Computational Optimization and Applications (2019) 72:87–113
https://doi.org/10.1007/s10589-018-0037-8

Asynchronous parallel primal–dual block coordinate update methods for affinely constrained convex programs

Yangyang Xu[1]

Received: 17 February 2018 / Published online: 27 September 2018

Abstract
Recent several years have witnessed the surge of asynchronous (async-) parallel computing methods due to the extremely big data involved in many modern applications and also the advancement of multi-core machines and computer clusters. In optimization, most works about async-parallel methods are on unconstrained problems or those with block separable constraints. In this paper, we propose an async-parallel method based on block coordinate update (BCU) for solving convex problems with *nonseparable* linear constraint. Running on a single node, the method becomes a novel randomized primal–dual BCU for multi-block affinely constrained problems. For these problems, Gauss–Seidel cyclic primal–dual BCU is not guaranteed to converge to an optimal solution if no additional assumptions, such as strong convexity, are made. On the contrary, assuming convexity and existence of a primal–dual solution, we show that the objective value sequence generated by the proposed algorithm converges *in probability* to the optimal value and also the constraint residual to zero. In addition, we establish an ergodic $O(1/k)$ convergence result, where k is the number of iterations. Numerical experiments are performed to demonstrate the efficiency of the proposed method and significantly better speed-up performance than its sync-parallel counterpart.

Keywords Asynchronous parallel · Block coordinate update · Primal–dual method

Mathematics Subject Classification 90C06 · 90C25 · 68W40 · 49M27

This work is partly supported by NSF Grant DMS-1719549.

✉ Yangyang Xu
xuy21@rpi.edu

1 Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY, USA

1 Introduction

Modern applications in various data sciences and engineering can involve huge amount of data and/or variables [43]. Driven by these very large-scale problems and also the advancement of multi-core computers, parallel computing has gained tremendous attention in recent years. In this paper, we consider the affinely constrained multi-block structured problem:

$$\min_{\mathbf{x}} f(\mathbf{x}_1, \ldots, \mathbf{x}_m) + \sum_{i=1}^{m} g_i(\mathbf{x}_i), \text{ s.t. } \sum_{i=1}^{m} \mathbf{A}_i \mathbf{x}_i = \mathbf{b}, \tag{1}$$

where the variable $\mathbf{x}$ is partitioned into multiple disjoint blocks $\mathbf{x}_1, \ldots, \mathbf{x}_m$, f is a continuously differentiable and convex function, and each g_i is a lower semi-continuous extended-valued convex but possibly non-differentiable function. Besides the non-separable affine constraint, (1) can also include certain block separable constraint by letting part of g_i be an indicator function of a convex set, e.g., nonnegativity constraint.

We will present a novel asynchronous (async-) parallel primal–dual method (see Algorithm 2) towards finding a solution to (1). Suppose there are multiple nodes (or cores, CPUs). We let one node (called *master node*) update both primal and dual variables and all the remaining ones (called *worker nodes*) compute and provide block gradients of f to the master node. We assume each g_i is proximable (see the definition in (5) below). When there is a single node, our method reduces to a novel serial primal–dual BCU for solving (1); see Algorithm 1.

1.1 Motivating examples

Problems in the form of (1) arise in many areas including signal processing, machine learning, finance, and statistics. For example, the basis pursuit problem [8] seeks a sparse solution on an affine subspace through solving the linearly constrained program:

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1, \text{ s.t. } \mathbf{A}\mathbf{x} = \mathbf{b}. \tag{2}$$

Partitioning $\mathbf{x}$ into multiple disjoint blocks in an arbitrary way, one can formulate (2) into the form of (1) with $f(\mathbf{x}) = 0$ and each $g_i(\mathbf{x}_i) = \|\mathbf{x}_i\|_1$.

Another example is the portfolio optimization [29]. Suppose we have a unit of capital to invest on m assets. Let x_i be the fraction of capital invested on the ith asset and ξ_i be the expected return rate of the ith asset. The goal is to minimize the risk measured by $\sqrt{\mathbf{x}^\top \boldsymbol{\Sigma} \mathbf{x}}$ subject to total unit capital and minimum expected return c, where $\mathbf{x} = (x_1; \ldots; x_m)$ and $\boldsymbol{\Sigma}$ is the covariance matrix. To find the optimal $\mathbf{x}$, one can solve the problem:

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^\top \boldsymbol{\Sigma} \mathbf{x}, \text{ s.t. } \sum_{i=1}^{m} x_i \le 1, \sum_{i=1}^{m} \xi_i x_i \ge c, \ x_i \ge 0, \forall i. \tag{3}$$

Introducing slack variables to the first two inequalities, one can easily write (3) into the form of (1) with a quadratic f and each g_i being an indicator function of the nonnegativity constraint set.

In addition, (1) includes as a special case the dual support vector machine (SVM) [10]. Given training data set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with $y_i \in \{-1, +1\}$, $\forall i$, let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ and $\mathbf{y} = [y_1; \ldots; y_N]$. The dual form of the linear SVM can be written as

$$\min_{\boldsymbol{\theta}} \frac{1}{2}\boldsymbol{\theta}^\top \mathrm{Diag}(\mathbf{y})\mathbf{X}^\top \mathbf{X}\mathrm{Diag}(\mathbf{y})\boldsymbol{\theta} - \mathbf{e}^\top\boldsymbol{\theta}, \text{ s.t. } \mathbf{y}^\top\boldsymbol{\theta} = 0,\ 0 \le \theta_i \le C,\ \forall i, \tag{4}$$

where $\boldsymbol{\theta} = [\theta_1; \ldots; \theta_N]$, and C is a given number relating to the soft margin size. It is easy to formulate (4) into the form of (1) with f being the quadratic objective function and each g_i the indicator function of the set $[0, C]$.

Finally, the penalized and constrained (PAC) regression problem [22] is also one example of (1) with $f(\mathbf{x}) = \frac{1}{N}\sum_{j=1}^N f_j(\mathbf{x})$ and linear constraint of J equations. As $N \gg J$ (that often holds for problems with massive training data), the PAC regression satisfies Assumption 0 below. In addition, if $m \gg 1$ and $N \gg 1$, both (3) and (4) satisfy that assumption, and thus the proposed async-parallel method will be efficient when applied to these problems. Although Assumption 0 does not hold for (2) as $p > 1$, our method running on a single node can still outperform state-of-the-art non-parallel solvers; see the numerical results in Sect. 4.1.

1.2 Block coordinate update

The block coordinate update (BCU) method breaks possibly very high-dimensional variable into small pieces and renews one at a time while all the remaining blocks are fixed. Although the problem (1) can be extremely large-scale and complicated, BCU solves a sequence of small-sized and easier subproblems. As (1) owns nice structures, e.g., coordinate friendly [31], BCU can not only have low per-update complexity but also enjoy faster overall convergence than the method that updates the whole variable every time. BCU has been applied to many unconstrained or block-separably constrained optimization problems (e.g., [21,30,34,36,40,41,45,46]), and it has also been used to solve affinely constrained separable problems, i.e., in the form of (1) without f term (e.g., [12,13,17–19]). However, only a few existing works (e.g., [14,15, 20]) have studied BCU on solving affinely constrained problems with a nonseparable objective function.

1.3 Asynchronization

Parallel computing methods distribute computation over and collect results from multiple nodes. Synchronous (sync) parallel methods require all nodes to keep in the same pace. Upon all nodes finish their own computation, they altogether proceed to the next step. This way, the faster node has to wait for the slowest one, and that wastes a lot of waiting time. On the contrary, async-parallel methods keep all nodes continuously working and eliminate the idle waiting time. Numerous works (e.g., [27,28,32,35]) have demonstrated that async-parallel methods can achieve significantly better speed-up than their sync-parallel counterparts.

Due to lack of synchronization, the information used by a certain node may be outdated. Hence the convergence of an async-parallel method cannot be easily inherited from its non-parallel counterpart but often requires a new tool of analysis. Most existing works only analyze such methods for unconstrained or block-separably constrained problems. Exceptions include [4,5,42,48] that consider separable problems with special affine constraint.

