HYBRID JACOBIAN AND GAUSS–SEIDEL PROXIMAL BLOCK COORDINATE UPDATE METHODS FOR LINEARLY CONSTRAINED CONVEX PROGRAMMING[*]

YANGYANG XU[†]

Abstract. Recent years have witnessed the rapid development of block coordinate update (BCU) methods, which are particularly suitable for problems involving large-sized data and/or variables. In optimization, BCU first appears as the coordinate descent method that works well for smooth problems or those with separable nonsmooth terms and/or separable constraints. As nonseparable constraints exist, BCU can be applied under primal-dual settings. In the literature, it has been shown that for weakly convex problems with nonseparable linear constraints, BCU with fully Gauss–Seidel updating rule may fail to converge and that with fully Jacobian rule can converge sublinearly. However, empirically the method with Jacobian update is usually slower than that with Gauss–Seidel rule. To maintain their advantages, we propose a hybrid Jacobian and Gauss–Seidel BCU method for solving linearly constrained multiblock structured convex programming, where the objective may have a nonseparable quadratic term and separable nonsmooth terms. At each primal block variable update, the method approximates the augmented Lagrangian function at an affine combination of the previous two iterates, and the affinely mixing matrix with desired nice properties can be chosen through solving a semidefinite program. We show that the hybrid method enjoys the theoretical convergence guarantee as Jacobian BCU. In addition, we numerically demonstrate that the method can perform as well as the Gauss–Seidel method and better than a recently proposed randomized primal-dual BCU method.

Key words. block coordinate update (BCU), Jacobian rule, Gauss–Seidel rule, alternating direction method of multipliers (ADMM)

AMS subject classifications. 90-08, 90C25, 90C06, 68W40

DOI. 10.1137/16M1084705

1. Introduction. Driven by modern applications in image processing and statistical and machine learning, block coordinate update (BCU) methods have been revived in recent years. BCU methods decompose a complicated large-scale problem into easy small subproblems and tend to have low per-update complexity and low memory requirements, and they give rise to powerful ways to handle problems involving large-sized data and/or variables. These methods originate from the coordinate descent method that applies only to optimization problems with separable constraints. Under primal-dual settings, they have been developed to deal with nonseparably constrained problems.

In this paper, we consider the linearly constrained multiblock structured problem with a quadratic term in the objective:

$$(1) \qquad \min_x \frac{1}{2} x^\top Q x + \sum_{i=1}^m g_i(x_i) \quad \text{s.t. } Ax = b,$$

where the variable is partitioned into m disjoint blocks $x = (x_1, x_2, \ldots, x_m)$, Q is a positive semidefinite (PSD) matrix, and each g_i is a proper closed convex and possibly

[†]Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180 (xuy21@rpi.edu).

nondifferentiable function. Note that part of g_i can be an indicator function of a convex set $\mathcal{X}_i$, and thus (1) can implicitly include certain separable block constraints $x_i \in \mathcal{X}_i$ in addition to the nonseparable linear constraint.

Due to its multiblock and also coordinate friendly [38] structure, we will derive a BCU method for solving (1) by performing BCU to x_i's based on the augmented Lagrangian function of (1), followed by an update to the Lagrangian multiplier; see the updates in (6) and (7).

1.1. Motivations. This work is motivated from two aspects. First, many applications can be formulated in the form of (1). Second, although numerous optimization methods can be applied to these problems, few of them are reliable and also efficient. Hence, we need a novel algorithm that can be applied to all these applications and also has a nice theoretical convergence result.

Motivating examples. If g_i is the indicator function of the nonnegative orthant for all i, then it reduces to the nonnegative linearly constrained quadratic program (NLCQP)

$$(2) \qquad \min_x \frac{1}{2}x^\top Q x + d^\top x \quad \text{s.t. } Ax = b, \, x \geq 0.$$

All convex QPs can be written as NLCQPs by adding a slack variable and/or decomposing a free variable into positive and negative parts. As Q is a huge-sized matrix, it would be beneficial to partition it into block matrices and, correspondingly, partition x and A into block variables and block matrices, and then apply BCU methods toward finding a solution to (2).

Another example is the constrained Lasso regression problem proposed in [29]:

$$(3) \qquad \min_x \frac{1}{2}\|Ax - b\|^2 + \mu\|x\|_1 \quad \text{s.t. } Cx \leq d.$$

If there is no constraint $Cx \leq d$, (3) simply reduces to the Lasso regression problem [45]. Introducing a nonnegative slack variable y, we can write (3) in the form of (1):

$$(4) \qquad \min_{x,y} \frac{1}{2}\|Ax - b\|^2 + \mu\|x\|_1 + \iota_+(y) \quad \text{s.t. } Cx + y = d,$$

where $\iota_+(y)$ is the indicator function of the nonnegative orthant, equaling zero if y is nonnegative, and $+\infty$ otherwise. Again for large-sized A or C, it is preferable to partition x into disjoint blocks and apply BCU methods to (4).

There are many other examples arising in signal and image processing and machine learning, such as the compressive principal component pursuit [52] (see (46) below) and regularized multiclass support vector machines (MSVMs) [55] (see (48) below, for instance). More examples, including basis pursuit, conic programming, and the exchange problem, can be found in [11, 21, 26, 31, 43] and the references therein.

Reliable and efficient algorithms. Toward a solution to (1), one may apply any traditional method, such as the projected subgradient method, the augmented Lagrangian method (ALM), and the interior-point method [37]. However, these methods do not utilize the block structure of the problem and are not fit to very large scale problems. To utilize the block structure, BCU methods are preferable. For unconstrained or block-constrained problems, recent works (e.g., [27, 41, 53, 57, 58]) have shown that BCU can be theoretically reliable and also practically efficient. Nonetheless, for problems with nonseparable linear constraints, most existing BCU methods

either require strong assumptions for convergence guarantee or converge slowly; see the review in section 1.2 below. Exceptions include [21, 22] and [9, 15]. However, the former two consider only separable convex problems, i.e., without the nonseparable quadratic term in (1), and the convergence result established by the latter two is stochastic rather than worst-case. In addition, numerically we notice that the randomized method in [15] does not perform so well when the number of blocks is small. Our novel algorithm utilizes the block structure of (1) and also enjoys fast worst-case convergence rate under mild conditions.

1.2. Related works. BCU methods in optimization first appeared in [25] as the coordinate descent (CD) method for solving quadratic programming with separable nonnegative constraints but without nonseparable equality constraints. The CD method updates one coordinate every time, while all the remaining ones are fixed. It may become stuck at a nonstationary point if there are nonseparable nonsmooth terms in the objective; see the examples in [42, 51]. On solving smooth problems or those with separable nonsmooth terms, the convergence properties of the CD method have been intensively studied (see, e.g., [27, 41, 47, 49, 53, 57, 58]). For the linearly constrained problem (1), the CD method can also become stuck at a nonstationary point, for example, if the linear constraint is simply $x_1 - x_2 = 0$. Hence, to directly apply BCU methods to linearly constrained problems, at least two coordinates need to be updated every time; see, for example, [36, 48].

Another way of applying BCU toward finding a solution to (1) is to perform primal-dual BCU (see, e.g., [7, 38, 39, 40]). These methods usually first formulate the first-order optimality system of the original problem and then apply certain operator splitting methods. Assuming monotonicity of the iteratively performed operator (that corresponds to convexity of the objective), almost sure iterate sequence convergence to a solution can be shown, and with a strong monotonicity assumption (that corresponds to strong convexity), linear convergence can be established.

In the literature, there are also plenty of works applying BCU to the augmented Lagrangian function as we did in (6). One popular topic is the alternating direction method of multipliers (ADMM) applied to separable multiblock structured problems, i.e., in the form of (1) without the nonseparable quadratic term. Originally, ADMM was proposed for solving separable two-block problems [13, 17] by cyclically updating the two block variables in a Gauss–Seidel way, followed by an update to the multiplier, and its convergence and also $O(1/t)$ sublinear rate are guaranteed by assuming merely weak convexity (see, e.g., [24, 33, 35]). While directly extended to problems with more than two blocks, ADMM may fail to converge as shown in [4] unless additional assumptions are made, such as strong convexity on part of the objective (see, e.g., [2, 6, 10, 19, 30, 32, 33]), an orthogonality condition on block coefficient matrices in the linear constraint (see, e.g., [4]), and Lipschitz differentiability of the objective and invertibility of the block matrix in the constraint about the last updated block variable (see, e.g., [50]). For problems that do not satisfy these conditions, ADMM can be modified and have guaranteed convergence and even rate estimate by adding extra correction steps (see, e.g., [21, 22]), by using the randomized BCU (see, e.g., [9, 15]), or by adopting Jacobian updating rules (see, e.g., [11, 20, 31]) that essentially reduce the method to proximal ALM or the two-block ADMM method.

When there is a nonseparable term coupling variables together in the objective like (1), existing works usually replace the nonseparable term by a relatively easier majorization function during the iterations and perform the upper-bound or majorized ADMM updates. For example, [26] considers generally multiblock problems with a

nonseparable Lipschitz-differentiable term. Under certain error bound conditions and a diminishing dual stepsize assumption, it shows subsequence convergence; i.e., any cluster point of the iterate sequence is a primal-dual solution. Along a similar direction, [8] specializes the method of [26] to two-block problems and establishes global convergence and also $O(1/t)$ rate with any positive dual stepsize. Without changing the nonseparable term, [16] adds proximal terms into the augmented Lagrangian function during each update and shows $O(1/t)$ sublinear convergence by assuming strong convexity on the objective. Recently, [5] directly applied ADMM to (1) with $m = 2$, i.e., only two block variables, and established iterate sequence convergence to a solution, while no rate estimate has been shown.

1.3. Jacobian and Gauss–Seidel block coordinate update. On updating one among the m block variables, the Jacobi method uses the values of all the other blocks from the previous iteration, while the Gauss–Seidel method always takes the most recent values. For optimization problems without constraint or with block separable constraint, Jacobian BCU enjoys the same convergence as the gradient descent or the proximal gradient method, and Gauss–Seidel BCU is also guaranteed to converge under mild conditions (see, e.g., [27, 41, 47, 57]). For linearly constrained problems, the Jacobi method converges to optimal value with merely weak convexity [11, 20, 23]. However, the Gauss–Seidel update requires additional assumptions for convergence, though it can empirically perform better than the Jacobi method if it happens to converge. Counterexamples are constructed in [4, 12] to show possible divergence of Gauss–Seidel BCU for linearly constrained problems. To guarantee convergence of the Gauss–Seidel update, many existing works assume strong convexity on the objective or part of it. For example, [19] considers linearly constrained convex programs with a separable objective function. It shows the convergence of multiblock ADMM by assuming strong convexity on the objective. Sublinear convergence of multiblock proximal ADMM is established for problems with a nonseparable objective in [16], which assumes strong convexity on the objective and also chooses parameters dependent on the strong convexity constant. In the three-block case, [2] assumes strong convexity on the third block and also full-column-rankness of the last two block coefficient matrices, and [6] assumes strong convexity on the last two blocks. There are also works that do not assume strong convexity but require some other conditions. For instance, [26] considers linearly constrained convex problems with nonseparable objective. It shows the convergence of a majorized multiblock ADMM with diminishing dual stepsizes and by assuming a local error bound condition. The work [4] assumes orthogonality between two block matrices in the linear constraint and proves the convergence of three-block ADMM by reducing it to the classic two-block case. Intuitively, Jacobian block update is a linearized ALM, and thus in some sense it is equivalent to performing an augmented dual gradient ascent to the multipliers [34]. On the contrary, Gauss–Seidel update uses inexact dual gradient, and because the blocks are updated cyclically, the error can accumulate; see [44], where random permutation is performed before block update to cancel the error, and iterate convergence in expectation is established.

