles in terms of the number of iterations for ADMM3c, SDPAD, 2EBD, and ADMM3g for all the tested problems. We may observe that for the majority of the tested problems, ADMM3c takes the least number of iterations. For the BIQ problems, the solver 2EDB typically takes the least number of iterations. However, as each iteration of 2EDB requires quite a number of intermediate calculations to estimate a step-length to achieve good convergence, the nontrivial overheads incurred often counteract the savings in the number of iterations. As a result, even though the performance profile of 2EDB in terms of the number of iterations dominates that of ADMM3g, its profile in terms of the computing time does not behave similarly.

5.2. Numerical results for SDP with many inequality constraints. In this subsection, we will consider (SDP) with many inequality constraints $\mathcal{A}_I x_I \geq b_I$. The dual of (SDP) takes the form of

$$(5.19) \quad \min\left\{\delta_{\mathcal{S}^n_+}(S) + (\delta_{\Re^{m_I}_+}(y_I) - \langle b_I, y_I\rangle) + \delta_{\mathcal{K}^*_p}(Z) - \langle b_E, y_E\rangle \mid \right.$$
$$\left. \times\ S + \mathcal{A}^*_I y_I + Z + \mathcal{A}^*_E y_E = C\right\}.$$

TABLE 2

The performance of ADMM3c, SDPAD, ADMM3g, 2EBD on θ_+, FAP, QAP, BIQ, and RCP problems (accuracy $= 10^{-6}$). In the table, 3c and 3g stand for ADMM3c and ADMM3g, respectively. The computation time is in the format hours:minutes:seconds.

problem	$m_E; m_I$	n_s;	Iteration 3c	SDPAD	3g	2EBD	η 3c	SDPAD	3g	$\hat{\eta}$ 2EBD	η_g 3c	SDPAD	3g	2EBD	Time 3c	SDPAD	3g	2EBD
theta10	12470;0	500;	354	351	636	490	8.5-7	9.9-7	9.9-7	9.9-7	-2.5-6	-7.8-7	7.4-7	1.5-6	45	42	1:34	1:11
theta102	37467;0	500;	157	130	232	355	9.5-7	9.0-7	9.7-7	9.9-7	-6.0-7	2.1-6	9.0-7	2.2-6	23	23	36	54
theta103	62516;0	500;	144	108	199	323	9.2-7	9.8-7	9.9-7	9.8-7	-3.0-8	5.9-7	1.8-7	2.6-6	22	21	32	49
theta104	87245;0	500;	169	123	226	338	9.3-7	9.8-7	9.0-7	9.9-7	-9.2-8	1.1-6	4.2-7	3.2-6	24	20	36	51
theta12	17979;0	600;	362	366	648	494	9.0-7	8.8-7	9.6-7	9.2-7	-2.2-6	1.0-6	1.2-6	1.3-6	1:14	1:11	2:28	1:52
theta123	90020;0	600;	156	107	197	345	9.3-7	9.9-7	9.5-7	9.9-7	-6.0-8	5.1-7	1.4-7	2.6-6	35	33	49	1:26
c-fat200-1	18367;0	200;	233	444	472	330	9.8-7	9.9-7	9.5-7	9.9-7	-6.9-7	-1.2-6	-1.1-6	2.1-6	03	06	06	04
hamming-8-4	11777;0	256;	124	104	179	214	4.7-7	9.6-7	8.5-7	8.9-7	-5.3-6	2.1-6	-1.4-6	1.0-5	02	03	04	04
hamming-10-2	23041;0	1024;	657	651	871	902	8.7-7	9.4-7	9.8-7	8.8-7	7.6-6	-2.6-6	7.9-7	3.4-5	3:05	5:17	4:43	3:47
hamming-8-3-4	16129;0	256;	232	189	297	180	7.8-7	5.5-7	9.5-7	9.5-7	2.0-7	9.9-7	1.6-6	-3.5-6	06	04	07	03
hamming-9-5-6	53761;0	512;	461	507	691	563	9.5-7	9.5-7	9.6-7	8.9-7	-1.2-5	-1.9-6	1.3-6	8.0-6	45	54	1:20	58
brock400-1	20078;0	400;	171	155	268	354	8.9-7	9.9-7	9.9-7	9.4-7	-1.6-6	1.6-6	-1.2-6	1.7-6	14	14	25	31
p-hat300-1	33918;0	300;	649	791	1901	759	9.9-7	9.9-7	8.7-7	9.9-7	-1.3-7	1.3-6	7.5-7	1.8-6	26	35	1:28	33
G43	9991;0	1000;	1151	1147	2145	934	9.8-7	9.4-7	9.7-7	9.9-7	-3.1-6	1.7-6	1.5-6	2.0-6	13:04	10:20	21:38	13:00
G44	9991;0	1000;	1151	1144	2141	968	9.3-7	9.9-7	9.9-7	9.9-7	-2.9-6	1.6-6	1.5-6	1.6-6	12:13	10:11	21:08	13:15
G45	9991;0	1000;	1175	1185	2181	966	9.5-7	9.4-7	9.8-7	9.9-7	2.9-6	-1.0-6	-1.1-6	1.6-6	13:24	10:36	21:22	13:28
G46	9991;0	1000;	1199	1180	2159	943	9.9-7	9.8-7	9.5-7	9.9-7	-3.2-6	-1.0-6	7.7-7	1.4-6	12:55	10:42	21:47	12:58
G47	9991;0	1000;	1186	1137	2154	992	9.5-7	9.5-7	9.9-7	9.9-7	2.9-6	-9.4-7	8.3-7	1.2-6	13:18	10:28	21:00	13:50
G51	5910;0	1000;	6207	10361	25000	9586	9.5-7	9.9-7	2.8-6	9.9-7	3.7-7	2.6-7	6.9-7	5.6-7	1:21:52	2:11:03	6:11:20	2:31:30
G52	5917;0	1000;	11463	14163	25000	12124	9.9-7	9.9-7	3.0-6	9.9-7	4.2-7	4.5-7	9.9-7	6.9-7	2:26:28	2:46:25	6:00:51	3:15:11
G53	5915;0	1000;	13289	23865	25000	20623	9.9-7	9.9-7	2.7-6	9.9-7	2.6-6	2.9-6	4.4-6	4.2-6	2:49:53	4:48:56	6:04:31	5:49:06
G54	5917;0	1000;	3262	7542	6253	5136	9.7-7	9.9-7	9.9-7	9.9-7	3.1-6	4.6-7	-1.7-6	1.3-6	38:42	1:26:47	1:28:08	1:17:01
1dc.512	9728;0	512;	2216	2269	2675	2634	9.9-7	9.9-7	9.9-7	9.9-7	4.1-7	2.2-6	6.4-7	3.3-6	5:03	7:01	6:28	6:14
1et.512	4033;0	512;	990	1470	3101	1530	9.9-7	9.9-7	9.7-7	9.9-7	-1.1-7	3.9-6	-7.6-9	5.6-6	1:58	3:15	7:40	3:08
1tc.512	3265;0	512;	2494	3340	3501	3807	9.9-7	9.9-7	8.8-7	9.9-7	9.4-7	2.5-6	6.6-7	3.3-6	5:04	10:15	7:59	9:03
2dc.512	54896;0	512;	2956	2701	5602	2173	9.9-7	9.9-7	9.9-7	9.9-7	8.5-6	7.5-6	1.4-5	1.6-5	5:34	6:36	13:05	4:45
1zc.512	6913;0	512;	490	1056	728	2120	8.5-7	9.9-7	9.8-7	9.9-7	4.7-6	2.2-7	1.2-6	3.0-7	54	3:08	1:28	4:16
1dc.1024	24064;0	1024;	2620	2681	3301	3641	9.9-7	9.9-7	9.2-7	9.9-7	1.3-6	3.4-6	2.3-6	4.0-6	32:22	45:21	46:28	53:12
1et.1024	9601;0	1024;	1144	2563	2263	2609	9.9-7	9.9-7	9.9-7	9.9-7	1.3-6	5.6-6	-2.3-9	5.9-6	12:54	39:53	30:09	35:35
1tc.1024	7937;0	1024;	2732	6545	25000	6675	9.9-7	9.9-7	1.8-6	9.9-7	4.5-6	4.5-6	6.8-6	4.2-6	32:08	1:48:31	6:05:36	1:40:06
1zc.1024	16641;0	1024;	711	770	1101	25000	7.7-7	9.9-7	9.9-7	3.1-5	5.4-6	2.0-6	1.2-6	7.9-4	7:19	12:18	13:24	7:48:20
2dc.1024	169163;0	1024;	4135	1896	6901	1891	9.9-7	9.9-7	9.7-7	9.9-7	1.3-5	1.0-5	2.3-5	1.5-5	45:59	29:02	1:34:09	24:59
1dc.2048	58368;0	2048;	4153	7277	5255	8476	9.9-7	9.9-7	9.9-7	9.9-7	4.2-6	6.4-6	-2.4-6	6.5-6	5:47:45	13:59:49	8:02:43	16:04:13
1et.2048	22529;0	2048;	3039	4422	4101	4739	9.9-7	9.9-7	8.8-7	9.9-7	1.1-6	4.8-6	4.2-7	7.8-6	4:04:34	8:47:18	6:33:13	8:28:46
1tc.2048	18945;0	2048;	2876	7329	8991	7482	9.9-7	9.9-7	9.9-7	9.9-7	1.5-6	5.5-6	1.7-6	5.6-6	3:50:16	13:29:15	14:15:48	13:50:32
2dc.2048	504452;0	2048;	2997	2147	4048	1849	9.9-7	9.9-7	9.9-7	9.9-7	8.3-6	1.0-5	2.0-5	2.2-5	3:52:42	4:13:47	6:04:58	3:07:46