1.4 Related works

Recent several years have witnessed the surge of async-parallel methods partly due to the increasingly large scale of data/variable involved in modern applications. However, only a few existing works discuss such methods for affinely constrained problems. Below we review the literature of async-parallel BCU methods in optimization and also primal–dual BCU methods for affinely constrained problems.

It appears that the first async-parallel method was proposed by Chazan and Miranker [6] for solving linear systems. Later, such methods have been applied in many others fields. In optimization, the first async-parallel BCU method was due to Bertsekas and Tsitsiklis [1] for problems with a smooth objective. It was shown that the objective gradient sequence converges to zero. Tseng [39] further analyzed its convergence rate and established local linear convergence by assuming isocost surface separation and a local Lipschitz error bound on the objective. Recently, Liu et al. [28] and Liu and Wright [27] developed async-parallel methods based on randomized BCU for convex problems with possibly block separable constraints. They established convergence and also rate results by assuming a bounded delay on the outdated block gradient information. The results have been extended to the case with unbounded probabilistic delay in [33], which also shows convergence of the async-parallel BCU methods for nonconvex problems. On solving problems with convex separable objective and linear constraints, Wei and Ozdaglar [42] proposed to apply the alternating direction method of multipliers (ADMM) in an asynchronous and distributive way. Assuming a special structure on the linear constraint, it established $O(1/k)$ ergodic convergence result, where k is the total number of iterations. In [2,4,5,48], the async-ADMM is applied to distributed multi-agent optimization, which can be equivalently formulated into (1) with $f = 0$ and consensus constraint. Among them, Bianchi et al. [2] proved an almost sure convergence result, Zhang and Kwok [48] showed sub-linear convergence of the async-ADMM for convex problems, and [5] established its linear convergence for strongly convex problems. Besides convex problems, Chang et al. [4] also considered nonconvex cases. Assuming certain structure on the problem and choosing appropriate parameters, it showed that any limit point of the iterates satisfies first-order optimality conditions. The works [9,32] developed async-parallel BCU methods for fixed-point or monotone inclusion problems. Although these settings are more general (including convex optimization as a special case), no convergence rate results have been shown under monotonicity assumption[1] (similar to convexity in optimization).

Running on a single node, the proposed async-parallel method reduces to a serial randomized primal–dual BCU. In the literature, various Gauss–Seidel (GS) cyclic BCU methods have been developed for solving separable convex programs with linear constraints. Although a cyclic primal–dual BCU can empirically work well, in general it may diverge [7,13,44]. By an example of 3×3 linear system, Chen et al. [7] showed that the direct extension of ADMM could diverge on solving problems with more than 2 blocks. The works [13,44] showed that even with proximal terms, the cyclic primal–dual BCU can still diverge. Hence, to guarantee convergence, additional assumptions besides convexity must be made, such as strong convexity on part of the objective [3,11,16,23,25,26] and orthogonality properties of block matrices in the linear constraint [7]. Assuming strong convexity of each block component function and choosing the penalty parameter within a region, Han and Yuan [16] showed the convergence of ADMM to an optimal solution for solving problems with multiple blocks. For 3-block problems, Cai et al. [3], Li et al. [23], Davis and Yin [11] established the convergence of ADMM and/or its variant by assuming strong convexity of one block component function. For general m-block problems, Lin et al. [26] showed that if $m - 1$ block component functions are strongly convex, then ADMM with appropriate penalty parameter is guaranteed to converge. Without these assumptions, modifications to the algorithm are necessary for convergence. For example, He et al. [18,19] performed a correction step after each cycle of updates. On solving linear system or quadratic programming, Sun et al. [38] proposed, at each iteration, to first randomly permute all block variables and then perform a cyclic update. Jacobi-type update together with proximal terms was used in [12,17] to ensure the convergence of the algorithm, which turns out to be a linearized augmented Lagrange method (ALM). In addition, a hybrid Jacobi-GS update was performed in [24,37,44]. Different from these modifications, our algorithm simply employs randomization in selecting block variable and can perform significantly better than Jacobi-type methods. In addition, convergence is guaranteed with convexity assumption and thus better than those results for GS-type methods.

1.5 Contributions

The contributions are summarized as follows.

- We propose an async-parallel BCU method for solving multi-block structured convex programs with linear constraint. The algorithm is the first async-parallel primal–dual method for affinely constrained problems with *nonseparable* objective. When there is only one node, it reduces to a novel serial primal–dual BCU method.
- With convexity and existence of a primal–dual solution, convergence of the proposed method is guaranteed. We first establish convergence of the serial BCU method. We show that the objective value converges in probability to the opti-

[1] In [32], a linear convergence result is established under strong monotonicity assumption, which is similar to strong convexity in optimization.

mal value and also the constraint residual to zero. In addition, we establish an ergodic convergence rate result. Then through bounding a cross term involving delayed block gradient, we prove that similar convergence results hold for the async-parallel BCU method if a delay-dependent stepsize is chosen.

– We implement the proposed algorithm and apply it to the basis pursuit, quadratic programming, and also the support vector machine problems. Numerical results demonstrate that the serial BCU is comparable to or better than state-of-the-art methods. In addition, the async-parallel BCU method can achieve significantly better speed-up performance than its sync-parallel counterpart.

1.6 Notation and outline

We use bold small letters $\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}, \ldots$ for vectors and bold capital letters $\mathbf{A}, \mathbf{L}, \mathbf{P}, \ldots$ for matrices. $[m]$ denotes the integer set $\{1, 2, \ldots, m\}$. $\mathbf{U}_i\mathbf{x}$ represents a vector with $\mathbf{x}_i$ for its ith block and zero for all other $m-1$ blocks. blkdiag$(\mathbf{P}_1, \ldots, \mathbf{P}_m)$ denotes a block diagonal matrix with $\mathbf{P}_1, \ldots, \mathbf{P}_m$ on the diagonal blocks. We denote $\|\mathbf{x}\|$ as the Euclidean norm of $\mathbf{x}$ and $\|\mathbf{x}\|_{\mathbf{P}} = \sqrt{\mathbf{x}^\top \mathbf{P}\mathbf{x}}$ for a symmetric positive semidefinite matrix $\mathbf{P}$. We reserve $\mathbf{I}$ for the identity matrix, and its size is clear from the context. $\mathbb{E}_{i_k}$ stands for the expectation about i_k conditional on previous history $\{i_1, \ldots, i_{k-1}\}$. We use $\boldsymbol{\xi}^k \xrightarrow{p} \boldsymbol{\xi}$ for convergence in probability of a random vector sequence $\boldsymbol{\xi}^k$ to $\boldsymbol{\xi}$.

For ease of notation, we let $g(\mathbf{x}) = \sum_{i=1}^m g_i(\mathbf{x}_i)$, $F = f + g$, and $\mathbf{A} = [\mathbf{A}_1, \ldots, \mathbf{A}_m]$. Denote

$$\Phi(\bar{\mathbf{x}}, \mathbf{x}, \boldsymbol{\lambda}) = F(\bar{\mathbf{x}}) - F(\mathbf{x}) - \langle \boldsymbol{\lambda}, \mathbf{A}\bar{\mathbf{x}} - \mathbf{b}\rangle.$$

Then $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ is a saddle point of (1) if $\mathbf{A}\mathbf{x}^* = \mathbf{b}$ and $\Phi(\mathbf{x}, \mathbf{x}^*, \boldsymbol{\lambda}^*) \geq 0,\ \forall \mathbf{x}$.

The proximal operator of a function ψ is defined as

$$\mathbf{prox}_\psi(\mathbf{x}) = \arg\min_{\mathbf{y}} \psi(\mathbf{y}) + \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|^2. \tag{5}$$

If $\mathbf{prox}_\psi(\mathbf{x})$ has a closed-form solution or is easy to compute, we call ψ *proximable*.

Outline The rest of the paper is organized as follows. In Sect. 2, we present the serial and also async-parallel primal–dual BCU methods for (1). Convergence results of the algorithms are shown in Sect. 3. Section 4 gives experimental results, and finally Sect. 5 concludes the paper.