1.4. Contributions. We summarize our contributions as follows.
- We propose a hybrid Jacobian and Gauss–Seidel BCU method for solving (1). Through affinely combining the two most recent iterates, the proposed method can update block variables, in a Jacobian or a Gauss–Seidel manner, or a mixture of them, Jacobian rules within groups of block variables, and a Gauss–Seidel way between groups. It can enjoy the theoretical conver-

gence guarantee as the BCU with fully Jacobian update rule, and also enjoy practical fast convergence as the method with fully Gauss–Seidel rule.

- We establish global iterate sequence convergence and $O(1/t)$ rate results of the proposed BCU method, by assuming certain conditions on the affinely mixing matrix (see Assumption 2) and choosing appropriate weight matrix P_i's in (6). In contrast to the after-correction step proposed in [21] to obtain convergence of multiblock ADMM with fully Gauss–Seidel rule, the affine combination of two iterates we propose can be regarded as a correction step before updating every block variable. In addition, our method allows nonseparable quadratic terms in the objective, while the method in [21] can only deal with separable terms. Furthermore, utilizing the prox-linearization technique, our method can have much simpler updates.
- We discuss how to choose the affinely mixing matrix, which can be determined through solving a semidefinite program (SDP); see (40). One can choose a desired matrix by adding certain constraints in the SDP. Compared to the original problem, the SDP is much smaller and can be solved offline. We demonstrate that the algorithm with the combining matrix found in this way can perform significantly better than that with all-ones matrix (i.e., with fully Jacobian rule).
- We apply the proposed BCU method to quadratic programming, the compressive principal component pursuit (see (45) below), and the MSVM (see (48) below). By adapting P_i's in (6), we demonstrate that our method can outperform a randomized BCU method recently proposed in [15] and is comparable to the direct ADMM that has no guaranteed convergence. Therefore, the proposed method can be a more reliable and also efficient algorithm for solving problems in the form of (1).

1.5. Outline. The rest of the paper is organized as follows. In section 2, we present our algorithm, and we show its convergence with sublinear rate estimate in section 3. In section 4, we discuss how to choose the affinely mixing matrices used in the algorithm. Numerical experiments are performed in section 5. Finally, section 6 concludes the paper.

2. Algorithm. In this section, we present a BCU method for solving (1). Algorithm 1 summarizes the proposed method. In the algorithm, A_i denotes the ith block matrix of A corresponding to x_i, and

$$(5) \qquad f(x) = \frac{1}{2}x^\top Q x, \quad \hat{x}_j^{k,i} = x_j^{k+1} - w_{ij}(x_j^{k+1} - x_j^k) \quad \forall i, j,$$

where w_{ij} is the (i, j)th entry of W.

Algorithm 1 Hybrid Jacobian and Gauss–Seidel proximal BCU for (1).

Initialization: choose x^1 and set $\lambda^1 = 0$; choose β, ρ, W, and $P_i \succeq 0$, $i = 1, \ldots, m$.
for $k = 1, 2, \ldots$ **do**

    for $i = 1, 2, \ldots, m$ **do**
        Update x_i by

        (6)
    $x_i^{k+1} \in \arg\min_{x_i} \left\langle \nabla_i f(\hat{x}^{k,i}) - A_i^\top \left(\lambda^k - \beta(A\hat{x}^{k,i} - b)\right), x_i \right\rangle + g_i(x_i) + \frac{1}{2}\|x_i - x_i^k\|_{P_i}^2,$

    Update the multipliers by

    (7) $\lambda^{k+1} = \lambda^k - \rho(Ax^{k+1} - b),$

    if *a certain stopping condition is satisfied* **then**
        Return (x^{k+1}, λ^{k+1})

The algorithm is derived by applying a cyclic proximal BCU to the augmented Lagrangian function of (1), that is,

$$(8) \qquad \mathcal{L}_\beta(x, \lambda) = f(x) + \sum_{i=1}^{m} g_i(x_i) - \langle \lambda, Ax - b \rangle + \frac{\beta}{2}\|Ax - b\|^2.$$

At each iteration, we first renew every x_i by minimizing a proximal approximation of $\mathcal{L}_\beta$ one primal block variable at a time, while all the remaining ones are fixed, and we then update the multiplier λ by a dual gradient ascent step.

Note that in (6), for simplicity we evaluate the partial gradients of the quadratic term f and augmented term $\frac{\beta}{2}\|Ax - b\|^2$ at the same point $\hat{x}^{k,i}$. In general, two different points can be used to explore structures of Q and A, and our convergence analysis still holds by choosing two different W's appropriately. Practically, one can set

$$W = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \ddots & \ddots & \vdots \\ 1 & \cdots & 1 & 1 \end{bmatrix} \quad \text{or} \quad W = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 0 & 1 & \cdots & 1 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix},$$

which give fully Jacobian and fully Gauss–Seidel updates, respectively. Empirically, the latter one usually performs better. However, theoretically, the latter case may fail to converge when $m \geq 3$, as shown in [4, 12]. Therefore, we will design a mixing matrix W between the above two choices to theoretically guarantee convergence and also maintain practically nice performance. Our choice is inspired from the convergence analysis and will be discussed in detail in section 4.

We finish this section by giving examples showing that the fully Gauss–Seidel update method may diverge even if P_i's are large, which strongly motivate our hybrid strategy. Let us consider the problem

$$\min_x 0 \quad \text{s.t. } A_1 x_1 + A_2 x_2 + A_3 x_3 = 0,$$

where $A_1 = [1, 1 - \varepsilon, 1 - \varepsilon]^\top$, $A_2 = [1, 1, 1 - \varepsilon]^\top$, and $A_3 = [1, 1, 1]^\top$. For any $\varepsilon \neq 0$, A is invertible, and thus the problem has a unique solution $[0, 0, 0]^\top$. Applied to the

above problem with fully Gauss–Seidel update, $\beta = \rho = 1$, and $P_i = \frac{\max_j \|A_j\|^2}{\tau} \forall i$, Algorithm 1 becomes the following iterative method (see [12, section 3]):

$$
\begin{bmatrix} x_1^{k+1} \\ x_2^{k+1} \\ x_3^{k+1} \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0_{1\times 3} \\ \tau A_2^\top A_1 & 1 & 0 & 0_{1\times 3} \\ \tau A_3^\top A_1 & \tau A_3^\top A_2 & 1 & 0_{1\times 3} \\ A_1 & A_2 & A_3 & I_{3\times 3} \end{bmatrix}^{-1}
$$

$$
\times \begin{bmatrix} 1 - \tau A_1^\top A_1 & -\tau A_1^\top A_2 & -\tau A_1^\top A_3 & \tau A_1^\top \\ 0 & 1 - \tau A_2^\top A_2 & -\tau A_2^\top A_3 & \tau A_2^\top \\ 0 & 0 & 1 - \tau A_3^\top A_3 & \tau A_3^\top \\ 0_{3\times 1} & 0_{3\times 1} & 0_{3\times 1} & I_{3\times 3} \end{bmatrix} \begin{bmatrix} x_1^k \\ x_2^k \\ x_3^k \\ \lambda^k \end{bmatrix}.
$$

Denote M_τ as the iterating matrix. Then the algorithm converges if the spectral radius of M_τ is smaller than one and diverges if larger than one. For ε varying among $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$, we search for the largest τ with initial value $\frac{1}{3}$ and decrement 10^{-5} such that the spectral radius of M_τ is less than one. The results are listed in Table 1 below. They indicate that to guarantee the convergence of the algorithm, a diminishing stepsize is required for the x-update, but note that τ can be as large as $\frac{1}{3}$ for convergence if the fully Jacobian update is employed.

TABLE 1
Values of ε and the corresponding largest τ such that spectral radius of M_τ is less than one.

ε	10^{-1}	10^{-2}	10^{-3}	10^{-4}
τ	1.45473×10^{-1}	1.34433×10^{-2}	1.33333×10^{-3}	1.33333×10^{-4}

3. Convergence analysis. In this section, we analyze the convergence of Algorithm 1. We establish its global iterate sequence convergence and $O(1/t)$ rate by choosing an appropriate mixing matrix W and assuming merely weak convexity on the problem.

3.1. Notation and preliminary results. Before proceeding with our analysis, we introduce some notation and a few preliminary lemmas.

We let

$$
g(x) = \sum_{i=1}^m g_i(x_i), \quad F = f + g.
$$

A point x^* is a solution to (1) if there exists λ^* such that the KKT conditions hold:

(9a) $0 \in \partial F(x^*) - A^\top \lambda^*,$

(9b) $Ax^* - b = 0,$

where ∂F denotes the subdifferential of F. Together with the convexity of F, (9) implies

(10) $F(x) - F(x^*) - \langle \lambda^*, Ax - b \rangle \geq 0 \quad \forall x.$

We denote $\mathcal{X}^*$ as the solution set of (1). For any vector v and any symmetric matrix S of appropriate size, we define $\|v\|_S^2 = v^\top S v$. Note this definition does not require S to be PSD, so $\|v\|_S^2$ may be negative. I is reserved for the identity matrix and E for the all-ones matrix, whose size is clear from the context. $A \otimes B$ represents the

Kronecker product of two matrices A and B. For any matrices A, B, C, and D of appropriate sizes, it holds that (cf. [28, Chapter 4])

$$(A \otimes B)^\top = A^\top \otimes B^\top, \tag{11}$$

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD). \tag{12}$$

The following lemmas can be found in [54] and also appear in [14, 15].