TABLE 2
(cont.).

fap10	183;0	183;	1424	2313	5145	2774	6.0-7	9.9-7	9.9-7	3.7-6	-1.3-4	-4.1-5	-6.8-5	25	27	1:31	42	
fap11	252;0	252;	1559	2585	4355	2771	5.3-7	9.9-7	9.7-7	-1.9-5	-2.2-4	-5.5-5	-1.1-4	50	1:07	2:25	1:18	
fap12	369;0	369;	1830	3394	7742	3325	8.4-7	9.9-7	9.9-7	-2.6-5	-2.2-4	-5.0-5	-1.3-4	1:55	3:32	9:19	3:08	
fap25	2118;0	2118;	5799	5495	11189	4498	9.9-7	9.9-7	9.9-7	-3.2-5	-1.1-4	-1.6-5	-7.1-5	10:55:33	13:26:47	20:20:27	8:11:50	
fap36	4110;0	4110;	2824	4445	7424	3500	9.9-7	9.8-7	9.8-7	-1.7-5	-3.0-5	-4.0-6	-2.8-5	30:57:53	78:43:03	89:33:09	43:37:44	
bur26a	1051;0	676;	25000	25000	25000	25000	5.6-6	1.1-5	2.7-5	-6.3-5	-7.7-5	-9.1-5	-8.2-5	2:05:11	2:07:44	2:02:50	2:38:24	
bur26b	1051;0	676;	25000	25000	25000	25000	6.8-6	1.1-5	2.3-5	-5.7-5	-8.0-5	-9.4-5	-7.5-5	2:07:13	1:57:30	1:58:25	2:49:59	
bur26c	1051;0	676;	25000	25000	25000	25000	4.2-6	1.4-5	3.2-5	-4.5-5	-1.2-4	-1.2-4	-1.8-4	2:05:11	2:02:35	2:06:14	2:50:08	
bur26d	1051;0	676;	25000	25000	25000	25000	6.4-6	1.5-5	3.2-5	-8.4-5	-1.2-4	-1.2-4	-1.4-4	2:02:24	1:51:20	2:05:16	2:53:07	
bur26e	1051;0	676;	25000	25000	25000	25000	3.1-6	6.4-6	4.7-6	-2.8-5	-3.6-5	-3.6-5	-1.9-4	2:03:18	2:28:06	2:03:16	2:46:03	
bur26f	1051;0	676;	20887	25000	25000	25000	9.9-7	8.1-6	7.0-6	-1.0-5	-4.8-5	-4.3-5	-7.5-5	1:45:08	2:09:11	2:02:20	2:44:28	
bur26g	1051;0	676;	17910	25000	25000	25000	8.6-7	1.6-6	2.4-6	-6.3-6	-4.0-5	-3.1-5	-6.9-5	1:29:22	1:57:13	2:00:33	2:46:34	
bur26h	1051;0	676;	23208	25000	25000	25000	9.4-7	1.4-6	2.3-6	-1.4-6	-2.3-5	-2.8-5	-1.7-4	1:57:33	2:01:12	1:58:07	2:54:20	
chr20a	628;0	400;	25000	25000	25000	25000	2.0-6	2.2-6	3.5-6	-2.4-4	-8.9-5	-1.0-4	-4.3-3	29:49	43:13	34:51	49:22	
chr20b	628;0	400;	8256	25000	17893	25000	9.6-7	2.9-5	8.8-7	5.2-4	-5.4-3	3.6-4	-4.0-3	10:39	45:06	23:16	51:14	
chr20c	628;0	400;	14673	25000	25000	25000	9.1-7	1.4-6	1.3-6	5.2-6	-2.7-4	-2.3-4	-1.1-2	13:45	26:08	25:54	48:58	
chr22a	757;0	484;	6457	22364	13547	25000	8.8-7	9.9-7	9.8-7	3.9-4	-2.7-4	2.5-4	-2.6-3	12:29	50:45	27:27	1:17:44	
chr22b	757;0	484;	7211	25000	14587	25000	9.7-7	1.5-5	9.8-7	3.4-4	-1.4-3	2.4-4	-1.8-3	13:52	1:07:51	30:26	1:15:34	
chr25a	973;0	625;	7127	25000	16196	25000	8.6-7	3.7-5	9.8-7	7.8-4	-9.5-3	5.6-4	-6.8-3	26:03	2:10:29	1:03:08	2:23:01	
esc32a	1582;0	1024;	4931	2247	5188	9630	9.9-7	9.9-7	9.9-7	-2.0-6	-1.1-6	-3.9-5	-2.7-6	1:06:25	27:49	1:00:38	2:44:37	
esc32b	1582;0	1024;	25000	25000	25000	25000	3.7-6	2.4-6	2.6-5	-2.4-4	-2.2-4	-6.5-4	-4.0-4	5:18:19	4:31:51	6:12:17	7:59:47	
esc32c	1582;0	1024;	25000	25000	25000	25000	5.3-6	5.1-6	8.2-6	-5.4-5	-5.2-5	-6.9-5	-9.8-5	4:58:47	4:00:23	6:01:30	8:01:19	
esc32d	1582;0	1024;	678	799	1679	1412	9.9-7	9.9-7	9.9-7	-9.4-6	-1.7-5	-6.8-7	-2.2-7	9:39	8:01	21:45	25:40	
esc32e	1582;0	1024;	1108	905	6873	784	9.8-7	8.6-7	9.9-7	-3.8-7	-9.2-6	-3.7-4	-3.1-6	16:09	8:32	1:23:01	14:30	
esc32f	1582;0	1024;	1108	905	6873	784	9.8-7	8.6-7	9.9-7	-3.8-7	-9.2-6	-3.7-4	-3.1-6	15:45	8:25	1:19:37	12:57	
esc32g	1582;0	1024;	520	588	2328	981	9.3-7	9.2-7	9.9-7	1.9-6	-3.3-6	-1.2-4	3.6-7	7:14	5:41	33:24	17:19	
esc32h	1582;0	1024;	25000	25000	25000	25000	9.4-6	1.2-5	6.1-5	-4.1-4	-5.0-4	-7.4-4	-7.4-4	5:01:41	4:13:31	5:57:23	7:30:56	
kra30a	1393;0	900;	25000	25000	25000	25000	1.3-5	1.4-5	7.5-5	-4.2-4	-5.7-4	-7.7-4	-5.9-5	3:45:29	5:11:22	4:22:39	5:42:45	
kra30b	1393;0	900;	25000	25000	25000	25000	1.1-5	1.1-5	6.8-5	-3.7-4	-4.9-4	-9.4-4	-6.1-4	3:56:14	5:15:18	4:18:02	5:45:45	
kra32	1582;0	1024;	25000	25000	25000	25000	9.1-6	1.1-5	5.4-5	-3.2-4	-4.0-4	-7.0-4	-5.0-4	5:07:43	6:57:49	5:44:04	8:11:14	
lipa20a	628;0	400;	1653	5698	4766	25000	8.8-7	9.8-7	4.8-5	-4.4-6	-2.0-5	2.2-5	-1.6-3	1:43	6:54	4:58	51:25	
lipa20b	628;0	400;	1514	4747	4444	25000	8.5-7	9.7-7	9.8-7	9.1-6	-3.1-5	-3.2-5	-2.0-2	1:19	3:51	3:59	48:31	
lipa30a	1393;0	900;	3533	11683	25000	16167	8.9-7	9.5-7	7.2-6	8.3-6	3.2-5	-2.4-4	-3.8-6	26:59	1:52:11	2:55:04	3:46:31	
lipa30b	1393;0	900;	2700	7516	25000	25000	9.1-7	9.9-7	1.4-4	4.3-5	4.7-5	-6.6-3	1.1-2	16:33	52:22	2:48:26	5:31:56	
lipa40a	2458;0	1600;	6483	18785	25000	25000	8.8-7	9.8-7	2.3-5	4.2-5	-4.1-5	-1.0-3	-1.5-4	3:32:47	19:15:19	14:17:50	26:56:27	
lipa40b	2458;0	1600;	4878	5970	25000	25000	9.0-7	9.8-7	3.6-4	1.1-4	-6.4-5	-2.3-2	-4.1-2	2:20:09	4:11:06	12:45:41	23:33:11	
nug20	628;0	400;	25000	25000	25000	25000	8.7-6	9.6-6	3.9-5	-1.7-4	-2.1-4	-4.0-4	-2.5-4	30:28	40:29	35:55	48:41	
nug21	691;0	441;	25000	25000	25000	25000	1.0-5	1.3-5	4.0-5	-2.2-4	-2.8-4	-4.7-4	-3.3-4	38:17	51:44	44:31	1:04:08	
nug22	757;0	484;	25000	25000	25000	25000	1.3-5	1.6-5	4.1-5	-2.7-4	-3.6-4	-5.8-4	-3.9-4	49:55	1:02:41	56:47	1:14:17	
nug24	898;0	576;	25000	25000	25000	25000	9.1-6	1.1-5	3.8-5	-1.9-4	-2.3-4	-4.1-4	-2.7-4	1:17:49	1:40:05	1:23:47	1:57:03	

TABLE 2
(cont.).