2 Algorithm

In this section, we propose an async-parallel primal–dual method for solving (1). Our algorithm is a BCU-type method based on the augmented Lagrangian function of (1):

$$\mathcal{L}_\beta(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + g(\mathbf{x}) - \langle \boldsymbol{\lambda}, \mathbf{A}\mathbf{x} - \mathbf{b}\rangle + \frac{\beta}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2,$$

where $\boldsymbol{\lambda}$ is the multiplier (or augmented Lagrangian dual variable), and β is a penalty parameter.

2.1 Non-parallel method

For ease of understanding, we first present a non-parallel method in Algorithm 1. At every iteration, the algorithm chooses one out of m block uniformly at random and renews it by (6) while fixing all the remaining blocks. Upon finishing the update to $\mathbf{x}$, it immediately changes the multiplier $\boldsymbol{\lambda}$. The linearization to possibly complicated smooth term f greatly eases the $\mathbf{x}$-subproblem. Depending on the form of g_i, we can choose appropriate $\mathbf{P}_i$ to make (6) simple to solve. Since each g_i is proximable, one can always easily find a solution to (6) if $\mathbf{P}_i = \eta_i \mathbf{I}$. For even simpler g_i such as ℓ_1-norm and indicator function of a box constraint set, we can set $\mathbf{P}_i$ to a diagonal matrix and have a closed-form solution to (6). Note that the algorithm is a special case of Algorithm 1 in [14] with only one group of variables. We include it here for ease of understanding our parallel method.

Randomly choosing a block to update has advantages over the cyclic way in both theoretical and empirical perspectives. We will show that *this randomized BCU has guaranteed convergence with convexity other than strong convexity assumed by the cyclic primal–dual BCU*. In addition, randomization enables us to parallelize the algorithm in an efficient way as shown in Algorithm 2.

Algorithm 1: Randomized primal–dual block update for (1)

```
1 Initialization: choose $\mathbf{x}^0$ and $\boldsymbol{\lambda}^0 = \mathbf{0}$; let $\mathbf{r}^0 = \mathbf{A}\mathbf{x}^0 - \mathbf{b}$ and $k = 0$; set $\beta$, $\rho$ and $\mathbf{P}_i$'s.
2 while the stopping conditions not satisfied do
3     Pick $i_k$ from $[m]$ uniformly at random.
4     For any $i \neq i_k$, keep $\mathbf{x}_i^{k+1} = \mathbf{x}_i^k$, and for $i = i_k$, update $\mathbf{x}_i$ by
          $$\mathbf{x}_i^{k+1} \in \arg\min_{\mathbf{x}_i} \langle \nabla_i f(\mathbf{x}^k) - \mathbf{A}_i^\top(\boldsymbol{\lambda}^k - \beta \mathbf{r}^k), \mathbf{x}_i \rangle + g_i(\mathbf{x}_i) + \frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_i^k\|_{\mathbf{P}_i}^2, \quad (6)$$
      Update residual $\mathbf{r}$ and multipliers $\boldsymbol{\lambda}$ by
          $$\mathbf{r}^{k+1} = \mathbf{r}^k + \mathbf{A}_{i_k}(\mathbf{x}_{i_k}^{k+1} - \mathbf{x}_{i_k}^k), \quad (7)$$
          $$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k - \rho \mathbf{r}^{k+1}. \quad (8)$$
      Let $k \leftarrow k + 1$.
```

2.2 Async-parallel method

Assume there are p nodes. Let the data and variables be stored in a global memory accessible to every node. We let one node (called *master node*) update both primal variable $\mathbf{x}$ and dual variable $\boldsymbol{\lambda}$ and the remaining ones (called *worker nodes*) compute block gradients of f and provide them to the master node. The method is summarized in Algorithm 2.

To achieve nice practical speed-up performance, we make the following assumption:

Assumption 0 The cost of computing $\nabla_i f(\mathbf{x})$ is roughly at least $p-1$ times of that of updating $\mathbf{x}_i$, $\mathbf{r}$, and $\boldsymbol{\lambda}$ respectively by (9), (7) and (8) for all i, where p is the number of nodes.

Note that our theoretical analysis does not require this assumption. Roughly speaking, the above assumption means that the worker nodes compute block gradients no faster than the master node can use them. When it holds, the master node can quickly digest the block gradient information fed by all worker nodes. Without this assumption, Algorithm 2 may not perform well in terms of parallel efficiency. For example, if $p > 2$, and computing $\nabla_i f(\mathbf{x})$ takes similar time as updating $\mathbf{x}_i$, $\mathbf{r}$ and $\boldsymbol{\lambda}$, then until the kth iteration, there would be roughly $k(p-2)$ partial gradients that have been sent to but not used by the master node. In this case, a lot of computation will be wasted.

We make a few remarks on Algorithm 2 as follows.

- *Special case* If there is only one node (i.e., $p = 1$), the algorithm simply reduces to the non-parallel Algorithm 1. In this case, Assumption 0 trivially holds.
- *Iteration number* Only the master node increases the iteration number k, which counts the times $\boldsymbol{\lambda}$ is updated and also the number of used block gradients. The sync-parallel method (e.g., in [14]) chooses to update multiple blocks every time, and the computation is distributed over multiple nodes. It generally requires larger weight in the proximal term for convergence. Hence, even if $\mathbf{v}^k = \nabla_{i_k} f(\mathbf{x}^k)$, $\forall k$, Algorithm 2 does not reduce to its sync-parallel counterpart.
- *Delayed information* Since all worker nodes provide block gradients to the master node, we cannot guarantee every computed block gradient will be immediately used to update $\mathbf{x}$. Hence, in (9), $\mathbf{v}^k$ may not equal $\nabla_i f(\mathbf{x}^k)$ but can be a delayed (i.e., outdated) block gradient. The delay is usually in the same order of p and can affect the stepsize, but the affect is negligible as the block number m is greater than the delay in an order (see Theorem 3.8).
  Because $\mathbf{x}$-blocks are computed in the master node, the values of $\mathbf{r}$ and $\boldsymbol{\lambda}$ used in the update are always up-to-date. One can let worker nodes compute new $\mathbf{x}_i$'s and then feed them (or also the changes in $\mathbf{r}$) to the master node. That way, $\mathbf{r}$ and $\boldsymbol{\lambda}$ will also be outdated when computing $\mathbf{x}$-blocks.
- *Load balance* Under Assumption 0, if (9) is easy to solve (e.g., $\mathbf{P}_i = \eta_i \mathbf{I}$) and all nodes have similar computing power, the master node will have used all received block gradients before a new one comes. We let the master node itself also compute block gradient if there is no new one sent from any worker node. This way, all nodes work continuously without idle wait. Compared to its sync-parallel counterpart that typically suffers serious load imbalance, the async-parallel can achieve better speed-up; see the numerical results in Sect. 4.3.

Algorithm 2: Async-parallel randomized primal–dual block update for (1)

```
1  Initialization: choose $\mathbf{x}^0$ and $\boldsymbol{\lambda}^0 = \mathbf{0}$; let $\mathbf{r}^0 = \mathbf{A}\mathbf{x}^0 - \mathbf{b}$ and $k = 0$; set $\beta$, $\rho$ and $\mathbf{P}_i$'s.
2  while the stopping conditions not satisfied do
3      if worker node then
4          Pick $j$ from $[m]$ uniformly at random.
5          Read $\mathbf{x}$ from the memory as $\hat{\mathbf{x}}$.
6          Compute $\nabla_j f(\hat{\mathbf{x}})$ and send it together with the block $j$ to master node
7      if master node then
8          if received one new pair $\left(j, \nabla_j f(\hat{\mathbf{x}})\right)$ then
9              Let $i_k = j$ and $\mathbf{v}^k = \nabla_j f(\hat{\mathbf{x}})$
10         else
11             Pick $i_k$ from $[m]$ uniformly at random and let $\mathbf{v}^k = \nabla_{i_k} f(\mathbf{x}^k)$
12         For any $i \neq i_k$, keep $\mathbf{x}_i^{k+1} = \mathbf{x}_i^k$, and for $i = i_k$, update $\mathbf{x}_i$ by
               $\mathbf{x}_i^{k+1} \in \arg\min_{\mathbf{x}_i} \langle \mathbf{v}^k - \mathbf{A}_i^\top(\boldsymbol{\lambda}^k - \beta \mathbf{r}^k), \mathbf{x}_i \rangle + g_i(\mathbf{x}_i) + \frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_i^k\|_{\mathbf{P}_i}^2,$   (9)
           Update residual $\mathbf{r}$ and multipliers $\boldsymbol{\lambda}$ by (7) and (8).
13         Let $k \leftarrow k + 1$.
```

3 Convergence analysis

In this section, we present convergence results of the proposed algorithm. First, we analyze the non-parallel Algorithm 1. We show that the objective value $F(\mathbf{x}^k)$ and the residual $\mathbf{A}\mathbf{x}^k - \mathbf{b}$ converge to the optimal value and zero respectively in probability. In addition, we establish a sublinear convergence rate result based on an averaged point. Then, through bounding a cross term involving the delayed block gradient, we establish similar results for the async-parallel Algorithm 2.