LEMMA 3.1. *For any two vectors u, v and a symmetric matrix S, we have*

$$2u^\top S v = \|u\|_S^2 + \|v\|_S^2 - \|u - v\|_S^2. \tag{13}$$

LEMMA 3.2. *Given (possibly nonconvex) functions F, ϕ, and a fixed point $\bar{x}$, if for any λ, it holds that*

$$F(\bar{x}) - F(x^*) - \langle \lambda, A\bar{x} - b \rangle \le \phi(\lambda),$$

then for any $\gamma > 0$, we have

$$F(\bar{x}) - F(x^*) + \gamma \|A\bar{x} - b\| \le \sup_{\|\lambda\| \le \gamma} \phi(\lambda).$$

LEMMA 3.3. *Let (x^*, λ^*) be any point satisfying the condition in (10). If $F(\bar{x}) - F(x^*) + \gamma\|A\bar{x} - b\| \le \epsilon$ for certain $\epsilon \ge 0$ and $\gamma > \|\lambda^*\|$, then*

$$\|A\bar{x} - b\| \le \frac{\epsilon}{\gamma - \|\lambda^*\|} \ \text{ and } \ -\frac{\|\lambda^*\|\epsilon}{\gamma - \|\lambda^*\|} \le F(\bar{x}) - F(x^*) \le \epsilon.$$

3.2. Technical assumptions. Throughout our analysis, we make the following assumptions.

ASSUMPTION 1. *In (1), Q is PSD, and g_i's are proper closed convex functions. There exists a pair (x^*, λ^*) satisfying the KKT conditions in (9).*

ASSUMPTION 2. *The solution set of the subproblem in (6) is nonempty for any i.*

ASSUMPTION 3. *The mixing matrix W satisfies*

$$w_{ij} = 1 \quad \forall j \ge i, \tag{14}$$

$$\exists u \text{ such that } U = W - eu^\top \text{ is symmetric}, \tag{15}$$

where e is the all-ones vector.

The first assumption is minimal for finding a solution to (1). The second one is for well-definedness of the proposed algorithm, and it can be guaranteed if P_i's are all positive definite.

The requirements in (14) are for easy implementation of the update in (6) because otherwise x_i^{k+1} may implicitly depend on the later updated block variables. The conditions in (15) are assumed for technical reasons; see (18) below. They can be satisfied by first choosing u and then determining the corresponding W; see formula (33) below. How to choose u will be discussed in the next section since it is inspired from our convergence analysis.

3.3. Convergence results of Algorithm 1. We show that with appropriate proximal terms, Algorithm 1 can have $O(1/t)$ convergence rate, where t is the number of iterations. The result includes several existing ones as special cases, and we will discuss it after presenting our convergence result.

We first establish a few inequalities. Since Q is PSD, there exists a matrix H such that $Q = H^\top H$. Corresponding to the partition of x, we let $H = (H_1, \ldots, H_m)$.

PROPOSITION 3.4. *Let W satisfy the conditions in Assumption 3 and define*

$$y_i = H_i x_i, \quad z_i = A_i x_i \quad \forall i.$$

Then for any $\alpha > 0$,

$$
(16) \quad \sum_{i=1}^m \sum_{j=1}^m w_{ij} \langle H_i(x_i^{k+1} - x_i), H_j(x_j^{k+1} - x_j^k) \rangle
$$

$$
\leq \frac{1}{2} \left(\|y^{k+1} - y\|_V^2 - \|y^k - y\|_V^2 + \|y^{k+1} - y^k\|_V^2 \right)
$$

$$
+ \frac{1}{2\alpha} \|H(x^{k+1} - x)\|^2 + \frac{\alpha}{2} \left\| (u^\top \otimes I)(y^{k+1} - y^k) \right\|^2
$$

and

$$
(17) \quad \sum_{i=1}^m \sum_{j=1}^m w_{ij} \langle A_i(x_i^{k+1} - x_i), A_j(x_j^{k+1} - x_j^k) \rangle
$$

$$
\leq \frac{1}{2} \left(\|z^{k+1} - z\|_V^2 - \|z^k - z\|_V^2 + \|z^{k+1} - z^k\|_V^2 \right)
$$

$$
+ \frac{1}{2\alpha} \|A(x^{k+1} - b)\|^2 + \frac{\alpha}{2} \left\| (u^\top \otimes I)(z^{k+1} - z^k) \right\|^2,
$$

where $V = U \otimes I$.

Proof. We only show (16), and (17) follows in the same way. By the definition of y, we have

$$(18)$$
$$
\sum_{i=1}^m \sum_{j=1}^m w_{ij} \langle H_i(x_i^{k+1} - x_i), H_j(x_j^{k+1} - x_j^k) \rangle
$$
$$
= (y^{k+1} - y)^\top (W \otimes I)(y^{k+1} - y^k)
$$
$$
\overset{(15)}{=} (y^{k+1} - y)^\top (U \otimes I)(y^{k+1} - y^k) + (y^{k+1} - y)^\top \left((eu^\top) \otimes I \right)(y^{k+1} - y^k)
$$
$$
\overset{(12)}{=} (y^{k+1} - y)^\top (U \otimes I)(y^{k+1} - y^k) + (y^{k+1} - y)^\top \left((e \otimes I)(u^\top \otimes I) \right)(y^{k+1} - y^k)
$$
$$
= (y^{k+1} - y)^\top (U \otimes I)(y^{k+1} - y^k) + \left(H(x^{k+1} - x) \right)^\top (u^\top \otimes I)(y^{k+1} - y^k)
$$
$$
\leq (y^{k+1} - y)^\top (U \otimes I)(y^{k+1} - y^k) + \frac{1}{2\alpha} \|H(x^{k+1} - x)\|^2 + \frac{\alpha}{2} \left\| (u^\top \otimes I)(y^{k+1} - y^k) \right\|^2
$$
$$
\overset{(13)}{=} \frac{1}{2} \left(\|y^{k+1} - y\|_V^2 - \|y^k - y\|_V^2 + \|y^{k+1} - y^k\|_V^2 \right) + \frac{1}{2\alpha} \|H(x^{k+1} - x)\|^2
$$
$$
+ \frac{\alpha}{2} \left\| (u^\top \otimes I)(y^{k+1} - y^k) \right\|^2,
$$

where the inequality follows from the Cauchy–Schwarz inequality. $\square$

PROPOSITION 3.5 (one-iteration result).   *Let $\{(x^k, \lambda^k)\}_{k \geq 1}$ be the sequence generated from Algorithm 1. Under Assumptions 2 and 3, for any (x, λ) such that $Ax = b$, it holds that*

(19)

$$
\begin{aligned}
& F(x^{k+1}) - F(x) - \langle \lambda, Ax^{k+1} - b \rangle + \frac{1}{2\rho} \left(\|\lambda^{k+1} - \lambda\|^2 - \|\lambda^k - \lambda\|^2 \right) \\
& + \frac{\alpha - 1}{2\alpha} \|H(x^{k+1} - x)\|^2 \\
& + \left(\frac{\beta - \rho}{2} + \frac{\beta(\alpha - 1)}{2\alpha} \right) \|Ax^{k+1} - b\|^2 \\
& + \frac{1}{2} \left(\|x^{k+1} - x\|_P^2 - \|x^k - x\|_P^2 + \|x^{k+1} - x^k\|_P^2 \right) \\
& \leq \frac{1}{2} \left(\|y^{k+1} - y\|_V^2 - \|y^k - y\|_V^2 + \|y^{k+1} - y^k\|_V^2 \right) + \frac{\alpha}{2} \left\| (u^\top \otimes I)(y^{k+1} - y^k) \right\|^2 \\
& + \frac{\beta}{2} \left(\|z^{k+1} - z\|_V^2 - \|z^k - z\|_V^2 + \|z^{k+1} - z^k\|_V^2 \right) + \frac{\beta\alpha}{2} \left\| (u^\top \otimes I)(z^{k+1} - z^k) \right\|^2,
\end{aligned}
$$

where

$$
P = \mathrm{blkdiag}(P_1, \ldots, P_m).
$$

Proof. From the update (6), we have the following optimality conditions: for $i = 1, \ldots, m$,

$$
\begin{aligned}
0 =& \nabla_i f(\hat{x}^{k,i}) - A_i^\top \lambda^k + \tilde{\nabla} g_i(x_i^{k+1}) + \beta A_i^\top \left(A\hat{x}^{k,i} - b \right) + P_i(x_i^{k+1} - x_i^k) \\
=& H_i^\top \left(Hx^{k+1} - \sum_{j=1}^m w_{ij} H_j(x_j^{k+1} - x_j^k) \right) - A_i^\top \lambda^k + \tilde{\nabla} g_i(x_i^{k+1}) \\
(20) & \quad + \beta A_i^\top \left((Ax^{k+1} - b) - \sum_{j=1}^m w_{ij} A_j(x_j^{k+1} - x_j^k) \right) + P_i(x_i^{k+1} - x_i^k),
\end{aligned}
$$

where $\tilde{\nabla} g_i(x_i^{k+1})$ is a subgradient of g_i at x_i^{k+1}. Doing the inner product of both sides

of (20) with $x_i^{k+1} - x_i$, and summing them together over i, we have

(21)

$$\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\langle H_i(x_i^{k+1} - x_i), H_j(x_j^{k+1} - x_j^k)\rangle$$

$$+ \beta\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\langle A_i(x_i^{k+1} - x_i), A_j(x_j^{k+1} - x_j^k)\rangle$$

$$= \langle H(x^{k+1} - x), Hx^{k+1}\rangle + \langle A(x^{k+1} - x), -\lambda^k + \beta(Ax^{k+1} - b)\rangle$$

$$\quad + \langle x^{k+1} - x, \tilde{\nabla}g(x^{k+1})\rangle$$

$$\quad + \langle x^{k+1} - x, P(x^{k+1} - x^k)\rangle$$

$$\geq \langle H(x^{k+1} - x), Hx^{k+1}\rangle + \langle A(x^{k+1} - x), -\lambda^k + \beta(Ax^{k+1} - b)\rangle + g(x^{k+1}) - g(x)$$

$$\quad + \langle x^{k+1} - x, P(x^{k+1} - x^k)\rangle$$

$$= \frac{1}{2}\big(\|H(x^{k+1} - x)\|^2 - \|Hx\|^2 + \|Hx^{k+1}\|^2\big) - \langle\lambda^{k+1}, Ax^{k+1} - b\rangle$$

$$\quad + (\beta - \rho)\|Ax^{k+1} - b\|^2 + g(x^{k+1}) - g(x)$$

$$\quad + \frac{1}{2}\big(\|x^{k+1} - x\|_P^2 - \|x^k - x\|_P^2 + \|x^{k+1} - x^k\|_P^2\big)$$

$$= \frac{1}{2}\|H(x^{k+1} - x)\|^2 + F(x^{k+1}) - F(x) - \langle\lambda^{k+1}, Ax^{k+1} - b\rangle$$

$$\quad + (\beta - \rho)\|Ax^{k+1} - b\|^2 + \frac{1}{2}\big(\|x^{k+1} - x\|_P^2 - \|x^k - x\|_P^2 + \|x^{k+1} - x^k\|_P^2\big),$$

where the inequality uses the convexity of g, and in the second equality, we have used (13), the update rule (7), and the condition $Ax = b$.