Problem	p1	p2	it1	it2	it3	it4	r1	r2	r3	r4	η1	η2	η3	η4	t1	t2	t3	t4
nug25	973;0	625;	25000	25000	25000	25000	-2.0-4	-2.0-4	-3.5-4	-2.5-4	1.2-5	1.0-5	3.6-5	1.7-5	1:35:46	1:53:26	1:45:18	2:16:27
nug27	1132;0	729;	25000	25000	25000	25000	-2.0-4	-2.6-4	-4.3-4	-2.8-4	1.0-5	1.3-5	3.8-5	1.7-5	2:21:06	2:51:26	2:36:56	3:28:20
nug28	1216;0	784;	25000	25000	25000	25000	-1.8-4	-2.2-4	-3.8-4	-2.6-4	9.3-6	1.2-5	3.4-5	1.7-5	2:47:04	3:27:54	3:02:26	4:02:11
nug30	1393;0	900;	25000	25000	25000	25000	-1.6-4	-1.9-4	-3.3-4	-2.2-4	8.7-6	1.1-5	3.3-5	1.7-5	3:48:43	4:58:12	4:23:21	5:39:31
ste36a	1996;0	1296;	25000	25000	25000	25000	-5.8-4	-6.8-4	-9.5-4	-6.7-4	9.7-6	1.3-5	3.7-5	1.6-5	9:38:26	12:37:18	11:11:59	14:09:11
ste36b	1996;0	1296;	25000	25000	25000	25000	-1.5-3	-2.0-3	-2.3-3	-2.1-3	1.2-5	1.8-5	4.4-5	1.3-5	9:19:24	12:10:09	10:45:58	14:23:33
ste36c	1996;0	1296;	25000	25000	25000	25000	-5.8-4	-7.3-4	-9.4-4	-7.2-4	1.2-5	1.5-5	4.3-5	1.6-5	9:26:42	12:22:19	11:01:56	14:23:52
tai25a	973;0	625;	2201	1845	2477	25000	-8.0-4	-7.2-4	-8.5-4	-1.8-3	9.5-7	9.9-7	9.9-7	1.7-6	8:38	9:24	11:18	2:27:04
tai25b	973;0	625;	25000	25000	25000	25000	-2.0-3	-2.4-3	-3.2-3	-2.5-3	2.9-5	3.7-5	6.3-5	4.2-5	1:28:33	1:55:04	1:43:51	2:21:35
tai30a	1393;0	900;	25000	25000	25000	25000	-6.3-5	-7.3-5	-1.8-4	-1.3-4	4.7-6	4.6-6	3.2-5	1.3-5	3:53:48	6:09:25	4:31:53	6:00:13
tai30b	1393;0	900;	25000	25000	25000	25000	-1.0-3	-1.2-3	-1.7-3	-1.2-3	2.0-5	2.4-5	4.4-5	2.6-5	3:42:12	4:28:02	4:17:53	5:38:24
tai35a	1888;0	1225;	25000	25000	25000	25000	-4.8-5	-5.6-5	-1.4-4	-1.0-4	3.9-6	4.0-6	2.8-5	1.3-5	9:21:21	15:00:46	10:43:14	12:53:01
tai35b	1888;0	1225;	25000	25000	25000	25000	-9.1-4	-1.0-3	-1.5-3	-1.1-3	2.1-5	2.4-5	4.4-5	2.8-5	8:51:20	11:15:52	10:28:44	12:51:27
tai40a	2458;0	1600;	25000	25000	25000	25000	-4.6-5	-5.3-5	-1.4-4	-1.0-4	3.7-6	4.0-6	2.7-5	1.4-5	20:22:53	31:45:29	23:23:44	26:00:47
tai40b	2458;0	1600;	25000	25000	25000	25000	-7.2-4	-8.1-4	-1.1-3	-8.5-4	1.9-5	2.5-5	4.6-5	3.1-5	17:50:19	23:17:25	19:57:37	25:23:31
tho30	1393;0	900;	25000	25000	25000	25000	-2.6-4	-3.4-4	-5.6-4	-4.0-4	1.1-5	1.5-5	4.0-5	2.2-5	3:46:49	4:46:03	4:23:12	5:44:33
tho40	2458;0	1600;	25000	25000	25000	25000	-2.1-4	-2.7-4	-4.5-4	-3.2-4	9.3-6	1.3-5	3.7-5	2.0-5	17:12:50	24:35:42	19:56:25	26:05:11
be250.1	251;0	251;	4327	5345	8441	3537	-2.0-7	-3.6-7	-4.3-6	-4.1-7	9.9-7	9.9-7	9.9-7	9.9-7	1:16	1:35	2:49	1:37
be250.2	251;0	251;	3827	5108	7900	3044	-8.6-7	-5.3-7	-1.8-6	-8.0-7	9.9-7	9.9-7	9.9-7	9.9-7	1:08	1:28	2:36	1:22
be250.3	251;0	251;	3796	4331	7816	2592	-1.1-6	-7.3-7	-1.8-6	-1.1-6	9.9-7	9.9-7	9.9-7	9.9-7	1:11	1:18	2:31	1:11
be250.4	251;0	251;	8023	8350	16334	6453	-1.1-6	-1.1-6	-2.0-7	-2.9-7	9.9-7	9.9-7	9.9-7	9.9-7	2:23	2:24	5:25	2:53
be250.5	251;0	251;	4460	5089	7757	3174	-7.5-7	-6.4-7	-4.6-7	-7.2-7	9.9-7	9.9-7	9.9-7	9.9-7	1:23	1:31	2:33	1:26
be250.6	251;0	251;	4095	4560	8149	2812	-5.7-7	-5.4-7	-6.4-7	-4.3-7	9.9-7	9.9-7	9.9-7	9.9-7	1:13	1:17	2:37	1:16
be250.7	251;0	251;	4345	5048	9076	3295	-1.1-7	-2.4-7	6.8-7	-2.4-7	9.9-7	9.9-7	9.9-7	9.9-7	1:20	1:28	2:57	1:31
be250.8	251;0	251;	3759	4663	7959	2911	1.9-7	-8.8-7	-3.9-6	2.5-7	9.9-7	9.9-7	9.9-7	9.9-7	1:08	1:18	2:32	1:18
be250.9	251;0	251;	4624	5976	10301	4169	-1.1-6	-1.1-6	-7.7-7	-4.8-7	9.9-7	9.9-7	9.9-7	9.9-7	1:26	1:49	3:28	1:54
be250.10	251;0	251;	5963	6638	14801	3989	-7.6-7	-7.4-7	2.8-8	-4.2-7	9.9-7	9.9-7	9.9-7	9.9-7	1:46	1:55	4:46	1:49
bqp250-1	251;0	251;	4593	4946	8603	3216	-8.8-7	1.0-6	2.6-6	-8.1-7	9.9-7	9.9-7	9.9-7	9.9-7	1:19	1:26	2:50	1:28
bqp250-2	251;0	251;	4388	5097	8650	3293	-1.1-6	-7.4-7	-1.1-6	-6.2-7	9.9-7	9.9-7	9.9-7	9.9-7	1:26	1:30	2:54	1:31
bqp250-3	251;0	251;	4039	5332	10230	3203	-2.1-6	-3.1-7	-1.5-6	-7.5-7	9.9-7	9.9-7	9.9-7	9.9-7	1:08	1:29	3:12	1:28
bqp250-4	251;0	251;	3662	4539	7392	2548	-1.4-6	-6.7-7	-3.4-6	1.5-7	9.9-7	9.9-7	9.9-7	9.9-7	1:05	1:20	2:26	1:09
bqp250-5	251;0	251;	4558	8062	16221	4487	-1.1-6	-4.8-7	-3.8-7	-5.6-7	9.9-7	9.9-7	9.9-7	9.9-7	1:23	2:23	5:22	2:03
bqp250-6	251;0	251;	4722	5380	8140	3480	-1.2-6	-1.2-6	-1.2-6	-2.6-7	9.9-7	9.9-7	9.9-7	9.9-7	1:27	1:35	2:41	1:34
bqp250-7	251;0	251;	4470	5138	9243	3128	-1.2-6	-1.7-6	-2.2-7	-1.2-6	9.9-7	9.9-7	9.9-7	9.9-7	1:22	1:28	2:55	1:25
bqp250-8	251;0	251;	2961	3534	6122	2126	-3.3-7	-6.5-7	-4.2-7	-5.8-8	9.9-7	9.9-7	9.9-7	9.9-7	55	1:00	1:57	57
bqp250-9	251;0	251;	4745	6121	10214	3440	-5.9-8	-4.0-7	-6.8-7	-3.3-7	9.9-7	9.9-7	9.9-7	9.9-7	1:25	1:45	3:14	1:34
bqp250-10	251;0	251;	3342	3992	6399	2122	-1.1-6	-1.2-6	3.4-6	-1.1-6	9.9-7	9.9-7	9.9-7	9.9-7	1:01	1:06	2:01	57
bqp500-1	501;0	501;	6473	6932	14654	4086	-1.4-6	-3.4-7	-1.3-6	-1.6-6	9.9-7	9.9-7	9.9-7	9.9-7	8:35	9:45	23:07	9:13
bqp500-2	501;0	501;	8008	10582	22875	4862	-4.2-7	-8.6-8	-2.4-8	-1.2-6	9.9-7	9.9-7	9.9-7	9.9-7	10:46	14:42	38:45	10:52
bqp500-3	501;0	501;	8192	8915	25000	4965	-1.5-6	3.7-7	-1.5-3	-5.8-7	9.9-7	9.9-7	1.7-4	9.9-7	12:53	12:25	39:46	11:22