Throughout our analysis, we make the following assumptions.

Assumption 1 (*Existence of a solution*) There exists one pair of primal–dual solution $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ such that $\mathbf{A}\mathbf{x}^* = \mathbf{b}$ and $\Phi(\mathbf{x}, \mathbf{x}^*, \boldsymbol{\lambda}^*) \geq 0, \ \forall \mathbf{x}$.

Assumption 2 (*Gradient Lipschitz continuity*) There exist constants L_i's and L_r such that for any $\mathbf{x}$ and $\mathbf{y}$,

$$\|\nabla_i f(\mathbf{x} + \mathbf{U}_i \mathbf{y}) - \nabla_i f(\mathbf{x})\| \leq L_i \|\mathbf{y}_i\|, \ i = 1, \ldots, m,$$

and

$$\|\nabla f(\mathbf{x} + \mathbf{U}_i \mathbf{y}) - \nabla f(\mathbf{x})\| \leq L_r \|\mathbf{y}_i\|, \ i = 1, \ldots, m.$$

Denote $\mathbf{L} = \mathrm{diag}(L_1, \ldots, L_m)$. Then under the above assumption, it holds that

$$f(\mathbf{x} + \mathbf{U}_i \mathbf{y}) \leq f(\mathbf{x}) + \langle \nabla_i f(\mathbf{x}), \mathbf{y}_i \rangle + \frac{L_i}{2}\|\mathbf{y}_i\|^2, \ \forall i, \ \forall \mathbf{x}, \mathbf{y}. \tag{10}$$

3.1 Convergence results of Algorithm 1

Although Algorithm 1 is a special case of the method in [14], its convergence analysis is easier and can be made more succinct. In addition, our analysis for Algorithm 2 is based on that for Algorithm 1. Hence, we provide a complete convergence analysis for Algorithm 1. First, we establish several lemmas, which will be used to show our main convergence results.

Lemma 3.1 *Let $\{\mathbf{x}^k\}$ be the sequence generated from Algorithm 1. Then for any $\mathbf{x}$ independent of i_k, it holds that*

$$\begin{aligned}\mathbb{E}_{i_k}\left\langle\nabla_{i_k} f(\mathbf{x}^k), \mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}\right\rangle \geq& -\left(1-\frac{1}{m}\right)[f(\mathbf{x}^k)-f(\mathbf{x})]\\ &+\mathbb{E}_{i_k}\left[f(\mathbf{x}^{k+1})-f(\mathbf{x})-\frac{1}{2}\|\mathbf{x}^{k+1}-\mathbf{x}^k\|_{\mathbf{L}}^2\right].\end{aligned}$$

Proof We write $\langle\nabla_{i_k} f(\mathbf{x}^k), \mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}\rangle = \langle\nabla_{i_k} f(\mathbf{x}^k), \mathbf{x}_{i_k}^{k}-\mathbf{x}_{i_k}\rangle + \langle\nabla_{i_k} f(\mathbf{x}^k), \mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}^k\rangle$. For the first term, we use the uniform distribution of i_k on $[m]$ and the convexity of f to have

$$\mathbb{E}_{i_k}\langle\nabla_{i_k} f(\mathbf{x}^k), \mathbf{x}_{i_k}^{k}-\mathbf{x}_{i_k}\rangle = \frac{1}{m}\langle\nabla f(\mathbf{x}^k), \mathbf{x}^k-\mathbf{x}\rangle \geq \frac{1}{m}\big[f(\mathbf{x}^k)-f(\mathbf{x})\big],$$

and for the second term, we use (10) to have

$$\begin{aligned}\langle\nabla_{i_k} f(\mathbf{x}^k), \mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}^k\rangle &\geq f(\mathbf{x}^{k+1})-f(\mathbf{x}^k)-\frac{L_{i_k}}{2}\|\mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}^k\|^2\\ &= f(\mathbf{x}^{k+1})-f(\mathbf{x}^k)-\frac{1}{2}\|\mathbf{x}^{k+1}-\mathbf{x}^k\|_{\mathbf{L}}^2.\end{aligned}\tag{11}$$

Combining the above two inequalities gives the desired result. □

Lemma 3.2 *For any $\mathbf{x}$ independent of i_k such that $\mathbf{A}\mathbf{x}=\mathbf{b}$, it holds*

$$\begin{aligned}&\mathbb{E}_{i_k}\langle-\mathbf{A}_{i_k}^\top(\boldsymbol{\lambda}^k-\beta\mathbf{r}^k), \mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}\rangle\\ &\quad=-\left(1-\frac{1}{m}\right)\left(-\langle\boldsymbol{\lambda}^k,\mathbf{r}^k\rangle+\beta\|\mathbf{r}^k\|^2\right)-\mathbb{E}_{i_k}\langle\boldsymbol{\lambda}^{k+1},\mathbf{r}^{k+1}\rangle+(\beta-\rho)\mathbb{E}_{i_k}\|\mathbf{r}^{k+1}\|^2\\ &\quad\quad-\frac{\beta}{2}\mathbb{E}_{i_k}\left[\|\mathbf{r}^{k+1}\|^2-\|\mathbf{r}^k\|^2+\|\mathbf{x}^{k+1}-\mathbf{x}^k\|_{\mathbf{A}^\top\mathbf{A}}^2\right].\end{aligned}$$

Proof Let $\mathbf{y}^k=-\mathbf{A}^\top(\boldsymbol{\lambda}^k-\beta\mathbf{r}^k)$. Then

$$\begin{aligned}\mathbb{E}_{i_k}\langle\mathbf{y}_{i_k}^k, \mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}\rangle &= \mathbb{E}_{i_k}\langle\mathbf{y}_{i_k}^k, \mathbf{x}_{i_k}^{k}-\mathbf{x}_{i_k}\rangle+\mathbb{E}_{i_k}\langle\mathbf{y}_{i_k}^k, \mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}^k\rangle\\ &= \frac{1}{m}\langle\mathbf{y}^k, \mathbf{x}^k-\mathbf{x}\rangle+\mathbb{E}_{i_k}\langle\mathbf{y}^k, \mathbf{x}^{k+1}-\mathbf{x}^k\rangle\end{aligned}$$

$$= -\left(1-\frac{1}{m}\right)\langle \mathbf{y}^k, \mathbf{x}^k-\mathbf{x}\rangle + \mathbb{E}_{i_k}\langle \mathbf{y}^k, \mathbf{x}^{k+1}-\mathbf{x}\rangle. \tag{12}$$

Note $\mathbf{y}^k = -\mathbf{A}^\top\boldsymbol{\lambda}^{k+1} + (\beta-\rho)\mathbf{A}^\top\mathbf{r}^{k+1} - \beta\mathbf{A}^\top(\mathbf{r}^{k+1}-\mathbf{r}^k)$ and $\mathbf{r}^{k+1}-\mathbf{r}^k = \mathbf{A}(\mathbf{x}^{k+1}-\mathbf{x}^k)$. In addition, from $\mathbf{A}\mathbf{x}=\mathbf{b}$, we have $\mathbf{A}(\mathbf{x}^{k+1}-\mathbf{x}) = \mathbf{r}^{k+1}$. Hence,

$$\begin{aligned}\langle \mathbf{y}^k, \mathbf{x}^{k+1}-\mathbf{x}\rangle =& \langle -\mathbf{A}^\top\boldsymbol{\lambda}^{k+1}, \mathbf{x}^{k+1}-\mathbf{x}\rangle\\ &+ (\beta-\rho)\|\mathbf{r}^{k+1}\|^2 - \beta\big\langle \mathbf{A}(\mathbf{x}^{k+1}-\mathbf{x}^k), \mathbf{A}(\mathbf{x}^{k+1}-\mathbf{x})\big\rangle.\end{aligned} \tag{13}$$