Substituting (16) and (17) into (21), we have

(22)

$$F(x^{k+1}) - F(x) - \langle\lambda^{k+1}, Ax^{k+1} - b\rangle + \frac{\alpha - 1}{2\alpha}\|H(x^{k+1} - x)\|^2$$

$$+ \left(\frac{\beta}{2} + \frac{\beta(\alpha - 1)}{2\alpha} - \rho\right)\|Ax^{k+1} - b\|^2$$

$$+ \frac{1}{2}\big(\|x^{k+1} - x\|_P^2 - \|x^k - x\|_P^2 + \|x^{k+1} - x^k\|_P^2\big)$$

$$\leq \frac{1}{2}\big(\|y^{k+1} - y\|_V^2 - \|y^k - y\|_V^2 + \|y^{k+1} - y^k\|_V^2\big) + \frac{\alpha}{2}\big\|(u^\top \otimes I)(y^{k+1} - y^k)\big\|^2$$

$$+ \frac{\beta}{2}\big(\|z^{k+1} - z\|_V^2 - \|z^k - z\|_V^2 + \|z^{k+1} - z^k\|_V^2\big) + \frac{\beta\alpha}{2}\big\|(u^\top \otimes I)(z^{k+1} - z^k)\big\|^2.$$

From update (7), we have

$$0 = \langle\lambda^{k+1} - \lambda, Ax^{k+1} - b\rangle + \frac{1}{\rho}\langle\lambda^{k+1} - \lambda, \lambda^{k+1} - \lambda^k\rangle$$

$$\overset{(13)}{=} \langle\lambda^{k+1} - \lambda, Ax^{k+1} - b\rangle + \frac{1}{2\rho}\big(\|\lambda^{k+1} - \lambda\|^2 - \|\lambda^k - \lambda\|^2 + \|\lambda^{k+1} - \lambda^k\|^2\big)$$

(23) $$= \langle\lambda^{k+1} - \lambda, Ax^{k+1} - b\rangle + \frac{1}{2\rho}\big(\|\lambda^{k+1} - \lambda\|^2 - \|\lambda^k - \lambda\|^2\big) + \frac{\rho}{2}\|Ax^{k+1} - b\|^2.$$

Summing (22) and (23) together gives the desired result. □

Now we are ready to present our main result.

THEOREM 3.6. *Under Assumptions 1–3, let $\{(x^k, \lambda^k)\}_{k \geq 1}$ be the sequence generated from Algorithm 1 with parameters*

$$(24) \qquad \qquad \qquad \beta \geq \rho > 0,$$

$$(25) \quad P - D_H^\top \big((W - eu^\top + \alpha uu^\top) \otimes I\big) D_H - \beta D_A^\top \big((W - eu^\top + \alpha uu^\top) \otimes I\big) D_A := \hat{P} \succeq 0,$$

where $\alpha \geq 1$, and

$$D_H = \text{blkdiag}(H_1, \ldots, H_m), \quad D_A = \text{blkdiag}(A_1, \ldots, A_m).$$

Let $\bar{x}^{t+1} = \sum_{k=1}^t \frac{x^{k+1}}{t}$. Then

(26a)

$$\left| F(\bar{x}^{t+1}) - F(x^*) \right| \leq \frac{1}{2t} \left(\frac{\max\{(1 + \|\lambda^*\|)^2, 4\|\lambda^*\|^2\}}{\rho} + \|x^1 - x^*\|_P^2 - \|y^1 - y^*\|_V^2 - \beta\|z^1 - z^*\|_V^2 \right),$$

(26b)

$$\|A\bar{x}^{t+1} - b\| \leq \frac{1}{2t} \left(\frac{\max\{(1 + \|\lambda^*\|)^2, 4\|\lambda^*\|^2\}}{\rho} + \|x^1 - x^*\|_P^2 - \|y^1 - y^*\|_V^2 - \beta\|z^1 - z^*\|_V^2 \right),$$

where V is defined as in Proposition 3.4, and (x^*, λ^*) is any point satisfying the KKT conditions in (9).

In addition, if $\alpha > 1$ and $\hat{P} \succ 0$, then (x^k, λ^k) converges to a point $(x^\infty, \lambda^\infty)$ that satisfies the KKT conditions in (9).

Proof. Summing the inequality (19) from $k = 1$ through t and noting $\beta \geq \rho$, we have

$$\sum_{k=1}^t \left[F(x^{k+1}) - F(x) - \langle \lambda, Ax^{k+1} - b \rangle \right] + \frac{1}{2\rho}\|\lambda^{t+1} - \lambda\|^2 + \frac{1}{2}\|x^{t+1} - x\|_P^2$$

$$+ \sum_{k=1}^t \left(\frac{1}{2}\|x^{k+1} - x^k\|_P^2 + \frac{\alpha - 1}{2\alpha}\|H(x^{k+1} - x)\|^2 + \frac{\beta(\alpha - 1)}{2\alpha}\|Ax^{k+1} - b\|^2 \right)$$

$$(27) \quad \leq \frac{1}{2\rho}\|\lambda^1 - \lambda\|^2 + \frac{1}{2}\|x^1 - x\|_P^2 - \frac{1}{2}\|y^1 - y\|_V^2 - \frac{\beta}{2}\|z^1 - z\|_V^2$$

$$+ \frac{1}{2}\|y^{t+1} - y\|_V^2 + \frac{1}{2}\sum_{k=1}^t \|y^{k+1} - y^k\|_V^2 + \frac{\alpha}{2}\sum_{k=1}^t \left\|(u^\top \otimes I)(y^{k+1} - y^k)\right\|^2$$

$$+ \frac{\beta}{2}\|z^{t+1} - z\|_V^2 + \frac{\beta}{2}\sum_{k=1}^t \|z^{k+1} - z^k\|_V^2 + \frac{\beta\alpha}{2}\sum_{k=1}^t \left\|(u^\top \otimes I)(z^{k+1} - z^k)\right\|^2.$$

Note that

$$\|y^{k+1} - y^k\|_V^2 + \alpha \left\|(u^\top \otimes I)(y^{k+1} - y^k)\right\|^2$$
$$= (y^{k+1} - y^k)^\top (V + \alpha uu^\top \otimes I)(y^{k+1} - y^k)$$
$$= (x^{k+1} - x^k)^\top D_H^\top \big((W - eu^\top + \alpha uu^\top) \otimes I\big) D_H (x^{k+1} - x^k),$$

and similarly,

$$\|z^{k+1} - z^k\|_V^2 + \alpha \left\|(u^\top \otimes I)(z^{k+1} - z^k)\right\|^2$$
$$= (x^{k+1} - x^k)^\top D_A^\top \big((W - eu^\top + \alpha uu^\top) \otimes I\big) D_A (x^{k+1} - x^k).$$

Hence, by the choice of P in (25), we have from (27) that

$$\sum_{k=1}^{t} \left[F(x^{k+1}) - F(x) - \langle \lambda, Ax^{k+1} - b \rangle \right] + \frac{1}{2\rho} \|\lambda^{t+1} - \lambda\|^2$$

$$+ \sum_{k=1}^{t} \left(\frac{1}{2} \|x^{k+1} - x^k\|_{\hat{P}}^2 + \frac{\alpha - 1}{2\alpha} \|H(x^{k+1} - x)\|^2 + \frac{\beta(\alpha - 1)}{2\alpha} \|Ax^{k+1} - b\|^2 \right)$$

$$(28) \leq \frac{1}{2\rho} \|\lambda^1 - \lambda\|^2 + \frac{1}{2} \|x^1 - x\|_P^2 - \frac{1}{2} \|y^1 - y\|_V^2 - \frac{\beta}{2} \|z^1 - z\|_V^2.$$

Since $\alpha \geq 1$ and $\hat{P} \succeq 0$, it follows from the above inequality and the convexity of F that

$$(29) \qquad F(\bar{x}^{t+1}) - F(x^*) - \langle \lambda, A\bar{x}^{t+1} - b \rangle$$

$$\leq \frac{1}{2t} \left(\frac{1}{\rho} \|\lambda^1 - \lambda\|^2 + \|x^1 - x^*\|_P^2 - \|y^1 - y^*\|_V^2 - \beta \|z^1 - z^*\|_V^2 \right).$$

Since $\lambda^1 = 0$, we use Lemmas 3.2 and 3.3 with $\gamma = \max\{1 + \|\lambda^*\|, 2\|\lambda^*\|\}$ to obtain (26).

If $\alpha > 1$ and $\hat{P} \succ 0$, then letting $x = x^*, \lambda = \lambda^*$ in (28) and also using (10), we have

$$(30) \qquad \lim_{k \to \infty} (x^{k+1} - x^k) = 0, \quad \lim_{k \to \infty} (\lambda^{k+1} - \lambda^k) = -\rho \lim_{k \to \infty} (Ax^{k+1} - b) = 0,$$

and thus

$$(31) \qquad \lim_{k \to \infty} (x^k - \hat{x}^{k,i}) = 0 \quad \forall i.$$

On the other hand, letting $(x, \lambda) = (x^*, \lambda^*)$ in (19), using (10), and noting $\alpha, \alpha \geq 1$, we have

$$\frac{1}{2\rho} \|\lambda^{k+1} - \lambda^*\|^2 + \frac{1}{2} \left(\|x^{k+1} - x^*\|_P^2 - \|y^{k+1} - y^*\|_V^2 - \beta \|z^{k+1} - z^*\|_V^2 \right)$$

$$(32) \qquad \leq \frac{1}{2\rho} \|\lambda^k - \lambda^*\|^2 + \frac{1}{2} \left(\|x^k - x^*\|_P^2 - \|y^k - y^*\|_V^2 - \beta \|z^k - z^*\|_V^2 \right),$$

which, together with the choice of P, indicates the boundedness of $\{(x^k, \lambda^k)\}_{k \geq 1}$. Hence, it must have a finite cluster point $(x^\infty, \lambda^\infty)$, and there is a subsequence $\{(x^k, \lambda^k)\}_{k \in \mathcal{K}}$ convergent to this cluster point. From (30), it immediately follows that $Ax^\infty - b = 0$. In addition, letting $\mathcal{K} \ni k \to \infty$ in (6) and using (30) and (31) gives

$$x_i^\infty = \arg\min_{x_i} \langle \nabla_i f(x^\infty) - A_i^\top \lambda^\infty, x_i \rangle + g_i(x_i) + \frac{1}{2} \|x_i - x_i^\infty\|_{P_i}^2 \quad \forall i,$$

and thus the following optimality condition holds:

$$0 \in \nabla_i f(x^\infty) + \partial g_i(x_i^\infty) - A_i^\top \lambda^\infty \quad \forall i.$$

Therefore, $(x^\infty, \lambda^\infty)$ satisfies the conditions in (9). Since (32) holds for any point (x^*, λ^*) satisfying (9), it also holds with $(x^*, \lambda^*) = (x^\infty, \lambda^\infty)$. Denote

$$v = (x, \lambda), \qquad S = \begin{bmatrix} P - D_H^\top V D_H - \beta D_A^\top V D_A & 0 \\ 0 & \frac{I}{\rho} \end{bmatrix}.$$

Then letting $(x^*, \lambda^*) = (x^\infty, \lambda^\infty)$ in (32), we have $\|v^{k+1} - v^\infty\|_S \leq \|v^k - v^\infty\|_S$. From (25) and $\hat{P} \succ 0$, it follows that $S \succ 0$, and hence v^k gets closer to v^∞ as k increases. Because $(x^\infty, \lambda^\infty)$ is a cluster point of $\{(x^k, \lambda^k)\}_{k \geq 1}$, we obtain the convergence of (x^k, λ^k) to $(x^\infty, \lambda^\infty)$ and complete the proof. $\qquad\qquad\qquad\qquad\qquad\qquad\square$

4. How to choose a mixing matrix. In this section, we discuss how to choose W such that it satisfies Assumption 3. Note that the upper triangular part of W has been fixed, and we need only set its strictly lower triangular part. Denote $\mathcal{U}(W)$ and $\mathcal{L}(W)$, respectively, as the upper and strictly lower triangular parts of W, i.e., $W = \mathcal{L}(W) + \mathcal{U}(W)$, and thus (15) is equivalent to requiring the existence of u such that

$$\mathcal{L}(W) + \mathcal{U}(W) - eu^\top = \mathcal{L}(W)^\top + \mathcal{U}(W)^\top - ue^\top.$$

It suffices to let

$$(33)\qquad \mathcal{L}(W) = \mathcal{L}(W^\top + eu^\top - ue^\top) = \mathcal{L}(eu^\top - ue^\top + E).$$

Therefore, given any vector u, we can find a corresponding W by setting its upper triangular part to all ones and its strictly lower triangular part according to the above formula.