TABLE 2
(cont.).

bqp500-4	501;0	501;	7188	9012	20498	4031	9.9-7	9.9-7	9.9-7	9.9-7	-1.0-6	-3.8-7	8.5-7	-1.2-6	10:37	12:10	32:48	9:11
bqp500-5	501;0	501;	6898	7641	25000	4541	9.9-7	9.9-7	9.9-7	**1.6-4**	-8.9-7	-1.2-6	**-1.4-3**	-8.2-7	10:34	10:57	41:02	10:19
bqp500-6	501;0	501;	6819	7010	17716	4236	9.9-7	9.9-7	9.9-7	9.9-7	-8.9-7	-1.4-6	3.2-6	-7.4-7	10:22	9:43	28:59	9:37
bqp500-7	501;0	501;	6878	8592	17128	4587	9.9-7	9.9-7	9.9-7	9.9-7	-5.4-7	-7.9-8	1.7-6	-5.2-7	10:23	12:40	28:01	10:33
bqp500-8	501;0	501;	7131	7647	17889	4867	9.9-7	9.9-7	9.9-7	9.9-7	-5.3-7	-9.0-7	-3.3-6	-7.5-7	10:54	10:22	27:42	11:02
bqp500-9	501;0	501;	6666	6700	17784	3803	9.9-7	9.9-7	9.9-7	9.9-7	-1.7-6	-1.2-6	1.5-6	-8.1-7	10:21	9:31	29:58	8:43
bqp500-10	501;0	501;	7189	9162	25000	5067	9.9-7	9.9-7	8.5-5	9.9-7	-1.4-6	-1.3-6	**7.1-4**	-1.6-6	10:43	12:37	40:40	11:34
gka1f	501;0	501;	6717	8147	16790	4600	9.9-7	9.9-7	9.9-7	9.9-7	-1.3-6	-1.2-6	-3.9-6	-5.6-7	9:38	11:28	27:55	10:31
gka2f	501;0	501;	7519	8949	15786	5403	9.9-7	9.9-7	9.9-7	9.9-7	-1.5-6	-1.4-6	-3.3-6	-1.0-6	10:50	12:52	26:27	12:10
gka3f	501;0	501;	6102	7037	13874	3957	9.9-7	9.9-7	9.9-7	9.9-7	-1.1-6	-2.0-6	-6.8-6	-1.6-7	9:07	10:46	23:23	9:07
gka4f	501;0	501;	6673	7529	19384	4070	9.9-7	9.9-7	9.9-7	9.9-7	-1.1-6	-4.2-7	-1.2-6	-3.5-7	9:13	11:20	31:45	9:14
gka5f	501;0	501;	6482	7023	16067	4210	9.9-7	9.9-7	9.9-7	9.9-7	-5.9-7	-9.4-7	1.5-6	-7.5-7	9:14	10:36	26:17	9:45
soybean-large.2	308;0	307;	1190	5050	12972	2261	9.2-7	9.9-7	9.9-7	9.9-7	-7.7-8	-1.2-7	-1.4-7	-7.0-8	29	3:45	5:32	3:09
soybean-large.3	308;0	307;	922	5993	9156	2159	8.8-7	9.6-7	9.7-7	8.5-7	-2.1-7	-5.4-9	-1.2-10	-1.4-7	24	4:47	3:59	3:54
soybean-large.4	308;0	307;	1609	13512	13201	3831	9.9-7	9.9-7	9.9-7	9.9-7	-2.8-7	-1.2-7	-1.1-7	-1.6-7	42	10:51	5:48	7:18
soybean-large.5	308;0	307;	850	2974	5410	1404	9.7-7	9.9-7	9.9-7	9.9-7	-9.1-8	-7.9-8	-8.4-8	-1.7-7	23	2:23	2:22	2:05
soybean-large.6	308;0	307;	413	545	1365	681	9.4-7	6.8-7	9.7-7	9.1-7	-1.9-7	3.5-8	3.4-7	1.3-6	12	21	37	44
soybean-large.7	308;0	307;	1042	3443	5001	1422	9.9-7	9.9-7	9.9-7	9.9-7	-2.5-8	-1.2-8	-8.9-8	-5.4-8	29	2:44	2:19	2:10
soybean-large.8	308;0	307;	741	2294	2176	1456	9.9-7	9.9-7	9.8-7	9.9-7	-1.5-7	-3.3-8	-3.7-8	8.0-8	21	1:46	1:01	1:52
soybean-large.9	308;0	307;	948	3585	5801	2059	9.9-7	9.9-7	9.9-7	9.9-7	1.4-7	-7.1-8	-9.1-8	-5.3-9	25	2:54	2:44	3:01
soybean-large.10	308;0	307;	359	434	627	1789	9.5-7	9.6-7	9.7-7	9.7-7	-1.0-7	8.1-7	4.4-7	-5.0-7	11	11	18	1:58
soybean-large.11	308;0	307;	948	1231	1914	1609	6.5-7	9.2-7	9.9-7	9.1-7	1.0-6	-2.6-6	1.6-6	-2.7-6	26	32	55	1:50
spambase-large.2	1501;0	1500;	535	992	4528	4429	9.9-7	9.9-7	9.9-7	9.9-7	-1.3-5	-1.2-5	-1.3-5	-1.3-5	11:17	22:17	1:51:27	3:12:36
spambase-large.3	1501;0	1500;	1705	1830	5301	6617	9.8-7	9.9-7	9.9-7	9.9-7	-7.6-6	-6.6-6	-1.1-5	-3.3-6	35:47	58:10	2:14:58	6:13:50
spambase-large.4	1501;0	1500;	3761	7091	10801	25000	9.9-7	9.9-7	**2.2-2**	**1.1-2**	9.4-8	-5.2-7	-4.2-7	**-10.0-1**	1:19:26	5:32:29	4:33:51	17:57:38
spambase-large.5	1501;0	1500;	8398	7510	10621	25000	9.9-7	9.8-7	9.9-7	**1.1-2**	-2.9-5	-2.4-5	2.4-5	**3.0-1**	3:26:14	3:21:11	4:32:59	18:14:24
spambase-large.6	1501;0	1500;	2031	2415	4674	25000	9.9-7	9.9-7	9.9-7	**1.8-2**	-4.2-5	-5.8-5	-3.0-5	**-10.0-1**	49:32	1:07:56	1:54:11	17:07:48
spambase-large.7	1501;0	1500;	1596	1584	3001	6042	9.9-7	9.9-7	9.9-7	9.9-7	-1.8-5	-1.0-5	-3.2-5	-1.5-7	36:51	50:54	1:19:23	6:02:07
spambase-large.8	1501;0	1500;	1449	1461	2787	6050	9.9-7	9.9-7	9.9-7	9.9-7	-5.0-5	-9.5-5	-6.6-5	9.2-5	31:01	39:20	1:16:31	4:16:11
spambase-large.9	1501;0	1500;	2010	1973	3695	9832	9.9-7	9.9-7	9.9-7	**1.4-5**	-9.3-5	**-1.4-4**	3.1-6	**4.5-4**	43:12	57:24	1:33:19	7:27:54
spambase-large.10	1501;0	1500;	2728	2450	4401	25000	9.7-7	9.9-7	9.9-7	9.8-7	**1.3-4**	**-1.3-4**	**-3.8-5**	**1.8-3**	1:00:43	1:12:14	1:52:10	18:26:46
spambase-large.11	1501;0	1500;	2704	2526	3496	4532	9.7-7	9.7-7	9.9-7	9.8-7	**1.8-4**	**-1.7-4**	-9.1-6	**1.5-4**	1:04:55	1:08:26	1:30:21	3:52:02
abalone-large.2	1001;0	1000;	576	650	3168	1493	9.9-7	9.9-7	9.9-7	9.9-7	1.2-5	6.6-6	-2.9-6	-1.4-6	5:07	5:08	26:57	31:13
abalone-large.3	1001;0	1000;	765	796	3141	1306	9.9-7	9.9-7	9.9-7	9.9-7	-3.6-6	-9.9-7	-1.7-6	-4.2-6	6:09	8:56	26:36	22:21
abalone-large.4	1001;0	1000;	545	629	1457	710	9.9-7	9.6-7	9.9-7	9.9-7	1.9-6	-6.9-6	2.4-5	-9.2-7	6:50	5:01	12:25	12:03
abalone-large.5	1001;0	1000;	834	1107	1429	833	9.9-7	9.9-7	9.9-7	9.9-7	-1.5-5	-2.1-5	-1.1-5	-2.1-5	8:39	9:11	12:20	14:17
abalone-large.6	1001;0	1000;	796	1101	1864	950	9.9-7	9.9-7	9.9-7	9.9-7	-1.4-5	-1.8-5	-1.9-5	-1.9-5	8:21	8:49	16:08	15:24
abalone-large.7	1001;0	1000;	1089	1388	2156	1230	9.9-7	9.9-7	9.9-7	9.8-7	-2.1-5	-2.7-6	-1.4-5	-1.8-5	10:57	11:52	18:32	25:24