Noting

$$\big\langle \mathbf{A}(\mathbf{x}^{k+1}-\mathbf{x}^k), \mathbf{A}(\mathbf{x}^{k+1}-\mathbf{x})\big\rangle = \frac{1}{2}\big[\|\mathbf{r}^{k+1}\|^2 - \|\mathbf{r}^k\|^2 + \|\mathbf{x}^{k+1}-\mathbf{x}^k\|^2_{\mathbf{A}^\top\mathbf{A}}\big],$$

we complete the proof by plugging (13) into (12). □

Lemma 3.3 *For any* $\mathbf{x}$ *independent of* i_k, *it holds*

$$\mathbb{E}_{i_k}\left\langle \tilde{\nabla} g_{i_k}(\mathbf{x}^{k+1}_{i_k}), \mathbf{x}^{k+1}_{i_k} - \mathbf{x}_{i_k}\right\rangle \ge \mathbb{E}_{i_k}[g(\mathbf{x}^{k+1}) - g(\mathbf{x})] - \left(1-\frac{1}{m}\right)[g(\mathbf{x}^k) - g(\mathbf{x})],$$

where $\tilde{\nabla} g_{i_k}(\mathbf{x}^{k+1}_{i_k})$ *denotes a subgradient of* g_{i_k} *at* $\mathbf{x}^{k+1}_{i_k}$.

Proof From the convexity of g_{i_k} and definition of subgradient, it follows that

$$\mathbb{E}_{i_k}\left\langle \tilde{\nabla} g_{i_k}(\mathbf{x}^{k+1}_{i_k}), \mathbf{x}^{k+1}_{i_k} - \mathbf{x}_{i_k}\right\rangle \ge \mathbb{E}_{i_k}\big[g_{i_k}(\mathbf{x}^{k+1}_{i_k}) - g_{i_k}(\mathbf{x}_{i_k})\big]. \tag{14}$$

Writing $g_{i_k}(\mathbf{x}^{k+1}_{i_k}) - g_{i_k}(\mathbf{x}_{i_k}) = g_{i_k}(\mathbf{x}^{k}_{i_k}) - g_{i_k}(\mathbf{x}_{i_k}) + g_{i_k}(\mathbf{x}^{k+1}_{i_k}) - g_{i_k}(\mathbf{x}^{k}_{i_k})$ and taking the conditional expectation give

$$\mathbb{E}_{i_k}\big[g_{i_k}(\mathbf{x}^{k+1}_{i_k}) - g_{i_k}(\mathbf{x}_{i_k})\big] = \frac{1}{m}\big[g(\mathbf{x}^k) - g(\mathbf{x})\big] + \mathbb{E}_{i_k}\big[g(\mathbf{x}^{k+1}) - g(\mathbf{x}^k)\big].$$

We obtain the desired result by plugging the above equation into (14). □

Using the above three lemmas, we show an inequality after each iteration of the algorithm.

Theorem 3.4 (Fundamental result) *Let* $\{(\mathbf{x}^k, \mathbf{r}^k, \boldsymbol{\lambda}^k)\}$ *be the sequence generated from Algorithm 1. Then for any* $\mathbf{x}$ *such that* $\mathbf{A}\mathbf{x}=\mathbf{b}$, *it holds*

$$\begin{aligned}&\mathbb{E}_{i_k}\left[F(\mathbf{x}^{k+1}) - F(\mathbf{x}) - \langle\boldsymbol{\lambda}^{k+1}, \mathbf{r}^{k+1}\rangle + (\beta-\rho)\|\mathbf{r}^{k+1}\|^2 - \frac{\beta}{2}\|\mathbf{r}^{k+1}\|^2\right]\\ &\quad+ \frac{1}{2}\mathbb{E}_{i_k}\left[\|\mathbf{x}^{k+1}-\mathbf{x}\|^2_{\mathbf{P}} + \|\mathbf{x}^{k+1}-\mathbf{x}^k\|^2_{\mathbf{P}-\mathbf{L}-\beta\mathbf{A}^\top\mathbf{A}}\right]\end{aligned}$$

$$\leq \left(1-\frac{1}{m}\right)\left[F(\mathbf{x}^k)-F(\mathbf{x})-\langle\boldsymbol{\lambda}^k,\mathbf{r}^k\rangle+\beta\|\mathbf{r}^k\|^2\right]-\frac{\beta}{2}\|\mathbf{r}^k\|^2+\frac{1}{2}\|\mathbf{x}^k-\mathbf{x}\|_{\mathbf{P}}^2, \tag{15}$$

where $\mathbf{P}=\text{blkdiag}(\mathbf{P}_1,\ldots,\mathbf{P}_m)$.

Proof Since $\mathbf{x}_{i_k}^{k+1}$ is one solution to (6), there is a subgradient $\tilde{\nabla} g_{i_k}(\mathbf{x}_{i_k}^{k+1})$ of g_{i_k} at $\mathbf{x}_{i_k}^{k+1}$ such that

$$\nabla_{i_k} f(\mathbf{x}^k)-\mathbf{A}_{i_k}^\top(\boldsymbol{\lambda}^k-\beta\mathbf{r}^k)+\tilde{\nabla} g_{i_k}(\mathbf{x}_{i_k}^{k+1})+\mathbf{P}_{i_k}(\mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}^k)=0,$$

Hence,

$$\mathbb{E}_{i_k}\left\langle\nabla_{i_k} f(\mathbf{x}^k)-\mathbf{A}_{i_k}^\top(\boldsymbol{\lambda}^k-\beta\mathbf{r}^k)+\tilde{\nabla} g_{i_k}(\mathbf{x}_{i_k}^{k+1})+\mathbf{P}_{i_k}(\mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}^k),\mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}\right\rangle=0. \tag{16}$$

In the above equation, using Lemmas 3.1 through 3.3 and noting

$$\left\langle\mathbf{P}_{i_k}(\mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}^k),\mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}\right\rangle=\frac{1}{2}\left[\|\mathbf{x}^{k+1}-\mathbf{x}\|_{\mathbf{P}}^2-\|\mathbf{x}^k-\mathbf{x}\|_{\mathbf{P}}^2+\|\mathbf{x}^{k+1}-\mathbf{x}^k\|_{\mathbf{P}}^2\right], \tag{17}$$

we have the desired result. □

Now we are ready to show the convergence results of Algorithm 1.

Theorem 3.5 (Global convergence in probability) *Let* $\{(\mathbf{x}^k,\mathbf{r}^k,\boldsymbol{\lambda}^k)\}$ *be the sequence generated from Algorithm* 1. *If* $0<\rho\leq\frac{\beta}{m}$ *and* $\mathbf{P}_i\succeq L_i\mathbf{I}+\beta\mathbf{A}_i^\top\mathbf{A}_i$, $\forall i$, *then*

$$F(\mathbf{x}^k)\xrightarrow{p}F(\mathbf{x}^*),\ \textit{and}\ \|\mathbf{r}^k\|\xrightarrow{p}0.$$

Before proving the theorem, we make a remark here. The dual stepsize ρ can be up to $\frac{\beta}{m}$, so it could be much smaller than β as m is big. However, note that $\boldsymbol{\lambda}$ is renewed more frequently than $\mathbf{x}$. It is updated once immediately after one change to $\mathbf{x}$. Hence, if $\rho=\frac{\beta}{m}$, after one epoch of $\mathbf{x}$-update, the dual variable $\boldsymbol{\lambda}$ has been updated m times and moved a step of size β. That is why we can still observe fast convergence of the algorithm to the optimal solution even though a small ρ is used; see the numerical results in Sect. 4.