4.1. Finding u by solving an SDP. Theoretically, proximal terms help convergence guarantee of the algorithm. However, empirically, these terms can slow the convergence speed. Based on these observations, we aim at finding a block diagonal P such that (25) holds and also is as close to zero as possible.

One choice of P satisfying (25) could be

$$(34)\quad P_i = (1 - D_i)H_i^\top H_i + \beta(1 - D_i)A_i^\top A_i + d(\|H_i\|_2^2 + \beta\|A_i\|_2^2)I, \quad i = 1, \ldots, m,$$

where $D = \mathrm{diag}(D_1, \ldots, D_m)$ with each $D_i \in \{0, 1\}$ for each i, and

$$(35)\qquad dI \succeq (1 + \beta)(W - eu^\top + \alpha uu^\top) - (1 + \beta)(I - D).$$

Note that if $P_i = \eta_i I + H_i^\top H_i$, then (6) reduces to

$$(36)\quad x_i^{k+1} \in \arg\min_{x_i} f(x_i, \hat{x}_{\neq i}^{k,i}) - \langle A_i^\top(\lambda^k - \beta(A\hat{x}^{k,i} - b)), x_i\rangle + g_i(x_i) + \frac{\eta_i}{2}\|x_i - x_i^k\|^2.$$

Hence, $D_i = 0$ indicates no linearization to f or $\|Ax - b\|^2$ at x_i^k, and $D_i = 1$ indicates linearization to them. If (36) is easy to solve, one can set $D_i = 0$. Otherwise, $D_i = 1$ is recommended to have easier subproblems.

With D fixed, to obtain P according to (34), we need only specify the value of d. Recall that we aim at finding a close-to-zero P, so it would be desirable to choose u such that d is as small as possible. For simplicity, we set $\alpha = 1$. Therefore, to minimize d, we solve the following optimization problem:

$$(37)\qquad \min_{u,W} \lambda_{\max}\left(W - eu^\top + uu^\top - I + D\right) \quad \text{s.t. } W - eu^\top = W^\top - ue^\top,$$

where $\lambda_{\max}(B)$ denotes the maximum eigenvalue of a symmetric matrix B.

Using (33), we represent W by u and write (37) equivalently to

$$(38)\qquad \min_u \lambda_{\max}\left(\mathcal{L}(eu^\top - ue^\top) + E - eu^\top + uu^\top - I + D\right),$$

which can be further formulated as an SDP by the relation between the positive-definiteness of a 2×2 block matrix and its Schur complement (cf. [1, Appendix A.5.5]):

$$(39) \qquad \begin{bmatrix} A & B \\ B^\top & C \end{bmatrix} \succeq 0 \Leftrightarrow A - BC^{-1}B^\top \succeq 0,$$

where A is symmetric and $C \succ 0$. Let

$$\sigma = \lambda_{\max}\left(\mathcal{L}(eu^\top - ue^\top) + E - eu^\top + uu^\top - I + D\right).$$

Then

$$\sigma I \succeq \mathcal{L}(eu^\top - ue^\top) + E - eu^\top + uu^\top - I + D,$$

and by (39) it is equivalent to

$$\begin{bmatrix} (\sigma+1)I - D - \mathcal{L}(eu^\top - ue^\top) - E + eu^\top & u \\ u^\top & 1 \end{bmatrix} \succeq 0.$$

Therefore, (38) is equivalent to the SDP

$$(40) \qquad \min_{\sigma,u} \sigma \quad \text{s.t.} \quad \begin{bmatrix} (\sigma+1)I - D - \mathcal{L}(eu^\top - ue^\top) - E + eu^\top & u \\ u^\top & 1 \end{bmatrix} \succeq 0.$$

Note that problem (1) can be extremely large. However, the dimension (i.e., m) of the SDP (40) could be much smaller (see examples in section 5) and can be efficiently and accurately solved by the interior-point method. In addition, (40) does not depend on the data matrix H and A, so we can solve it offline.

If the sizes of H and A are not large, upon solving (40), one can explicitly form the matrix $D_H^\top\left((W - eu^\top + uu^\top) \otimes I\right)D_H + \beta D_A^\top\left((W - eu^\top + uu^\top) \otimes I\right)D_A$ and compute its spectral norm. This way, one can have a smaller P. However, for large-scale H or A, it can be overwhelmingly expensive to do so.

In addition, note that we can add more constraints to (40) to obtain a desired W. For instance, we can partition the m blocks into several groups. Then we update the blocks in the same group in parallel in Jacobian manner and cyclically renew the groups. This corresponds to fixing a few block matrices in the lower triangular part of W to all ones.

4.2. Special cases. A few special cases are as follows.

- If u is the zero vector, then by (33), we have the lower triangular part of W to be all ones, and this way gives a fully Jacobian BCU method. Hence, Theorem 3.6 applies for the Jacobian update method. In this case, if $D = I$ in (35), then we have $d \geq (1 + \beta)m$ that is significantly greater than the optimal value of (40).
- If we enforce the lower triangular part of W to all zeros, Algorithm 1 will reduce to a fully Gauss–Seidel BCU method. However, adding such constraints into (40) would lead to infeasibility. Hence, Theorem 3.6 does not apply to this case.
- If $m = 2$ and $D = 0$, i.e., if there were only two blocks and no linearization is performed, solving (40) would give the solution $\sigma = 0$ and $u = (0,1)^\top$. This way, we have

$$W = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \qquad P = \begin{bmatrix} H_1^\top H_1 + \beta A_1^\top A_1 & 0 \\ 0 & H_2^\top H_2 + \beta A_2^\top A_2 \end{bmatrix}$$

and thus recover the two-block ADMM with nonseparable quadratic term f in the objective. Theorem 3.6 implies an $O(1/t)$ convergence rate for this case, and it improves the result in [5], which shows convergence of this special case but without rate estimate.

4.3. Different mixing matrices. We can choose two different W's to explore the structures of A and Q. Let us give an example to illustrate this. Suppose A is a generic matrix and Q block tridiagonal. The W used to linearize the augmented term $\frac{\beta}{2}\|Ax - b\|^2$ can be set in the way discussed in section 4.1. For the mixing matrix to Q, we note $H_i^\top H_j = 0$ if $|i - j| > 1$. Then the left-hand side of (16) becomes $\sum_{|i-j|\leq 1} w_{ij}\langle H_i(x_i^{k+1} - x_i), H_j(x_j^{k+1} - x_j^k)\rangle$. Hence, following our analysis, we would require that there exist $\hat{u}$ such that $\hat{W} - e^\top \hat{u}$ is symmetric, where $\hat{w}_{ij} = 0$ if $|i-j| > 1$ and $\hat{w}_{ij} = 1$ if $i \leq j \leq i+1$. To completely determine $\hat{W}$, we need only set its values on the subdiagonal. Similarly to (37), we can find $\hat{u}$ and $\hat{w}_{i+1,i}$'s through solving

$$\min_{\hat{u},\hat{W}} \lambda_{\max}\left(\hat{W} - e\hat{u}^\top + \hat{u}\hat{u}^\top - I + D\right)$$

$$\text{s.t. } \hat{W} - e\hat{u}^\top = \hat{W}^\top - \hat{u}e^\top, \hat{w}_{ij} = 0 \quad \forall |i - j| > 1, \hat{w}_{ij} = 1, \ i \leq j \leq i+1.$$

The optimal value of the above problem is significantly smaller than that of (37), and thus in (34) the coefficient before $\|H_i\|^2$ can be set smaller. This way, we will have smaller P_i's, which potentially can make the algorithm converge faster.

5. Numerical experiments. In this section, we apply Algorithm 1 to three problems: quadratic programming, compressive principal component pursuit, and the MSVM problem. We test it with two different mixing matrices: the all-ones matrix and the matrix given by the method discussed in section 4.1. The former corresponds to a fully Jacobian update method and the latter to a hybrid Jacobian and Gauss–Seidel method, dubbed Jacobi-PC and JaGS-PC, respectively. We compare them to a recently proposed randomized proximal BCU method (named random-PC) in [15], the ADMM with Gauss back substitution (named as ADMM-GBS) in [21], and also the direct ADMM. Note that the direct ADMM is not guaranteed to converge for problems with more than two blocks, but empirically it can often perform well. ADMM-GBS is designed for separable multiblock convex problems. It does not allow linearization to the augmented term. In addition, it requires all A_i's to be full-column rank. Hence, the proposed algorithm is applicable to a broader class of problems, but we observe that JaGS-PC can be comparable to direct ADMM and ADMM-GBS.

We choose to compare with random-PC, direct ADMM, and ADMM-GBS because, as well as the proposed methods, all of them have low per-iteration complexity and low memory requirement and belong to the inexact ALM framework. On solving the three problems, one can also apply some other methods, such as the interior-point method and the projected subgradient method. The interior-point method can converge faster than the proposed ones in terms of iteration number. However, its per-iteration complexity is much higher, and thus total running time can be longer (that is observed for quadratic programming). In addition, it has a high demand on machine memory and may be inapplicable for large-scale problems, such as the compressive principal component pursuit. The projected subgradient method has per-iteration complexity similar to the proposed ones but converges much slower.

In all our tests, we report the results based on the actual iterate x^k, which is guaranteed to converge to an optimal solution. Although the convergence rate in Theorem 3.6 is based on the averaged point $\bar{x}^{t+1}$ (i.e., in ergodic sense), numerically

we notice that the convergence speed based on the iterate x^k is often faster than that based on the averaged point. This phenomenon also happens to the classic two-block ADMM. The work [24] shows that ADMM has an ergodic sublinear convergence rate, but all applications of ADMM still use the actual iterate as the solution.