TABLE 2
(cont.).

abalone-large.10	1001;0	1000;	1855	2022	2781	3093	9.8-7	8.6-7	9.7-7	9.9-7	-2.2-5	-6.1-5	-4.8-5	-9.2-5	16:45	16:25	25:57	50:13
abalone-large.11	1001;0	1000;	2212	2604	3724	3118	9.9-7	9.9-7	9.9-7	9.9-7	-4.1-5	9.9-6	-1.9-5	-4.7-5	19:27	21:44	34:06	55:26
segment-large.2	1001;0	1000;	1264	1080	4201	1745	9.9-7	9.8-7	8.9-7	9.9-7	5.0-6	-4.7-6	3.5-8	-5.0-7	9:15	8:27	36:13	34:22
segment-large.3	1001;0	1000;	373	412	3714	1956	9.9-7	9.8-7	9.9-7	9.9-7	1.8-6	-7.1-7	-5.4-6	-1.1-6	2:41	3:33	31:35	37:08
segment-large.4	1001;0	1000;	2024	19479	21201	6354	9.9-7	9.9-7	9.9-7	9.9-7	-5.5-7	-4.5-7	-4.5-7	-5.0-7	14:50	5:23:13	3:05:07	3:07:06
segment-large.5	1001;0	1000;	2711	22003	25000	8257	9.9-7	9.9-7	1.0-6	9.9-7	-6.7-7	-6.0-7	-5.8-7	-6.4-7	20:31	6:09:59	3:44:53	4:19:44
segment-large.6	1001;0	1000;	3262	25000	25000	10211	9.9-7	1.3-6	1.3-6	9.9-7	-1.5-6	-9.6-7	-1.0-6	-1.0-6	24:06	7:10:04	3:38:02	5:25:59
segment-large.7	1001;0	1000;	3600	25000	25000	11657	9.9-7	1.8-6	1.7-6	9.9-7	-1.3-6	-1.9-6	-1.8-6	-1.3-6	27:48	7:15:10	3:40:26	6:13:44
segment-large.8	1001;0	1000;	3161	20284	25000	9511	9.9-7	9.9-7	1.2-6	9.9-7	-1.1-6	-9.4-7	-1.1-6	-1.1-6	24:42	5:46:25	3:47:00	5:15:17
segment-large.9	1001;0	1000;	2383	12121	14501	8064	9.9-7	9.9-7	9.9-7	9.9-7	-1.9-6	-5.3-7	-6.2-7	-1.1-6	18:03	3:23:40	2:12:38	4:10:03
segment-large.10	1001;0	1000;	1789	1676	4701	4527	9.9-7	9.9-7	9.9-7	9.9-7	-3.1-7	-6.1-6	-2.8-7	-3.0-7	13:42	13:51	43:14	1:53:18
segment-large.11	1001;0	1000;	1683	1827	2225	25000	9.9-7	9.9-7	9.9-7	2.9-5	-1.9-6	6.0-6	-1.5-7	1.1-4	13:07	15:29	20:12	6:50:17
housing.2	507;0	506;	3284	2679	5397	2566	9.6-7	9.9-7	9.9-7	8.6-7	-5.4-6	-5.2-6	5.8-6	-5.3-6	4:31	3:26	8:32	7:52
housing.3	507;0	506;	1247	1523	2411	1338	9.9-7	9.9-7	9.9-7	9.8-7	8.0-6	-6.7-6	3.2-5	5.2-6	1:34	1:56	3:41	4:29
housing.4	507;0	506;	1368	1064	2008	1090	9.9-7	9.9-7	9.9-7	8.4-7	-3.5-6	-4.9-6	-4.2-6	8.3-5	2:00	1:25	3:05	3:40
housing.5	507;0	506;	1319	1916	2593	1451	9.6-7	9.3-7	9.9-7	8.8-7	-3.2-5	3.6-5	3.1-5	6.3-5	2:07	2:36	4:04	5:03
housing.6	507;0	506;	536	842	964	1958	9.9-7	9.8-7	9.9-7	9.5-7	-9.7-6	5.9-6	-1.7-5	6.3-5	53	1:20	1:32	6:29
housing.7	507;0	506;	645	856	1147	2235	9.9-7	9.8-7	9.9-7	9.9-7	-2.6-5	-4.6-5	-3.1-5	-7.5-5	1:06	1:15	1:51	7:35
housing.8	507;0	506;	638	924	1017	1700	9.8-7	9.7-7	9.9-7	9.5-7	-1.9-5	-1.3-5	-1.7-5	-5.6-5	1:05	1:23	1:43	5:40
housing.9	507;0	506;	794	1173	1634	2466	9.5-7	9.8-7	9.9-7	9.9-7	-3.7-5	3.7-5	-4.7-5	3.8-5	1:27	1:43	2:50	8:23
housing.10	507;0	506;	1016	1275	1538	25000	9.9-7	9.9-7	9.9-7	6.4-5	-1.7-5	-2.6-5	-2.6-5	2.2-3	1:40	1:57	2:39	1:18:42
housing.11	507;0	506;	844	1310	1342	25000	9.9-7	9.6-7	9.9-7	6.7-5	-2.9-5	-2.5-5	-2.9-5	-7.4-3	1:24	1:54	2:15	1:20:22

Let $\mathcal{D} : \mathcal{S}^n \to \mathcal{S}^n$ be a given nonsingular linear operator and $\mathcal{D}^*$ be its adjoint. By introducing an extra variable, we can reformulate (5.20) into the form of (4.8), for which our proposed Conic-sPADMM3c can be used.

For the SDP problems described in (5.5) arising from relaxing the BIQ problems, in order to get tighter bounds, we may add in some valid inequalities to get the following problems:

(5.20)

$$\min \frac{1}{2}\langle Q, Y \rangle + \langle c, x \rangle$$

$$\text{s.t.} \quad \text{diag}(Y) - x = 0, \quad \alpha = 1, \quad X = \begin{bmatrix} Y & x \\ x^T & \alpha \end{bmatrix} \in \mathcal{S}^n_+, \quad X \in \mathcal{K}_\mathcal{P},$$

$$-Y_{ij} + x_i \geq 0, \ -Y_{ij} + x_j \geq 0, \ Y_{ij} - x_i - x_j \geq -1, \ \forall \, i < j, \ j = 2, \ldots, n-1,$$

where $\mathcal{K}_\mathcal{P} = \{X \in \mathcal{S}^n \mid X \geq 0\}$. For convenience, we call the problem in (5.21) an extended BIQ problem. Note that the last set of inequalities in (5.21) are obtained from the valid inequalities $x_i(1 - x_j) \geq 0$, $x_j(1 - x_i) \geq 0$, $(1 - x_i)(1 - x_j) \geq 0$ when x_i, x_j are binary variables.

Note that one may also apply the directly extended ADMM (see (1.4)) with 4 blocks by adding a proximal term for the y_I-part (similar to the discussion in section 4.2). We call this method sPADMM4d. Of course, we are mindful that sPADMM4d has no convergence guarantee. In this subsection, we compare the algorithms sPADMM3c, sPADMM4d, and a convergent linearized alternating direction method with a Gaussian back substitution proposed in [21] (we call the method LADMM4g here and use the parameter $\alpha = 0.999$ in the Gaussian back substitution step) for the extended BIQ problems (5.21). We have implemented sPADMM3c, sPADMM4d, and LADMM4g in MATLAB. For sPADMM4d, we set the step-length $\tau = 1.618$. For the purpose of comparison, we also test the directly extended sPADMM with unit step-length (i.e., $\tau = 1$), which is called sPADMM4d(1). The computational results for all the extended BIQ problems are obtained on the same Linux server as before.

We note here that neither SDPAD nor 2EBD can be directly applied to solve the problems (5.21). One may of course try to first convert the inequality constraints into linear equalities by introducing slack variables and then apply both SDPAD and 2EBD to the reformulated problems. However, such an approach is inefficient as the linear system of equations which needs to be solved at each iteration is very large but not so sparse. Not surprisingly, this approach is very slow for the extended BIQ problems according to our numerical experience.