Proof Note that

$$F(\mathbf{x}^k)-F(\mathbf{x})-\langle\boldsymbol{\lambda}^k,\mathbf{r}^k\rangle=\Phi(\mathbf{x}^k,\mathbf{x},\boldsymbol{\lambda})+\langle\boldsymbol{\lambda}-\boldsymbol{\lambda}^k,\mathbf{r}^k\rangle.$$

Hence, taking expectation over both sides of (15) and summing up from $k=0$ through K yield

$$\begin{aligned}
&\mathbb{E}\left[\Phi(\mathbf{x}^{K+1},\mathbf{x},\boldsymbol{\lambda})+\langle\boldsymbol{\lambda}-\boldsymbol{\lambda}^{K+1},\mathbf{r}^{K+1}\rangle\right]\\
&\quad+\frac{1}{m}\sum_{k=1}^{K}\mathbb{E}\left[\Phi(\mathbf{x}^{k},\mathbf{x},\boldsymbol{\lambda})+\langle\boldsymbol{\lambda}-\boldsymbol{\lambda}^{k},\mathbf{r}^{k}\rangle\right]+(\beta-\rho)\mathbb{E}\|\mathbf{r}^{K+1}\|^2\\
&\quad+\left(\frac{\beta}{m}-\rho\right)\sum_{k=1}^{K}\|\mathbf{r}^k\|^2-\frac{\beta}{2}\mathbb{E}\|\mathbf{r}^{K+1}\|^2\\
&\quad+\frac{1}{2}\mathbb{E}\|\mathbf{x}^{K+1}-\mathbf{x}\|_{\mathbf{P}}^2+\frac{1}{2}\sum_{k=0}^{K}\mathbb{E}\|\mathbf{x}^{k+1}-\mathbf{x}^k\|^2_{\mathbf{P}-\mathbf{L}-\beta\mathbf{A}^\top\mathbf{A}}\\
&\quad\le\left(1-\frac{1}{m}\right)\left[F(\mathbf{x}^0)-F(\mathbf{x})-\langle\boldsymbol{\lambda}^0,\mathbf{r}^0\rangle+\beta\|\mathbf{r}^0\|^2\right]+\frac{1}{2}\|\mathbf{x}^0-\mathbf{x}\|_{\mathbf{P}}^2-\frac{\beta}{2}\|\mathbf{r}^0\|^2.
\end{aligned}\tag{18}$$

Since $\boldsymbol{\lambda}^{K+1}=\boldsymbol{\lambda}^K-\rho\mathbf{r}^{K+1}$, it follows from Young's inequality that

$$\langle\boldsymbol{\lambda}-\boldsymbol{\lambda}^{K+1},\mathbf{r}^{K+1}\rangle+(\beta-\rho)\|\mathbf{r}^{K+1}\|^2-\frac{\beta}{2}\|\mathbf{r}^{K+1}\|^2\ge-\frac{1}{2\beta}\|\boldsymbol{\lambda}-\boldsymbol{\lambda}^K\|^2.\tag{19}$$

In addition,

$$\begin{aligned}
\sum_{k=1}^{K}\langle\boldsymbol{\lambda}-\boldsymbol{\lambda}^k,\mathbf{r}^k\rangle&=\frac{1}{2\rho}\sum_{k=1}^{K}\big[\|\boldsymbol{\lambda}-\boldsymbol{\lambda}^k\|^2-\|\boldsymbol{\lambda}-\boldsymbol{\lambda}^{k-1}\|^2\\
&\quad+\|\boldsymbol{\lambda}^{k-1}-\boldsymbol{\lambda}^k\|^2\big]\\
&=\frac{1}{2\rho}\left[\|\boldsymbol{\lambda}-\boldsymbol{\lambda}^K\|^2-\|\boldsymbol{\lambda}-\boldsymbol{\lambda}^0\|^2\right]+\frac{\rho}{2}\sum_{k=1}^{K}\|\mathbf{r}^k\|^2.
\end{aligned}\tag{20}$$

Plugging (19) and (20) into (18) and using $\boldsymbol{\lambda}^0=0$, we have

$$\begin{aligned}
&\mathbb{E}\Phi(\mathbf{x}^{K+1},\mathbf{x},\boldsymbol{\lambda})+\frac{1}{m}\sum_{k=1}^{K}\mathbb{E}\Phi(\mathbf{x}^k,x,\boldsymbol{\lambda})+\left(\frac{\beta}{m}+\frac{\rho}{2m}-\rho\right)\sum_{k=1}^{K}\mathbb{E}\|\mathbf{r}^k\|^2\\
&\quad+\left(\frac{1}{2m\rho}-\frac{1}{2\beta}\right)\mathbb{E}\|\boldsymbol{\lambda}-\boldsymbol{\lambda}^K\|^2\\
&\quad+\frac{1}{2}\mathbb{E}\|\mathbf{x}^{K+1}-\mathbf{x}\|_{\mathbf{P}}^2+\frac{1}{2}\sum_{k=0}^{K}\mathbb{E}\|\mathbf{x}^{k+1}-\mathbf{x}^k\|^2_{\mathbf{P}-\mathbf{L}-\beta\mathbf{A}^\top\mathbf{A}}\\
&\quad\le\left(1-\frac{1}{m}\right)\left[F(\mathbf{x}^0)-F(\mathbf{x})+\beta\|\mathbf{r}^0\|^2\right]+\frac{1}{2}\|\mathbf{x}^0-x\|_{\mathbf{P}}^2-\frac{\beta}{2}\|\mathbf{r}^0\|^2+\frac{1}{2m\rho}\mathbb{E}\|\boldsymbol{\lambda}\|^2.
\end{aligned}\tag{21}$$

Letting $(\mathbf{x},\boldsymbol{\lambda})=(\mathbf{x}^*,\boldsymbol{\lambda}^*)$ in the above equality, we have from $\mathbf{P}_i\succeq L_i\mathbf{I}+\beta\mathbf{A}_i^\top\mathbf{A}_i$ and $\beta\ge m\rho$ that

$$\frac{1}{m}\sum_{k=1}^{K}\mathbb{E}\Phi(\mathbf{x}^k,\mathbf{x}^*,\boldsymbol{\lambda}^*)+\left(\frac{\beta}{m}+\frac{\rho}{2m}-\rho\right)\sum_{k=1}^{K}\mathbb{E}\|\mathbf{r}^k\|^2<\infty,\ \forall K,$$

which together with $|\mathbb{E}\xi|^2 \le \mathbb{E}\xi^2$ implies that

$$\lim_{k\to\infty}\mathbb{E}\Phi(\mathbf{x}^k,\mathbf{x}^*,\boldsymbol{\lambda}^*)=0, \tag{22a}$$

$$\lim_{k\to\infty}\mathbb{E}\|\mathbf{r}^k\|=0. \tag{22b}$$

For any $\epsilon>0$, it follows from the Markov's inequality that

$$\text{Prob}(\|\mathbf{r}^k\|>\epsilon)\le\frac{\mathbb{E}\|\mathbf{r}^k\|}{\epsilon}\to 0,\ \text{as } k\to\infty,$$

and

$$\begin{aligned}&\text{Prob}(|F(\mathbf{x}^k)-F(\mathbf{x}^*)|\ge\epsilon)\\&=\text{Prob}(F(\mathbf{x}^k)-F(\mathbf{x}^*)\ge\epsilon)+\text{Prob}(F(\mathbf{x}^k)-F(\mathbf{x}^*)\le-\epsilon)\\&\le\ \text{Prob}\left(F(\mathbf{x}^k)-F(\mathbf{x}^*)-\langle\boldsymbol{\lambda}^*,\mathbf{r}^k\rangle\ge\frac{\epsilon}{2}\right)+\text{Prob}\left(\langle\boldsymbol{\lambda}^*,\mathbf{r}^k\rangle\ge\frac{\epsilon}{2}\right)+\text{Prob}(-\langle\boldsymbol{\lambda}^*,\mathbf{r}^k\rangle\ge\epsilon)\\&\le\ \text{Prob}\left(F(\mathbf{x}^k)-F(\mathbf{x}^*)-\langle\boldsymbol{\lambda}^*,\mathbf{r}^k\rangle\ge\frac{\epsilon}{2}\right)+\text{Prob}\left(\|\boldsymbol{\lambda}^*\|\cdot\|\mathbf{r}^k\|\ge\frac{\epsilon}{2}\right)+\text{Prob}(\|\boldsymbol{\lambda}^*\|\cdot\|\mathbf{r}^k\|\ge\epsilon)\\&\to 0,\ \text{as } k\to\infty,\end{aligned} \tag{23}$$

where in the first inequality, we have used the fact $F(\mathbf{x})-F(\mathbf{x}^*)-\langle\boldsymbol{\lambda}^*,\mathbf{A}\mathbf{x}-\mathbf{b}\rangle\ge 0,\ \forall\mathbf{x}$, and the last equation follows from (22) and the Markov's inequality. This completes the proof. □

Given any $\epsilon>0$ and $\sigma\in(0,1)$, we can also estimate the number of iterations for the algorithm to produce a solution satisfying an error bound ϵ with probability no less than $1-\sigma$.