5.1. Adaptive proximal terms. As we mentioned previously, the proximal terms used in (6) help the convergence guarantee but can empirically slow the convergence speed (see Figures 1 and 2). Here, we set P_i's similarly to (34) but in a simple adaptive way as follows:

$$(41) \quad P_i^k = (1 - D_i)H_i^\top H_i + \beta(1 - D_i)A_i^\top A_i + d^k(\|H_i\|_2^2 + \beta\|A_i\|_2^2)I, \quad i = 1, \ldots, m.$$

After each iteration k, we check whether the following inequality holds:

$$(42) \quad \eta\|x^{k+1} - x^k\|_{P^k}^2 \leq \|y^{k+1} - y^k\|_V^2 + \left\|(u^\top \otimes I)(y^{k+1} - y^k)\right\|^2 \\ + \beta\|z^{k+1} - z^k\|_V^2 + \beta\left\|(u^\top \otimes I)(z^{k+1} - z^k)\right\|^2,$$

and set

$$(43) \quad d^k = \begin{cases} \min\left(d^{k-1} + d_{\text{inc}}, d_{\max}\right) & \text{if (42) holds,} \\ d^{k-1} & \text{otherwise,} \end{cases}$$

where d_{inc} is a small positive number, and $\eta = 0.999$ is used[1] in all the tests. For stability and also efficiency, we choose d^1 and d_{inc} such that (42) does not happen many times. Specifically, we first run the algorithm to 20 iterations with (d^1, d_{inc}) selected from $\{0, 0.5, 1\} \times \{0.01, 0.1\}$. If there is one pair of values such that (42) does not always hold within the 20 iterations,[2] we accept that pair of (d^1, d_{inc}). Otherwise, we simply set $d^k = d_{\max}$ $\forall k$. For Jacobi-PC, we set $d_{\max} = \lambda_{\max}(E - I + D)$, and for JaGS-PC, we set $d_{\max}$ to the optimal value of (40), which is solved by SDPT3 [46] to high accuracy with stopping tolerance 10^{-12}. Note that as long as d_{inc} is positive, d^k can only be incremented finitely many times, and thus (42) can only happen in finitely many iterations. In addition, note that both sides of (42) can be evaluated as cheaply as computing $x^\top Qx$ and $x^\top A^\top Ax$.

The above adaptive way of setting P^k is inspired by our convergence analysis. If, after k_0 iterations, (42) never holds, then we can also show a sublinear convergence result by summing (19) from $k = k_0$ through t and then following the same arguments as those in the proof of Theorem 3.6.

5.2. Quadratic programming. We test Jacobi-PC and JaGS-PC on the non-negative linearly constrained quadratic programming

$$(44) \quad \min_x F(x) = \frac{1}{2}x^\top Qx + c^\top x \quad \text{s.t. } Ax = b, \, x \geq 0,$$

where $Q \in \mathbb{R}^{n \times n}$ is a symmetric PSD matrix, $A \in \mathbb{R}^{p \times n}$, and $b \in \mathbb{R}^p, c \in \mathbb{R}^n$. In the test, we set $p = 200, n = 2000$, and $Q = H^\top H$ with $H \in \mathbb{R}^{(n-10) \times n}$ generated

---

[1]In the proof of Theorem 3.6, we bound the y- and z-terms by the x-term. If the left-hand side of (42) with $\eta < 1$ can upper bound the right-hand side, then Theorem 3.6 guarantees the convergence of x^k to an optimal solution. Numerically, taking η close to 1 would make the algorithm more efficient.

[2]We notice that if (42) happens many times, the iterate may be far away from the optimal solution in the beginning, and that may affect the overall performance; see Figures 3 and 5.
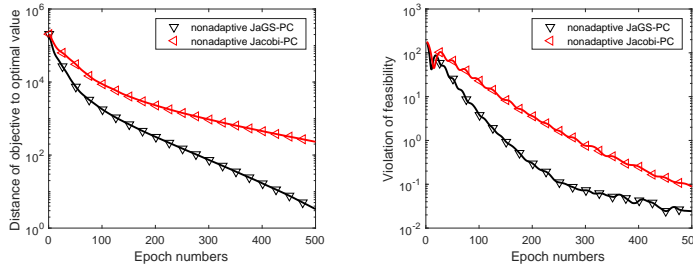
Fig. 1. *Results by applying Jacobi-PC and JaGS-PC without adapting proximal terms on the quadratic programming* (44). *Left: the distance of the objective to optimal value* $|F(x^k) - F(x^*)|$. *Right: the violation of feasibility* $\|Ax^k - b\|$.

according to the standard Gaussian distribution. This generated Q is degenerate, and thus the problem is only weakly convex. The entries of c follow i.i.d. standard Gaussian distribution, and those of b follow from uniform distribution on $[0, 1]$. We set $A = [B, I]$ to guarantee feasibility of the problem, with B generated according to standard Gaussian distribution.

We evenly partition the variable x into $m = 40$ blocks, each one consisting of 50 coordinates. The same values of parameters are set for both Jacobi-PC and JaGS-PC as follows:

$$\beta = \rho = 1, \ D = I, \ d^1 = 0.5, \ d_{\text{inc}} = 0.1.$$

They are compared to random-PC that uses the same penalty parameter $\beta = 1$ and $\rho = \frac{1}{m}$ according to the analysis in [15]. We also adaptively increase the proximal parameter of random-PC in a way similar to that in section 5.1. All three methods run to 500 epochs, where each epoch is equivalent to updating all blocks one time. Their per-epoch complexity is almost the same. To evaluate their performance, we compute the distance of the objective to optimal value $|F(x^k) - F(x^*)|$ and the violation of feasibility $\|Ax^k - b\|$ at each iteration k, where the optimal solution is obtained by MATLAB solver `quadprog` with the "interior-point-convex" option. Figures 1 and 2 plot the results by the three methods in terms of both iteration number and running time (sec). In Figure 1, we simply set $d^k = d_{\max}$ for Jacobi-PC and JaGS-PC, i.e., without adapting the proximal terms, where in this test, $d_{\max} = 18.3273$ for JaGS-PC and $d_{\max} = 40$ for Jacobi-PC. From the figures, we see that JaGS-PC is significantly faster than Jacobi-PC in terms of both objective and feasibility for both adaptive and nonadaptive cases, and random-PC is slightly slower than JaGS-PC.

5.3. Compressive principal component pursuit. In this subsection, we test Jacobi-PC and JaGS-PC on

$$(45) \qquad \min_{X,Y} F(X, Y) = \mu\|X\|_1 + \|Y\|_* \text{ s.t. } \mathcal{A}(X + Y) = b,$$

where $\|X\|_1 = \sum_{i,j} |X_{ij}|$, $\|Y\|_*$ denotes the matrix operator norm and equals the largest singular value of Y, $\mathcal{A}$ is a linear operator, and b contains the measurements. If $\mathcal{A}$ is the identity operator, (45) is called the principal component pursuit proposed in [3], and it is called compressive PCP [52] when $\mathcal{A}$ is an underdetermined measuring operator. We consider the sampling operator, i.e., $\mathcal{A} = \mathcal{P}_\Omega$, where Ω is an index set and $\mathcal{P}_\Omega$ is a projection keeping the entries in Ω and zeroing out all others.

Assume M to be the underlying matrix and $b = \mathcal{A}(M)$. Upon solving (45), $X + Y$ recovers M with sparse part X and low-rank part Y. We use the Escalator
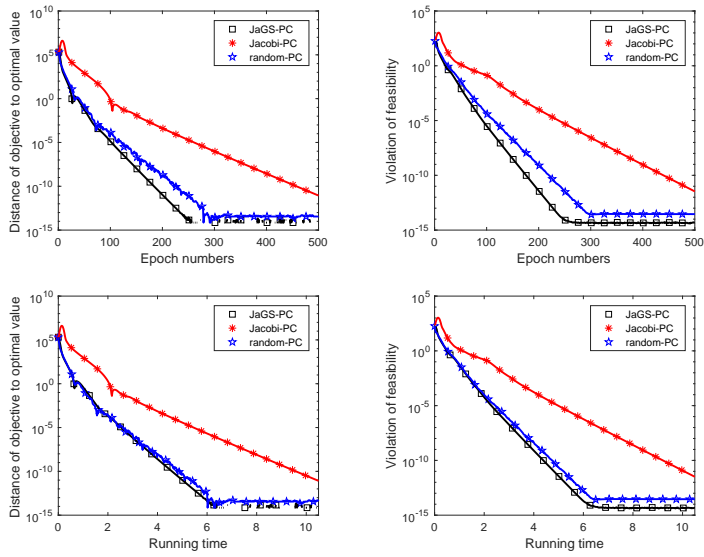
FIG. 2. *Results of applying three different proximal BCU methods: Jacobi-PC, JaGS-PC, and random-PC on the quadratic programming* (44). *Left: the distance of the objective to optimal value* $|F(x^k) - F(x^*)|$. *Right: the violation of feasibility* $\|Ax^k - b\|$. *The running time of JaGS-PC includes that for finding the mixing matrix* W.

video dataset,[3] which has 200 frames of 130×160 images. Each 2D image frame is reshaped into a column vector to form a slim and tall matrix M with 200 columns. For this data, X and Y will encode the foreground and the background of the video, respectively. We generate the index set Ω uniformly at random, and 30% samples are selected from each image frame.

Introducing another variable Z, we write (45) equivalently to

$$(46) \qquad \min_{X,Y,Z} \mu\|X\|_1 + \|Y\|_* \quad \text{s.t. } X + Y = Z, \ \mathcal{A}(Z) = b,$$

which naturally has three block variables. Applying Algorithm 1 to (46) with adaptive P_i's as in (41) and $D = 0$ and noting $\|I + \mathcal{A}^\top \mathcal{A}\|_2 = 2$, we iteratively perform the

---

[3]Available from http://pages.cs.wisc.edu/~jiaxu/projects/gosus/supplement/.

updates:

(47a)
$$X^{k+1} = \arg\min_X \mu\|X\|_1 - \langle \Lambda^k, X \rangle + \frac{\beta}{2}\|X + Y^k - Z^k\|_F^2 + \frac{d^k\beta}{2}\|X - X^k\|^2,$$

(47b)
$$Y^{k+1} = \arg\min_Y \|Y\|_* - \langle \Lambda^k, Y \rangle + \frac{\beta}{2}\|\tilde{X}^k + Y - Z^k\|_F^2 + \frac{d^k\beta}{2}\|Y - Y^k\|^2,$$

(47c)
$$Z^{k+1} = \arg\min_Z \langle \Lambda^k - \mathcal{A}^\top(\Pi^k), Z \rangle + \frac{\beta}{2}\|\tilde{X}^k + \tilde{Y}^k - Z^k\|_F^2 + \frac{\beta}{2}\|\mathcal{A}(Z) - b\|^2 + d^k\beta\|Z - Z^k\|^2,$$

(47d)
$$\Lambda^{k+1} = \Lambda^k - \rho(X^{k+1} + Y^{k+1} - Z^{k+1}),$$

(47e)
$$\Pi^{k+1} = \Pi^k - \rho(\mathcal{A}(Z^{k+1}) - b).$$

Since $\mathcal{A}\mathcal{A}^\top = I$, all three primal subproblems have closed-form solutions. We set $\beta = \rho = 0.05$ and $d_{\text{inc}} = 0.01$ for both JaGS-PC and Jacobi-PC methods, and set $d^1 = 0$ for JaGS-PC and $d^1 = 1$ for Jacobi-PC because the latter can deviate from optimality very far away in the beginning if it starts with a small d^1 (see Figure 3). They are compared to random-PC, direct ADMM, and ADMM-GBS. At every iteration, random-PC performs one update among (47a)—(47c) with $d^k = 0$ and then updates Λ and Π by (47d) and (47e) with $\rho = \frac{\beta}{3}$; the direct ADMM sets $d^k = 0 \,\forall k$ in (47); and ADMM-GBS runs the direct ADMM first and then performs a correction step by Gauss back substitution. We use the same β and ρ for the direct ADMM and ADMM-GBS and set the correction step parameter of ADMM-GBS to 0.99. On solving the SDP (40), we have for JaGS-PC $d_{\max} = 0.4270$ and the mixing matrix,

$$W = \begin{bmatrix} 1 & 1 & 1 \\ 0.3691 & 1 & 1 \\ -0.2618 & 0.3691 & 1 \end{bmatrix}.$$

Figure 4 plots the results by all five methods, where the optimal solution is obtained by running JaGS-PC to 10,000 epochs. From the figure, we see that JaGS-PC performs significantly better than Jacobi-PC. and JaGS-PC; direct ADMM and ADMM-GBS perform almost the same; and random-PC performs the worst. Note that although direct ADMM works well on this example, its convergence is not guaranteed in general.