We measure the accuracy of an approximate optimal solution (X, y_E, y_I, S, Z) for (SDP) and its dual (5.20) by using the following relative residual:

(5.21) $$\eta = \max\{\eta_P, \eta_D, \eta_\mathcal{K}, \eta_\mathcal{P}, \eta_{\mathcal{K}^*}, \eta_{\mathcal{P}^*}, \eta_{C_1}, \eta_{C_2}, \eta_I, \eta_{I^*}\},$$

where $\eta_\mathcal{K}, \eta_\mathcal{P}, \eta_{\mathcal{K}^*}, \eta_{\mathcal{P}^*}, \eta_{C_1}, \eta_{C_2}$ are defined as in (5.13), and

$$\eta_P = \frac{\|\mathcal{A}_E X - b_E\|}{1 + \|b_E\|}, \ \eta_D = \frac{\|\mathcal{A}_E^* y_E + \mathcal{A}_I^* y_I + S + Z - C\|}{1 + \|C\|},$$

$$\eta_I = \frac{\|\max(0, b_I - \mathcal{A}_I X)\|}{1 + \|b_I\|}, \ \eta_{I^*} = \frac{\|\max(0, -y_I)\|}{1 + \|y_I\|}.$$
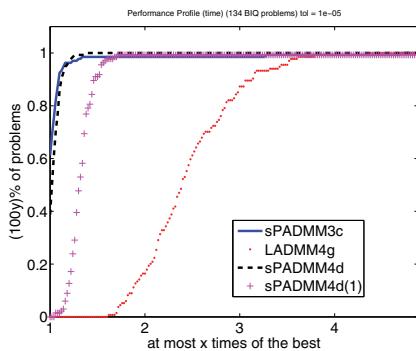
FIG. 3. *Performance profiles (time) of* sPADMM3c, LADMM4g, sPADMM4d, *and* sPADMM4d(1).
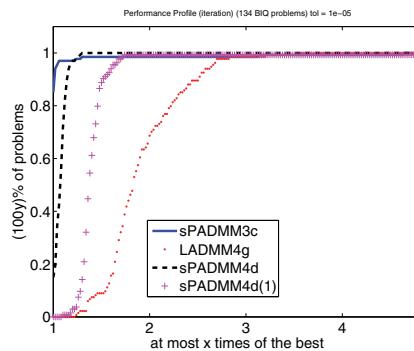
FIG. 4. *Performance profiles (iteration) of* sPADMM3c, LADMM4g, sPADMM4d, *and* sPADMM4d(1).

Additionally, we compute the relative gap by

$$(5.22) \qquad \eta_g = \frac{\langle C, X \rangle - (\langle b_E, y_E \rangle + \langle b_I, y_I \rangle)}{1 + |\langle C, X \rangle| + |\langle b_E, y_E \rangle + \langle b_I, y_I \rangle|}.$$

We terminate sPADMM3c, sPADMM4d, sPADMM4d(1), and LADMM4g when $\eta < 10^{-5}$ or when they reach the maximum number of 50,000 iterations.

In Table 3, we report some detailed numerical results for the solvers sPADMM3c, LADMM4g, sPADMM4d, and sPADMM4d(1) in solving a collection of 134 extended BIQ problems.

Figure 3 shows the performance profiles in terms of computing time for sPADMM3c, LADMM4g, sPADMM4d, and sPADMM4d(1) in solving 134 extended BIQ problems. One can observe that LADMM4g is much slower than the other three solvers. The solver sPADMM3c is clearly more efficient than the directly extended sPADMM with unit step-length, i.e., sPADMM4d(1), and it is even faster than sPADMM4d with $\tau = 1.618$, though only marginally.

Figure 4 shows the performance profiles in terms of the number of iterations for sPADMM3c, LADMM4g, sPADMM4d, and sPADMM4d(1). Observe that for the majority of the test problems, sPADMM3c takes fewer iterations than sPADMM4d with step-length $\tau = 1.618$. However, for some test problems, due to the overhead incurred in handling the additional matrix variable introduced to reformulate (5.20) into the form (4.8), sPADMM3c may take slightly more time than sPADMM4d, even though the latter may take slightly more iterations.

6. Conclusions. In this paper, we have proposed a 3-block semiproximal ADMM that is both convergent and efficient for finding a solution of medium accuracy to conic programming problems with four types of constraints. By conducting numerical experiments on a large number of DNN-SDP problems with equality and/or inequality constraints, we have presented convincing numerical results showing that for the vast majority of problems tested, our proposed (semiproximal) ADMM is at least 20% faster than the directly extended (semiproximal) ADMM with unit step-length. At least for the class of conic programming (P) problems, we can safely say that we have resolved the dilemma that an ADMM is either efficient in practice but without convergent guarantee in theory or the contrary. This opens up the possibility of designing a convergent and yet practically efficient ADMM with an intelligent

TABLE 3

The performance of sPADMM3C, LADMM4G, sPADMM4D, sPADMM4D(1) on extended BIQ problems (accuracy = 10^{-5}). In the table, we have omitted the command string "ADMM" in the names of the solvers. The computation time is in the format hours:minutes:seconds.