Definition 3.1 (*(ϵ,σ)-solution*) Given $\epsilon>0$ and $0<\sigma<1$, a random vector $\mathbf{x}$ is called an (ϵ,σ)-solution to (1) if $\text{Prob}(|F(\mathbf{x})-F(\mathbf{x}^*)|\ge\epsilon)\le\sigma$ and $\text{Prob}(\|\mathbf{A}\mathbf{x}-\mathbf{b}\|\ge\epsilon)\le\sigma$.

Theorem 3.6 (Ergodic convergence rate) *Let $\{(\mathbf{x}^k,\mathbf{r}^k,\boldsymbol{\lambda}^k)\}$ be the sequence generated from Algorithm 1. Assume $0<\rho\le\frac{\beta}{m}$ and $\mathbf{P}_i\succeq L_i\mathbf{I}+\beta\mathbf{A}_i^\top\mathbf{A}_i,\ \forall i$. Let $\bar{\mathbf{x}}^{K+1}=\frac{\mathbf{x}^{K+1}+\sum_{k=1}^{K}\mathbf{x}^{k+1}/m}{1+K/m}$ and*

$$C_0=\left(1-\frac{1}{m}\right)\left[F(\mathbf{x}^0)-F(\mathbf{x}^*)\right]+\frac{1}{2}\|\mathbf{x}^0-\mathbf{x}^*\|_{\mathbf{P}}^2+\left(\frac{\beta}{2}-\frac{\beta}{m}\right)\|\mathbf{r}^0\|^2.$$

Then

$$-\frac{1}{1+K/m}\left(C_0+\frac{2}{m\rho}\|\boldsymbol{\lambda}^*\|^2\right)\le\mathbb{E}[F(\bar{\mathbf{x}}^{K+1})-F(\mathbf{x}^*)]\le\frac{C_0}{1+K/m}, \tag{24}$$

$$\mathbb{E}\|\mathbf{A}\bar{\mathbf{x}}^{K+1}-\mathbf{b}\| \le \frac{1}{1+K/m}\left(C_0+\frac{1}{2m\rho}(1+\|\boldsymbol{\lambda}^*\|)^2\right). \tag{25}$$

In addition, given any $\epsilon > 0$ *and* $0 < \sigma < 1$, *if*

$$K \ge m\cdot\max\left(\frac{C_0+\frac{1}{2m\rho}(1+\|\boldsymbol{\lambda}^*\|)^2}{\epsilon\sigma}-1,\ \frac{5C_0+\frac{13}{2m\rho}\|\boldsymbol{\lambda}^*\|^2}{\epsilon\sigma}-1\right), \tag{26}$$

then $\bar{\mathbf{x}}^{K+1}$ *is an* (ϵ,σ)*-solution to* (1).

Proof Since F is convex, it follows from (21) that

$$\mathbb{E}\Phi(\bar{\mathbf{x}}^{K+1},\mathbf{x},\boldsymbol{\lambda}) \le \frac{1}{1+K/m}\left(C_0+\frac{1}{2m\rho}\mathbb{E}\|\boldsymbol{\lambda}\|^2\right), \tag{27}$$

which with $\mathbf{x}=\mathbf{x}^*$ and $\boldsymbol{\lambda}=0$ implies the second inequality in (24). From $\Phi(\mathbf{x},\mathbf{x}^*,\boldsymbol{\lambda}^*)\ge 0,\ \forall\mathbf{x}$ and Cauchy–Schwarz inequality, we have that

$$F(\mathbf{x})-F(\mathbf{x}^*) \ge -\|\boldsymbol{\lambda}^*\|\cdot\|\mathbf{A}\mathbf{x}-\mathbf{b}\|,\ \forall\mathbf{x}. \tag{28}$$

Letting $\mathbf{x}=\mathbf{x}^*$ and $\boldsymbol{\lambda}=-\frac{1+\|\boldsymbol{\lambda}^*\|}{\|\mathbf{A}\bar{\mathbf{x}}^{K+1}-\mathbf{b}\|}(\mathbf{A}\bar{\mathbf{x}}^{K+1}-\mathbf{b})$ in (27) and using (28) give (25), where we have used the convention $\frac{0}{0}=0$. By Markov's inequality,

$$\text{Prob}(\|\mathbf{A}\bar{\mathbf{x}}^{K+1}-\mathbf{b}\|\ge\epsilon) \le \frac{\mathbb{E}\|\mathbf{A}\bar{\mathbf{x}}^{K+1}-\mathbf{b}\|}{\epsilon},$$

and thus to have $\text{Prob}(\|\mathbf{A}\bar{\mathbf{x}}^{K+1}-\mathbf{b}\|\ge\epsilon)\le\sigma$, it suffices to let

$$K \ge \frac{C_0+\frac{1}{2m\rho}(1+\|\boldsymbol{\lambda}^*\|)^2}{\epsilon\sigma}m-m. \tag{29}$$

Similarly, letting $\mathbf{x}=\mathbf{x}^*$ and $\boldsymbol{\lambda}=-\frac{2\|\boldsymbol{\lambda}^*\|}{\|\mathbf{A}\bar{\mathbf{x}}^{K+1}-\mathbf{b}\|}(\mathbf{A}\bar{\mathbf{x}}^{K+1}-\mathbf{b})$ in (27) and using (28) give

$$\|\boldsymbol{\lambda}^*\|\cdot\mathbb{E}\|\mathbf{A}\bar{\mathbf{x}}^{K+1}-\mathbf{b}\| \le \frac{1}{1+K/m}\left(C_0+\frac{2}{m\rho}\|\boldsymbol{\lambda}^*\|^2\right),$$

which together with (28) implies the first inequality in (24). Through the same arguments that show (23), we have

$$\begin{aligned}
&\text{Prob}(|F(\bar{\mathbf{x}}^{K+1})-F(\mathbf{x}^*)|\ge\epsilon)\\
&\quad\le \text{Prob}\left(\Phi(\bar{\mathbf{x}}^{K+1},\mathbf{x}^*,\boldsymbol{\lambda}^*)\ge\frac{\epsilon}{2}\right)+\text{Prob}\left(\|\boldsymbol{\lambda}^*\|\cdot\|\mathbf{A}\bar{\mathbf{x}}^{K+1}-\mathbf{b}\|\ge\frac{\epsilon}{2}\right)\\
&\qquad+\text{Prob}\left(\|\boldsymbol{\lambda}^*\|\cdot\|\mathbf{A}\bar{\mathbf{x}}^{K+1}-\mathbf{b}\|\ge\epsilon\right)
\end{aligned}$$

$$\leq \frac{\mathbb{E}\Phi\left(\bar{\mathbf{x}}^{K+1}, \mathbf{x}^*, \boldsymbol{\lambda}^*\right)}{\epsilon/2} + \frac{\|\boldsymbol{\lambda}^*\| \cdot \mathbb{E}\|\mathbf{A}\bar{\mathbf{x}}^{K+1} - \mathbf{b}\|}{\epsilon/2} + \frac{\|\boldsymbol{\lambda}^*\| \cdot \mathbb{E}\|\mathbf{A}\bar{\mathbf{x}}^{K+1} - \mathbf{b}\|}{\epsilon}. \tag{30}$$

Hence, to have $\text{Prob}(|F(\bar{\mathbf{x}}^{K+1}) - F(\mathbf{x}^*)| \geq \epsilon) \leq \sigma$, it suffices to let

$$K \geq \frac{5C_0 + \frac{13}{2m\rho}\|\boldsymbol{\lambda}^*\|^2}{\epsilon\sigma} m - m,$$

which together with (29) gives the desired result and thus completes the proof. □

3.2 Convergence results of Algorithm 2

The key difference between Algorithms 1 and 2 is that $\mathbf{v}^k$ used in (9) may not equal the block gradient of f at $\mathbf{x}^k$ but another outdated vector, which we denote as $\hat{\mathbf{x}}^k$. This delayed vector may not be any iterate that ever exists in the memory, i.e., inconsistent read can happen [27]. Besides Assumptions 1 and 2, we make an additional assumption on the delayed vector.