5.4. Multiclass support vector machine. In this subsection, we test Jacobi-PC, JaGS-PC, and random-PC on the MSVM problem that is considered in [55]:

$$(48) \qquad \min_X F(X) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} \iota_{\cdot \neq j}(b_i)[x_j^\top a_i + 1]_+ + \mu\|X\|_1 \quad \text{s.t. } Xe = 0,$$

where x_j is the jth column of X, $\{(a_i, b_i)\}_{i=1}^n$ is the training dataset with label $b_i \in \{1, \ldots, c\} \,\forall i$, $\iota_{\cdot \neq j}(b_i)$ equals one if $b_i \neq j$ and zero otherwise, and $[d]_+ = \max(0, d)$. We set the number of classes to $c = 3$ and randomly generate the data according to Gaussian distribution $\mathcal{N}(v_j, \Sigma_j)$ for the jth class, where $v_j \in \mathbb{R}^p$ and $\Sigma_j \in \mathbb{R}^{p \times p}$ for
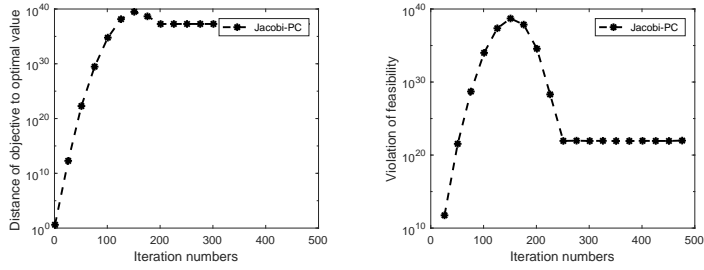
Fig. 3. *Results of Jacobi-PC with $d^1 = 0$ and $d_{\mathrm{inc}} = 0.01$ for solving the compressive PCP (46) on the Escalator dataset. Left: the relative error between the objective and optimal value $\frac{|F(X^k,Y^k)-F(X^*,Y^*)|}{F(X^*,Y^*)}$. Right: relative violation of feasibility: $\frac{\|X^k+Y^k-Z^k\|_F+\|\mathcal{A}(Z^k)-b\|_F}{\|M\|_F}$.*

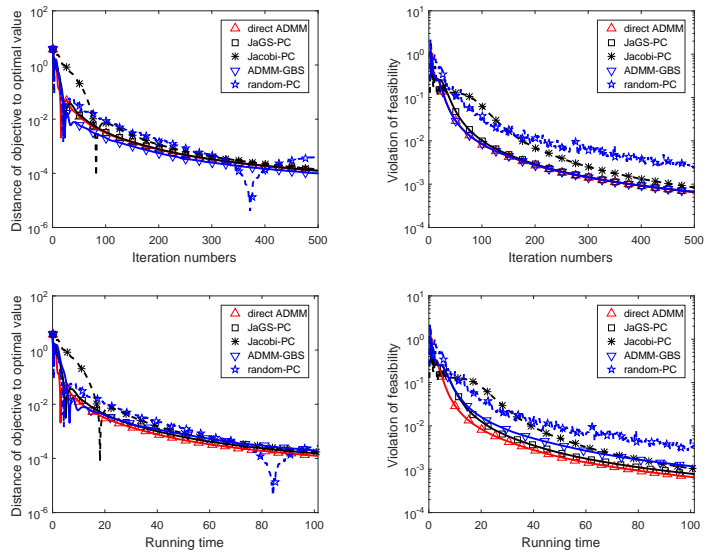Fig. 4. *Results by five different methods for solving the compressive PCP (46) on the Escalator dataset. Left: the relative error between the objective and optimal value $\frac{|F(X^k,Y^k)-F(X^*,Y^*)|}{F(X^*,Y^*)}$. Right: relative violation of feasibility: $\frac{\|X^k+Y^k-Z^k\|_F+\|\mathcal{A}(Z^k)-b\|_F}{\|M\|_F}$. The running time of JaGS-PC includes that for finding the mixing matrix W.*

$j = 1, 2, 3$ are

$$v_1 = \begin{bmatrix} E_{s\times 1} \\ 0 \end{bmatrix}, \; v_2 = \begin{bmatrix} 0_{s/2\times 1} \\ E_{s\times 1} \\ 0 \end{bmatrix}, \; v_3 = \begin{bmatrix} 0_{s\times 1} \\ E_{s\times 1} \\ 0 \end{bmatrix}, \; \Sigma_1 = \begin{bmatrix} \sigma E_{s\times s} + (1-\sigma)I & 0 \\ 0 & I \end{bmatrix},$$

$$\Sigma_2 = \begin{bmatrix} I_{\frac{s}{2}\times\frac{s}{2}} & 0 & 0 \\ 0 & \sigma E_{s\times s} + (1-\sigma)I & 0 \\ 0 & 0 & I \end{bmatrix}, \; \Sigma_3 = \begin{bmatrix} I_{s\times s} & 0 & 0 \\ 0 & \sigma E_{s\times s} + (1-\sigma)I & 0 \\ 0 & 0 & I \end{bmatrix},$$

where $E, I, 0$, respectively, represent all-ones, identity, and all-zero matrices of appropriate sizes, and the subscript specifies the size. The parameter σ measures correlation of features. This kind of dataset has also been used in [56] for testing binary SVM. In
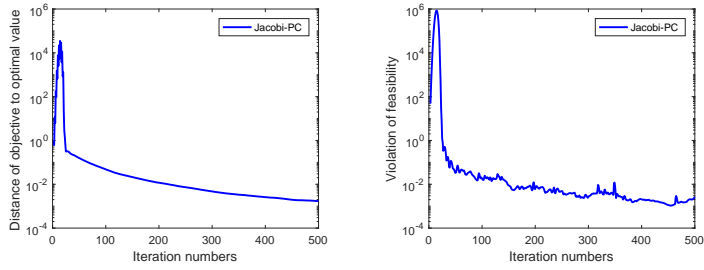
FIG. 5. *Results by Jacobi-PC with $d^1 = 0.5$ and $d_{\mathrm{inc}} = 0.1$ for solving the MSV problem* (48) *on randomly generated data. Left: distance of the objective to optimal value $|F(X^k) - F(X^*)|$. Right: violation of feasibility $\|X^k e\|$.*

the test, we set $\sigma = 0.1, \mu = 0.001, p = 200$, and $n = 300$, with each class consisting of 100 samples.

Letting $A = [a_1, \ldots, a_n]$ and $Y = A^\top X + 1$, we write (48) equivalently to

$$(49) \qquad \min_{X,Y} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} \iota_{\cdot \neq j}(b_i)[Y_{ij}]_+ + \mu\|X\|_1 \quad \text{s.t. } A^\top X - Y + 1 = 0, \ Xe = 0.$$

To apply Algorithm 1 to the above model, we partition the variable into four blocks (x_1, x_2, x_3, Y). Linearization to the augmented term is employed, i.e., $D = I$ in (41). The parameters are set to $\beta = \rho = 0.005$ and $d_{\mathrm{inc}} = 0.1$ for both Jacobi-PC and JaGS-PC, and to $d^1 = 0.5$ for JaGS-PC and $d^1 = 1$ for Jacobi-PC because again the latter can deviate from optimality far away in the beginning if it starts with a small d^1 (see Figure 5). At each iteration, random-PC picks one block from x_1, x_2, x_3 and Y uniformly at random and updates it by minimizing the proximal linearized augmented Lagrangian function with respect to the selected block and with the other three blocks fixed. The proximal parameter is adaptively increased as well. In the multiplier update, $\rho = \frac{\beta}{4}$ is set, and β is the same as that for JaGS-PC. On solving the SDP (40), we have for JaGS-PC $d_{\max} = 1.8711$ and the mixing matrix,

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.5353 & 1 & 1 & 1 \\ 0.0705 & 0.5353 & 1 & 1 \\ -0.3942 & 0.0705 & 0.5353 & 1 \end{bmatrix}.$$

We plot the results in Figure 6, where the optimal solution is given by CVX [18] with a "high precision" option. In terms of objective value, JaGS-PC and random-PC perform significantly better than Jacobi-PC, and the former two perform comparably well. However, random-PC is significantly worse than JaGS-PC and Jacobi-PC in terms of feasibility violation.

6. Conclusions. We have proposed a hybrid Jacobian and Gauss–Seidel block coordinate update (BCU) method for solving linearly constrained convex programming. The method performs each primal block variable update by minimizing a function that approximates the augmented Lagrangian at affinely combined points of the previous two iterates. We have presented a way to choose the mixing matrix with desired properties. Global iterate sequence convergence and sublinear rate results of the hybrid method have been established. In addition, numerical experiments have been performed to demonstrate its efficiency.
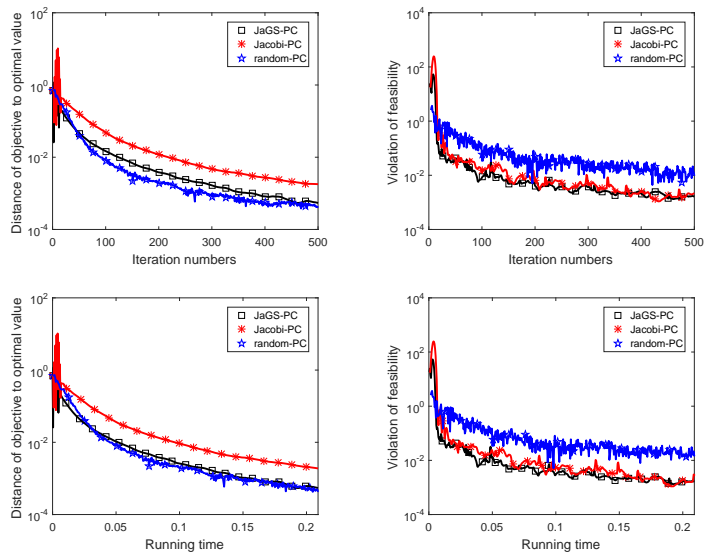
FIG. 6. *Results of Jacobi-PC and JaGS-PC for solving the MSVM problem* (48) *on randomly generated data. Left: distance of the objective to optimal value* $|F(X^k) - F(X^*)|$. *Right: violation of feasibility* $\|X^k e\|$. *The running time of JaGS-PC includes that for finding the mixing matrix* W.