Problem	$m_E;m_I$	$n_s;$	Iteration				η				η_g				Time			
			P3c	L4g	P4d	P4d(1)	P3c	L4g	P4d	P4d(1)	P3c	L4g	P4d	P4d(1)	P3c	L4g	P4d	P4d(1)
be200.3.1	201:59700	201;	8943	14140	9729	12236	9.9-6	9.9-6	9.9-6	9.9-6	3.2-7	-7.1-6	-1.8-6	-1.4-7	7:34	12:45	7:22	9:04
be200.3.2	201:59700	201;	7665	14170	8854	11243	9.9-6	9.9-6	9.9-6	9.9-6	7.8-6	-7.1-6	2.4-6	7.7-6	5:27	15:12	5:54	7:56
be200.3.3	201:59700	201;	11187	22332	11429	14882	9.9-6	9.9-6	9.9-6	9.9-6	8.9-7	-7.0-6	2.4-7	-7.2-7	8:58	20:21	9:29	10:50
be200.3.4	201:59700	201;	9513	21265	9628	12833	9.9-6	9.9-6	9.9-6	9.9-6	1.1-6	-7.1-6	-1.7-6	-1.3-6	7:11	21:43	6:58	9:59
be200.3.5	201:59700	201;	9373	15368	9370	12526	9.9-6	9.9-6	9.9-6	9.9-6	-4.3-7	-1.2-5	-2.9-6	-9.7-7	8:43	14:15	8:21	10:14
be200.3.6	201:59700	201;	8777	13901	10180	12967	9.9-6	9.8-6	9.9-6	9.9-6	-5.0-6	-1.1-5	-9.6-6	-7.3-6	7:07	14:23	7:53	9:22
be200.3.7	201:59700	201;	12965	24430	13904	17684	9.9-6	9.9-6	9.9-6	9.9-6	-2.6-6	-8.2-6	-3.5-6	-3.7-6	10:52	22:31	12:29	14:15
be200.3.8	201:59700	201;	11102	18030	10906	14622	9.9-6	9.9-6	9.9-6	9.9-6	3.7-7	-9.7-6	-8.5-7	-1.6-6	8:24	19:35	7:46	11:03
be200.3.9	201:59700	201;	9612	17226	9806	13405	9.9-6	9.9-6	9.9-6	9.9-6	-4.3-6	-9.9-6	-6.6-6	-3.1-6	9:10	16:18	8:29	10:11
be200.3.10	201:59700	201;	7964	14501	8851	10934	9.9-6	9.9-6	9.9-6	9.9-6	2.3-6	-8.6-6	-4.8-6	8.1-7	6:31	14:00	6:44	8:06
be200.8.1	201:59700	201;	11162	25001	11862	15840	9.9-6	9.9-6	9.9-6	9.9-6	-3.5-6	-7.8-6	-4.9-6	-5.0-6	8:32	23:29	9:12	11:34
be200.8.2	201:59700	201;	8392	14101	8693	12297	9.9-6	9.8-6	9.9-6	9.9-6	9.0-6	-8.0-6	4.1-6	5.6-6	5:46	13:34	5:47	8:06
be200.8.3	201:59700	201;	10385	20301	10882	14115	9.9-6	9.9-6	9.9-6	9.9-6	-5.2-6	-8.5-6	-5.8-6	-6.0-6	9:42	18:05	9:46	10:56
be200.8.4	201:59700	201;	9457	17380	10001	14201	9.9-6	9.9-6	9.9-6	9.9-6	7.4-7	-7.9-6	7.4-7	9.8-7	7:31	17:57	8:12	10:56
be200.8.5	201:59700	201;	10011	19060	10008	12977	9.9-6	9.9-6	9.9-6	9.9-6	3.6-6	-7.1-6	-3.6-7	-4.2-7	8:18	17:23	8:28	10:12
be200.8.6	201:59700	201;	11144	28038	12027	14979	9.9-6	9.9-6	9.9-6	9.9-6	1.8-6	-7.3-6	3.5-7	9.7-7	9:07	28:44	10:10	12:19
be200.8.7	201:59700	201;	9261	20201	9226	12573	9.9-6	9.9-6	9.9-6	9.9-6	6.9-6	-6.3-6	7.2-6	7.2-6	8:39	19:05	7:58	10:04
be200.8.8	201:59700	201;	11002	22942	11331	15218	9.9-6	9.9-6	9.9-6	9.9-6	-8.5-6	-8.6-6	-9.3-6	-9.2-6	9:13	23:57	9:08	12:22
be200.8.9	201:59700	201;	9102	16901	9945	12465	9.9-6	9.9-6	9.9-6	9.9-6	-2.5-6	-9.2-6	-5.0-6	-4.3-6	7:20	16:18	7:38	8:39
be200.8.10	201:59700	201;	9137	16001	9905	13232	9.9-6	9.9-6	9.9-6	9.9-6	9.3-6	-6.7-6	9.0-6	7.3-6	7:19	15:36	6:54	10:21
be250.1	251:93375	251;	13774	24878	15373	19553	9.9-6	9.9-6	9.9-6	9.9-6	-2.6-6	-1.2-5	-4.7-6	-4.0-6	17:50	38:20	21:43	24:28
be250.2	251:93375	251;	11808	21286	12138	16078	9.9-6	9.9-6	9.9-6	9.9-6	5.6-6	-9.6-6	3.8-6	3.8-6	14:38	35:35	15:55	19:06
be250.3	251:93375	251;	14517	27463	16134	19608	9.9-6	9.9-6	9.9-6	9.9-6	1.4-6	-1.3-5	7.5-7	1.0-6	19:53	47:55	19:41	23:42
be250.4	251:93375	251;	14715	28926	16367	21328	9.9-6	9.9-6	9.9-6	9.9-6	6.3-6	-9.3-6	5.9-7	3.1-6	21:30	52:33	21:16	29:05
be250.5	251:93375	251;	11316	18201	12801	16164	9.9-6	9.9-6	9.9-6	9.9-6	5.2-6	-1.4-5	1.2-6	1.6-6	15:56	26:40	14:47	18:54
be250.6	251:93375	251;	14606	25701	15956	21016	9.9-6	9.9-6	9.9-6	9.9-6	-3.3-6	-1.1-5	-5.1-6	-4.6-6	18:29	41:31	18:35	23:59
be250.7	251:93375	251;	14442	27701	17230	21587	9.9-6	9.9-6	9.9-6	9.9-6	-4.8-6	-1.0-5	-6.4-6	-6.5-6	19:09	48:12	26:17	28:05
be250.8	251:93375	251;	14305	24976	15139	19218	9.9-6	9.9-6	9.9-6	9.9-6	1.5-7	-1.1-5	-1.5-6	-1.7-6	19:50	44:48	18:16	24:24
be250.9	251:93375	251;	10701	16508	12268	15170	9.9-6	9.9-6	9.9-6	9.9-6	9.3-7	-1.8-5	-2.7-6	-9.4-7	13:57	26:28	14:48	17:45
be250.10	251:93375	251;	12160	20301	13176	15343	9.9-6	9.9-6	9.9-6	9.9-6	7.9-6	-6.9-6	6.6-6	7.5-6	15:40	36:24	16:29	18:13
bqp250-1	251:93375	251;	14332	29301	14903	18899	9.9-6	9.9-6	9.9-6	9.9-6	2.1-6	-9.8-6	1.6-6	1.3-6	19:47	49:11	20:47	23:03
bqp250-2	251:93375	251;	12301	22601	13322	17692	9.9-6	9.9-6	9.9-6	9.9-6	-6.1-6	-9.9-6	-9.6-6	-8.3-6	16:06	39:47	15:38	21:42
bqp250-3	251:93375	251;	15229	26201	16301	20920	9.9-6	9.9-6	9.9-6	9.9-6	8.3-6	-1.1-5	7.9-6	6.4-6	21:15	42:39	21:56	25:21
bqp250-4	251:93375	251;	12303	22801	12918	16310	9.9-6	9.9-6	9.9-6	9.9-6	2.8-7	-9.3-6	-1.3-6	-9.3-7	17:19	40:40	15:03	21:49
bqp250-5	251:93375	251;	13210	27801	14302	18589	9.9-6	9.9-6	9.9-6	9.9-6	-1.1-6	-1.0-5	-4.7-6	-4.6-6	17:39	41:46	17:15	21:05
bqp250-6	251:93375	251;	11312	19910	10823	14626	9.9-6	9.9-6	9.9-6	9.9-6	4.5-6	-1.0-5	-2.8-6	4.4-7	14:11	32:17	12:27	16:50

TABLE 3
(cont.).

bqp250-7	251;93375	251;	13962	24701	15631	20411	9.9-6	9.9-6	9.9-6	6.8-6	-1.1-5	5.3-6	5.7-6	20:45	40:32	20:42	23:42
bqp250-8	251;93375	251;	10601	16801	11459	14972	9.9-6	9.9-6	9.9-6	-10.0-7	-1.2-5	-3.1-6	-1.9-6	13:08	27:56	14:00	17:30
bqp250-9	251;93375	251;	14855	25470	16401	21439	9.9-6	9.9-6	9.9-6	-7.3-6	-1.1-5	-7.9-6	-8.7-6	23:11	44:17	23:15	27:09
bqp250-10	251;93375	251;	10344	18901	11268	13902	9.9-6	9.9-6	9.9-6	7.5-6	-6.5-6	2.3-6	5.3-6	15:31	32:22	15:14	15:51
bqp500-1	501;374250	501;	17258	26843	18284	23591	9.9-6	9.9-6	9.9-6	3.3-6	-1.4-5	-5.2-6	-5.1-6	1:57:15	3:55:29	1:58:50	2:26:39
bqp500-2	501;374250	501;	18453	36844	20364	25027	9.9-6	9.9-6	9.9-6	5.4-6	-9.4-6	2.1-6	4.5-6	2:06:15	5:38:45	2:00:28	2:25:11
bqp500-3	501;374250	501;	19161	31404	19904	25469	9.9-6	9.9-6	9.9-6	9.7-6	-1.8-5	7.8-6	8.7-6	1:58:24	4:50:34	1:42:24	2:09:39
bqp500-4	501;374250	501;	16801	32116	18402	22449	9.9-6	9.9-6	9.9-6	1.1-5	-1.1-5	8.1-6	7.9-6	1:34:01	3:57:05	1:32:31	1:52:18
bqp500-5	501;374250	501;	17522	30985	18789	23397	9.9-6	9.9-6	9.9-6	3.4-6	-1.0-5	-1.9-6	5.2-7	1:57:31	4:54:17	2:01:25	2:27:00
bqp500-6	501;374250	501;	17826	31655	19213	24230	9.9-6	9.9-6	9.9-6	-1.1-6	-9.4-6	-2.3-6	-2.8-6	1:47:20	4:40:17	1:39:56	2:14:33
bqp500-7	501;374250	501;	18004	29439	18232	23837	9.9-6	9.9-6	9.9-6	2.8-6	-1.3-5	3.1-8	-3.1-7	1:58:09	3:57:05	1:35:35	1:58:49
bqp500-8	501;374250	501;	18685	30701	20343	50000	9.9-6	9.9-6	1.8-5	2.8-6	-1.1-5	1.3-6	-1.8-5	2:05:47	4:26:00	2:08:58	4:59:12
bqp500-9	501;374250	501;	17648	27801	18504	23491	9.9-6	9.9-6	9.9-6	2.3-6	-1.5-5	1.3-6	-1.4-6	1:47:18	3:52:54	1:40:11	1:57:46
bqp500-10	501;374250	501;	18128	34147	19935	25577	9.9-6	9.9-6	9.9-6	7.9-6	-1.1-5	5.9-6	4.8-6	1:47:23	4:11:38	1:42:53	2:16:53
gka1e	201;59700	201;	13605	34550	15348	19616	9.9-6	9.9-6	9.9-6	-7.2-6	-1.2-5	-7.5-6	-7.7-6	10:15	31:16	11:27	14:09
gka2e	201;59700	201;	9104	15606	10010	12613	9.9-6	9.9-6	9.9-6	-4.8-6	-8.6-6	-7.3-6	-6.4-6	7:22	13:47	9:00	9:58
gka3e	201;59700	201;	10503	18001	10402	13767	9.9-6	9.9-6	9.9-6	-5.6-7	-7.9-6	-2.8-6	-2.3-6	7:52	17:59	8:32	10:52
gka4e	201;59700	201;	10713	21201	11413	14151	9.9-6	9.9-6	9.9-6	5.7-6	-8.8-6	1.5-6	5.2-7	9:10	19:17	8:27	11:23
gka5e	201;59700	201;	9506	17470	9806	13201	9.9-6	9.9-6	9.9-6	4.7-6	-7.4-6	1.8-6	-7.5-8	7:04	17:57	6:55	9:30
gka1f	501;374250	501;	16313	27102	17074	21619	9.9-6	9.9-6	9.9-6	8.8-6	-9.9-6	-2.6-7	1.4-6	1:36:46	3:50:26	1:51:22	2:03:09
gka2f	501;374250	501;	18281	29792	19502	23907	9.9-6	9.9-6	9.9-6	4.7-6	-1.1-5	-1.9-6	2.7-7	1:55:57	4:07:38	1:44:35	2:20:24
gka3f	501;374250	501;	17311	29801	17992	24553	9.9-6	9.9-6	9.9-6	7.2-6	-9.9-6	7.7-6	-2.0-7	2:04:10	4:35:38	1:55:23	2:12:30
gka4f	501;374250	501;	16714	27846	17360	21744	9.9-6	9.9-6	9.9-6	8.0-6	-1.2-5	8.3-6	4.7-6	1:32:58	3:28:28	1:28:34	1:48:07
gka5f	501;374250	501;	17647	29644	18157	23407	9.9-6	9.9-6	9.9-6	2.0-6	-1.0-5	3.1-6	-1.4-6	1:56:52	4:05:27	1:42:52	2:12:16

BCD cycle rather than the usual nonconvergent Gauss–Seidel BCD cycle for solving multiblock convex optimization problem (1.1). We leave this as one of our future research topics. In fact, our primary motivation of introducing this convergent 3-block semiproximal ADMM is to quickly produce an initial point for conic programming (P) so as to warm-start methods which have fast local convergence properties. For SDP problems in standard form, this has already been done by Zhao, Sun, and Toh in [44] by first using the classic 2-block ADMM (it was called the boundary point method in [31] at that time) to generate a starting point and then switching it to the fast convergent Newton-CG augmented Lagrangian method. The resulting software SDPNAL has been successfully employed by Nie and Wang [27, 28] to solve very large-scale SDP problems in standard form arising from polynomial optimization and rank-1 tensor approximation problems. Naturally, our next target is to extend this approach to multiblock convex optimization problems beyond conic programming (P) in standard form. We will report our corresponding findings in subsequent works.

REFERENCES

[1] J. M. BORWEIN AND A. S. LEWIS, *Convex Analysis and Nonlinear Optimization: Theory and Examples*, Springer, New York, 2006.

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

[3] 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.

[4] C. CHEN, B. HE, Y. YE, AND X. YUAN, *The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent*, Math. Program., Ser. A, to appear.

[5] W. DENG AND W. YIN, *On the Global and Linear Convergence of the Generalized Alternating Direction Method of Multipliers*, Rice CAAM Technical report 12-14, 2014.

[6] J. ECKSTEIN AND D. P. BERTSEKAS, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Program., 55 (1992), pp. 293–318.

[7] J. ECKSTEIN AND W. YAO, *Understanding the convergence of the alternating direction method of multipliers: Theoretical and computational perspectives*, Pac. J. Optim., to appear.

[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] M. FAZEL, T. K. PONG, D. F. SUN, AND P. TSENG, *Hankel matrix rank minimization with applications in system identification and realization*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 946–977.

[10] M. FORTIN AND R. GLOWINSKI, Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary Value Problems, Stud. Math. Appl. 15, Elsevier, Amsterdam, 1983.

[11] D. GABAY, *Applications of the method of multipliers to variational inequalities*, in Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems, M. Fortin and R. Glowinski, eds., Stud. Math. Appl. 15, Elsevier, Amsterdam, 1983, pp. 299–331.

[12] D. GABAY AND B. MERCIER, *A dual algorithm for the solution of nonlinear variational problems via finite element approximation*, Comput. Math. Appl., 2 (1976), pp. 17–40.

[13] R. GLOWINSKI, *Lectures on Numerical Methods for Nonlinear Variational Problems*, Tata Institute of Fundamental Research Lectures on Mathematics and Physics, Tata Inst. Fund. Res., Bombay, 1980.

[14] R. Glowinski and A. Marrocco, *Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité, d'une classe de problèmes de dirichlet non linéares*, Rev. Francaise Automat., Inform. Recherche Opér., 9 (1975), pp. 41–76.

[15] P. Hahn and M. Anjos, *QAPLIB: A Quadratic Assignment Problem Library*, http://www.seas.upenn.edu/qaplib.

[16] D. Han, X. Yuan, W. Zhang, and X. Cai, *An ADM-based splitting method for separable convex programming*, Comput. Optim. Appl., 54 (2013), pp. 343–369.

[17] B. He, M. Tao, M. Xu, and X. Yuan, *An alternating direction-based contraction method for linearly constrained separable convex programming problems*, Optimization, 62 (2013), pp. 573–596.

[18] B. He, M. Tao, and X. Yuan, *Alternating direction method with gaussian back substitution for separable convex programming*, SIAM J. Optim., 22 (2012), pp. 313–340.

[19] B. He, M. Tao, and X. Yuan, *A splitting method for separable convex programming*, IMA J. Numer. Anal., 20 (2014), pp. 1–33.

[20] B. He, H. Yang, and S. Wang, *Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities*, J. Optim. Theory Appl., 106 (2000), pp. 337–356.

[21] B. He and X. Yuan, *Linearized alternating direction method of multipliers with Gaussian back substitution for separable convex programming*, Numer. Algebra Control Optim., 3 (2013), pp. 247–260.

[22] M. R. Hestenes, *Multiplier and gradient methods*, J. Optim. Theory Appl., 4 (1969), pp. 303–320.

[23] M. Y. Hong and Z. Q. Luo, *On the Linear Convergence of the Alternating Direction Method of Multipliers*, preprint, arXiv:1208.3922, 2012.

[24] X. D. Li, D. F. Sun, and K.-C. Toh, *A Schur complement based semi-proximal ADMM for convex quadratic conic programming and extensions*, Math. Program., Ser. A, to appear.

[25] R. D. Monteiro, C. Ortiz, and B. F. Svaiter, *A first-order block-decomposition method for solving two-easy-block structured semidefinite programs*, Math. Program. Comput., 6 (2014), pp. 103–150.

[26] J. J. Moreau, *Décomposition orthogonale d'un espace hilbertien selon deux cones mutuelle-ment polaires*, C. R. Acad. Sci., 255 (1962), pp. 238–240.

[27] J. Nie and L. Wang, *Regularization methods for SDP relaxations in large-scale polynomial optimization*, SIAM J. Optim., 22 (2012), pp. 408–428.

[28] J. Nie and L. Wang, *Semidefinite Relaxations for Best Rank-1 Tensor Approximations*, preprint, arXiv:1308.6562, 2013.

[29] J. Peng and Y. Wei, *Approximating k-means-type clustering via semidefinite programming*, SIAM J. Optim., 18 (2007), pp. 186–205.

[30] J. Povh and F. Rendl, *Copositive and semidefinite relaxations of the quadratic assignment problem*, Discrete Optim., 6 (2009), pp. 231–241.

[31] J. Povh, F. Rendl, and A. Wiegele, *A boundary point method to solve semidefinite programs*, Computing, 78 (2006), pp. 277–286.

[32] M. Powell, *A method for nonlinear constraints in minimization problems*, in Optimization, R. Fletcher, ed., Academic Press, New York, 1969, pp. 283–298.

[33] R. T. Rockafellar, *Convex Analysis*, Princeton Math. Ser. 28, Princeton University Press, Princeton, NJ, 1970.

[34] R. T. Rockafellar, *Augmented Lagrangians and applications of the proximal point algorithm in convex programming*, Math. Oper. Res., 1 (1976), pp. 97–116.

[35] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*, Grundlehren Math. Wiss. 317, Springer-Verlag, Berlin, 1998.

[36] N. Sloane, *Challenge Problems: Independent Sets in Graphs*, http://www.research.att.com/njas/doc/graphs.html (2005).

[37] M. Tao and X. Yuan, *Recovering low-rank and sparse components of matrices from incomplete and noisy observations*, SIAM J. Optim., 21 (2011), pp. 57–81.

[38] K. C. Toh, *Solving large scale semidefinite programs via an iterative solver on the augmented systems*, SIAM J. Optim., 14 (2004), pp. 670–698.

[39] 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* http://dimacs.rutgers.edu/Challenges (1992).

[40] P. Tseng, *Convergence of a block coordinate descent method for nondifferentiable minimization*, J. Optim. Theory Appl., 109 (2001), pp. 475–494.

[41] S. Wang and L. Liao, *Decomposition method with a variable parameter for a class of monotone variational inequality problems*, J. Optim. Theory Appl., 109 (2001), pp. 415–429.

[42] X. WANG, M. HONG, S. MA, AND Z.-Q. LUO, *Solving Multiple-Block Separable Convex Minimization Problems Using Two-Block Alternating Direction Method of Multipliers*, preprint, arXiv:1308.5294, 2013.
[43] Z. WEN, D. GOLDFARB, AND W. YIN, *Alternating direction augmented Lagrangian methods for semidefinite programming*, Math. Program. Comput., 2 (2010), pp. 203–230.
[44] X. Y. ZHAO, D. F. SUN, AND K.-C. TOH, *A Newton-CG augmented Lagrangian method for semidefinite programming*, SIAM J. Optim., 20 (2010), pp. 1737–1765.