REFERENCES

[1] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[2] X. Cai, D. Han, and X. Yuan, *The Direct Extension of ADMM for Three-Block Separable Convex Minimization Models Is Convergent When One Function is Strongly Convex*, e-print, Optimization Online, 2014.

[3] E. J. Candès, X. Li, Y. Ma, and J. Wright, *Robust principal component analysis?*, J. ACM, 58 (2011), 11.

[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. Programming, 155 (2016), pp. 57–79.

[5] C. Chen, M. Li, X. Liu, and Y. Ye, *Extended ADMM and BCD for Nonseparable Convex Minimization Models with Quadratic Coupling Terms: Convergence Analysis and Insights*, preprint, https://arxiv.org/abs/1508.00193, 2015.

[6] C. Chen, Y. Shen, and Y. You, *On the convergence analysis of the alternating direction method of multipliers with three blocks*, Abstract Appl. Anal., 2013, 183961.

[7] P. L. Combettes and J.-C. Pesquet, *Stochastic quasi-Fejér block-coordinate fixed point iterations with random sweeping*, SIAM J. Optim., 25 (2015), pp. 1221–1248, https://doi.org/10.1137/140971233.

[8] Y. Cui, X. Li, D. Sun, and K.-C. Toh, *On the Convergence Properties of a Majorized ADMM for Linearly Constrained Convex Optimization Problems with Coupled Objective Functions*, preprint, https://arxiv.org/abs/1502.00098, 2015.

[9] C. Dang and G. Lan, *Randomized Methods for Saddle Point Computation*, preprint, https://arxiv.org/abs/1409.8625, 2014.

[10] D. Davis and W. Yin, *A Three-Operator Splitting Scheme and Its Optimization Applications*, preprint, https://arxiv.org/abs/1504.01032, 2015.

[11] W. Deng, M.-J. Lai, Z. Peng, and W. Yin, *Parallel multi-block ADMM with* $o(1/k)$ *convergence*, J. Sci. Comput., 71 (2017), pp. 712–736.

[12] J.-K. Feng, H.-B. Zhang, C.-Z. Cheng, and H.-M. Pei, *Convergence analysis of l-ADMM for multi-block linear-constrained separable convex minimization problem*, J. Oper. Res. Soc. China, 3 (2015), pp. 563–579.

[13] 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.

[14] X. Gao, B. Jiang, and S. Zhang, *On the Information-Adaptive Variants of the ADMM: An Iteration Complexity Perspective*, e-print, Optimization Online, 2014.

[15] X. Gao, Y. Xu, and S. Zhang, *Randomized Primal-Dual Proximal Block Coordinate Updates*, preprint, https://arxiv.org/abs/1605.05969, 2016.

[16] X. Gao and S.-Z. Zhang, *First-order algorithms for convex optimization with nonseparable objective and coupled constraints*, J. Oper. Res. Soc. China, 5 (2017), pp. 131–159.

[17] R. Glowinski and A. Marrocco, *Sur l'approximation, par eléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires*, ESAIM Math. Model. Numer. Anal., 9 (1975), pp. 41–76.

[18] M. Grant, S. Boyd, and Y. Ye, *CVX: MATLAB Software for Disciplined Convex Programming*, Version 2.1, 2017, http://cvxr.com/cvx/ (accessed 2-5-18).

[19] D. Han and X. Yuan, *A note on the alternating direction method of multipliers*, J. Optim. Theory Appl., 155 (2012), pp. 227–238.

[20] B. He, L. Hou, and X. Yuan, *On full Jacobian decomposition of the augmented Lagrangian method for separable convex programming*, SIAM J. Optim., 25 (2015), pp. 2274–2312, https://doi.org/10.1137/130922793.

[21] 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, https://doi.org/10.1137/110822347.

[22] B. He, M. Tao, and X. Yuan, *Convergence rate analysis for the alternating direction method of multipliers with a substitution procedure for separable convex programming*, Math. Oper. Res., 42 (2017), pp. 662–691.

[23] B. He, H.-K. Xu, and X. Yuan, *On the proximal Jacobian decomposition of ALM for multiple-block separable convex minimization problems and its relationship to ADMM*, J. Sci. Comput., 66 (2016), pp. 1204–1217.

[24] B. He and X. Yuan, *On the $O(1/n)$ convergence rate of the Douglas–Rachford alternating direction method*, SIAM J. Numer. Anal., 50 (2012), pp. 700–709, https://doi.org/10.1137/110836936.

[25] C. Hildreth, *A quadratic programming procedure*, Naval Res. Logist. Quart., 4 (1957), pp. 79–85.

[26] M. Hong, T.-H. Chang, X. Wang, M. Razaviyayn, S. Ma, and Z.-Q. Luo, *A Block Successive Upper Bound Minimization Method of Multipliers for Linearly Constrained Convex Optimization*, preprint, https://arxiv.org/abs/1401.7079, 2014.

[27] M. Hong, X. Wang, M. Razaviyayn, and Z.-Q. Luo, *Iteration complexity analysis of block coordinate descent methods*, Math. Programming, 163 (2016), pp. 85–114.

[28] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, New York, 1991.

[29] G. M. James, C. Paulson, and P. Rusmevichientong, *The Constrained Lasso*, Technical report, 2012.

[30] M. Li, D. Sun, and K.-C. Toh, *A convergent 3-block semi-proximal ADMM for convex minimization problems with one strongly convex block*, Asia-Pacific J. Oper. Res., 32 (2015), 1550024.

[31] X. Li, D. Sun, and K.-C. Toh, *A Schur complement based semi-proximal ADMM for convex quadratic conic programming and extensions*, Math. Programming, 155 (2016), pp. 333–373.

[32] T. Lin, S. Ma, and S. Zhang, *On the global linear convergence of the ADMM with multiblock variables*, SIAM J. Optim., 25 (2015), pp. 1478–1497, https://doi.org/10.1137/140971178.

[33] T. Lin, S. Ma, and S. Zhang, *On the sublinear convergence rate of multi-block ADMM*, J. Oper. Res. Soc. China, 3 (2015), pp. 251–274.

[34] Y.-F. Liu, X. Liu, and S. Ma, *On the Non-ergodic Convergence Rate of an Inexact Augmented Lagrangian Framework for Composite Convex Programming*, preprint, http://arxiv.org/abs/1603.05738, 2016.

[35] R. D. C. Monteiro and B. F. Svaiter, *Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers*, SIAM J. Optim., 23 (2013), pp. 475–507, https://doi.org/10.1137/110849468.

[36] I. Necoara and A. Patrascu, *A random coordinate descent algorithm for optimization problems with composite objective function and linear coupled constraints*, Comput. Optim. Appl., 57 (2014), pp. 307–337.

[37] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, 2006.

[38] Z. Peng, T. Wu, Y. Xu, M. Yan, and W. Yin, *Coordinate friendly structures, algorithms and applications*, Ann. Math. Sci. Appl., 1 (2016), pp. 57–119.

[39] Z. Peng, Y. Xu, M. Yan, and W. Yin, *ARock: An algorithmic framework for asynchronous parallel coordinate updates*, SIAM J. Sci. Comput., 38 (2016), pp. A2851–A2879, https://doi.org/10.1137/15M1024950.

[40] J.-C. Pesquet and A. Repetti, *A Class of Randomized Primal-Dual Algorithms for Distributed Optimization*, preprint, https://arxiv.org/abs/1406.6404, 2014.

[41] M. Razaviyayn, M. Hong, and Z.-Q. Luo, *A unified convergence analysis of block successive minimization methods for nonsmooth optimization*, SIAM J. Optim., 23 (2013), pp. 1126–1153, https://doi.org/10.1137/120891009.

[42] H.-J. M. Shi, S. Tu, Y. Xu, and W. Yin, *A Primer on Coordinate Descent Algorithms*, preprint, https://arxiv.org/abs/1610.00040, 2016.

[43] D. Sun, K.-C. Toh, and L. Yang, *A convergent 3-block semiproximal alternating direction method of multipliers for conic programming with 4-type constraints*, SIAM J. Optim., 25 (2015), pp. 882–915, https://doi.org/10.1137/140964357.

[44] R. Sun, Z.-Q. Luo, and Y. Ye, *On the Expected Convergence of Randomly Permuted ADMM*, preprint, https://arxiv.org/abs/1503.06387, 2015.

[45] R. Tibshirani, *Regression shrinkage and selection via the lasso*, J. Roy. Statist. Soc. Ser. B Methodol., 58 (1996), pp. 267–288.

[46] K.-C. Toh, M. J. Todd, and R. H. Tütüncü, *SDPT3—a MATLAB software package for semidefinite programming, version* 1.3, Optim. Methods Software, 11 (1999), pp. 545–581.

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

[48] P. Tseng and S. Yun, *Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization*, J. Optim. Theory Appl., 140 (2009), pp. 513–535.

[49] P. Tseng and S. Yun, *A coordinate gradient descent method for nonsmooth separable minimization*, Math. Programming, 117 (2009), pp. 387–423.

[50] Y. Wang, W. Yin, and J. Zeng, *Global Convergence of ADMM in Nonconvex Nonsmooth Optimization*, preprint, https://arxiv.org/abs/1511.06324, 2015.

[51] J. Warga, *Minimizing certain convex functions*, J. Soc. Indust. Appl. Math., 11 (1963), pp. 588–593, https://doi.org/10.1137/0111043.

[52] J. Wright, A. Ganesh, K. Min, and Y. Ma, *Compressive principal component pursuit*, Inf. Inference, 2 (2013), pp. 32–68.

[53] S. J. Wright, *Coordinate descent algorithms*, Math. Programming, 151 (2015), pp. 3–34.

[54] Y. Xu, *Accelerated first-order primal-dual proximal methods for linearly constrained composite convex programming*, SIAM J. Optim., 27 (2017), pp. 1459–1484, https://doi.org/10.1137/16M1082305.

[55] Y. Xu, I. Akrotirianakis, and A. Chakraborty, *Alternating direction method of multipliers for regularized multiclass support vector machines*, in Proc. International Workshop on Machine Learning, Optimization and Big Data, Springer, 2015, pp. 105–117.

[56] Y. Xu, I. Akrotirianakis, and A. Chakraborty, *Proximal gradient method for Huberized support vector machine*, Pattern Anal. Appl., 19 (2015), pp. 989–1005.

[57] Y. Xu and W. Yin, *A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion,* SIAM J. Imaging Sci., 6 (2013), pp. 1758–1789, https://doi.org/10.1137/120887795.

[58] Y. Xu and W. Yin, *A globally convergent algorithm for nonconvex optimization based on block coordinate update*, J. Sci. Comput., 72 (2017), pp. 700–734.