Assumption 3 (*Bounded delay*) The delay is uniformly bounded by an integer τ, and $\hat{\mathbf{x}}^k$ can be related to $\mathbf{x}^k$ by the equation

$$\hat{\mathbf{x}}^k = \mathbf{x}^k + \sum_{d \in J(k)} (\mathbf{x}^d - \mathbf{x}^{d+1}), \tag{31}$$

where $J(k)$ is a subset of $\{k-\tau, k-\tau+1, \ldots, k-1\}$.

The boundedness of the delay holds if there is no "dead" node. The relation between $\mathbf{x}^k$ and $\hat{\mathbf{x}}^k$ in (31) is satisfied if the read of each block variable is consistent, which can be guaranteed by a *dual memory approach*; see [32].

Similar to (16), we have from the optimality condition of (9) that

$$\mathbb{E}_{i_k}\left\langle \nabla_{i_k} f(\hat{\mathbf{x}}^k) - \mathbf{A}_{i_k}^\top(\boldsymbol{\lambda}^k - \beta\mathbf{r}^k) + \tilde{\nabla} g_{i_k}(\mathbf{x}_{i_k}^{k+1}) + \mathbf{P}_{i_k}(\mathbf{x}_{i_k}^{k+1} - \mathbf{x}_{i_k}^k), \mathbf{x}_{i_k}^{k+1} - \mathbf{x}_{i_k}\right\rangle = 0, \tag{32}$$

where we have used $\mathbf{v}^k = \nabla_{i_k} f(\hat{\mathbf{x}}^k)$. Except $\mathbb{E}_{i_k}\langle \nabla_{i_k} f(\hat{\mathbf{x}}^k), \mathbf{x}_{i_k}^{k+1} - \mathbf{x}_{i_k}\rangle$, all the other terms in (32) can be bounded in the same ways as those in Sect. 3.1. We first show how to bound this term and then present the convergence results of Algorithm 2.

Lemma 3.7 *Under Assumptions 2 and 3, we have for any* $\alpha > 0$ *that*

$$\mathbb{E}_{i_k}\langle \nabla_{i_k} f(\hat{\mathbf{x}}^k), \mathbf{x}_{i_k}^{k+1} - \mathbf{x}_{i_k}\rangle$$
$$\geq \mathbb{E}_{i_k}[f(\mathbf{x}^{k+1}) - f(\mathbf{x})] - \left(1 - \frac{1}{m}\right)[f(\mathbf{x}^k) - f(\mathbf{x})] - \frac{1}{2}\mathbb{E}_{i_k}\|\mathbf{x}^{k+1} - \mathbf{x}^k\|_{\mathbf{L}+\alpha L_c \mathbf{I}}^2$$

$$-\frac{\kappa L_r\tau/\alpha+2L_r\tau}{2m}\sum_{d=k-\tau}^{k-1}\|\mathbf{x}^{d+1}-\mathbf{x}^d\|^2-\frac{1}{2m}\sum_{d=k-\tau}^{k-1}\|\mathbf{x}^{d+1}-\mathbf{x}^d\|_{\mathbf{L}}^2, \tag{33}$$

where $L_c=\max_i L_i>0$, *and* $\kappa=\frac{L_r}{L_c}$ *denotes the condition number.*

Proof We split $\mathbb{E}_{i_k}\langle\nabla_{i_k}f(\hat{\mathbf{x}}^k),\mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}\rangle$ into four terms:

$$\begin{aligned}&\mathbb{E}_{i_k}\langle\nabla_{i_k}f(\hat{\mathbf{x}}^k),\mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}\rangle\\&\quad=\mathbb{E}_{i_k}\langle\nabla_{i_k}f(\mathbf{x}^k),\mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}^k\rangle-\mathbb{E}_{i_k}\langle\nabla_{i_k}f(\mathbf{x}^k)-\nabla_{i_k}f(\hat{\mathbf{x}}^k),\mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}^k\rangle\\&\qquad+\mathbb{E}_{i_k}\langle\nabla_{i_k}f(\hat{\mathbf{x}}^k),\mathbf{x}_{i_k}^k-\hat{\mathbf{x}}_{i_k}^k\rangle+\mathbb{E}_{i_k}\langle\nabla_{i_k}f(\hat{\mathbf{x}}^k),\hat{\mathbf{x}}_{i_k}^k-\mathbf{x}_{i_k}\rangle,\end{aligned}\tag{34}$$

and we bound each of the four cross terms in (34). The first is bounded in (11). Secondly, from the convexity of f, we have

$$\mathbb{E}_{i_k}\langle\nabla_{i_k}f(\hat{\mathbf{x}}^k),\hat{\mathbf{x}}_{i_k}^k-\mathbf{x}_{i_k}\rangle=\frac{1}{m}\langle\nabla f(\hat{\mathbf{x}}^k),\hat{\mathbf{x}}^k-\mathbf{x}\rangle\geq\frac{1}{m}[f(\hat{\mathbf{x}}^k)-f(\mathbf{x})]. \tag{35}$$

For the other two terms, we use the relation between $\hat{\mathbf{x}}^k$ and $\mathbf{x}^k$ in (31). From the result in [28, pp. 306], it holds that

$$\|\nabla f(\mathbf{x}^k)-\nabla f(\hat{\mathbf{x}}^k)\|\leq L_r\sum_{d\in J(k)}\|\mathbf{x}^{d+1}-\mathbf{x}^d\|. \tag{36}$$

Hence by Young's inequality, we have for any $\alpha>0$ that

$$\begin{aligned}&-\mathbb{E}_{i_k}\langle\nabla_{i_k}f(\mathbf{x}^k)-\nabla_{i_k}f(\hat{\mathbf{x}}^k),\mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}^k\rangle\\&\quad\geq-\frac{1}{2\alpha L_c}\mathbb{E}_{i_k}\|\nabla_{i_k}f(\mathbf{x}^k)-\nabla_{i_k}f(\hat{\mathbf{x}}^k)\|^2-\frac{\alpha L_c}{2}\mathbb{E}_{i_k}\|\mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}^k\|^2\\&\quad=-\frac{1}{2m\alpha L_c}\|\nabla f(\mathbf{x}^k)-\nabla f(\hat{\mathbf{x}}^k)\|^2-\frac{\alpha L_c}{2}\mathbb{E}_{i_k}\|\mathbf{x}^{k+1}-\mathbf{x}^k\|^2\\&\quad\overset{(36)}{\geq}-\frac{L_r^2|J(k)|}{2m\alpha L_c}\sum_{d\in J(k)}\|\mathbf{x}^{d+1}-\mathbf{x}^d\|^2-\frac{\alpha L_c}{2}\mathbb{E}_{i_k}\|\mathbf{x}^{k+1}-\mathbf{x}^k\|^2.\end{aligned}\tag{37}$$

Let $\tau_k=|J(k)|$ and order the elements in $J(k)$ as $d_1<d_2<\ldots<d_{\tau_k}$. Define $\hat{\mathbf{x}}^{k,0}=\hat{\mathbf{x}}^k$ and $\hat{\mathbf{x}}^{k,j}=\hat{\mathbf{x}}^k+\sum_{t=1}^{j}(\mathbf{x}^{d_t+1}-\mathbf{x}^{d_t}),\ j=1,\ldots,\tau_k$. Then we have

$$\begin{aligned}&\mathbb{E}_{i_k}\langle\nabla_{i_k}f(\hat{\mathbf{x}}^k),\mathbf{x}_{i_k}^k-\hat{\mathbf{x}}_{i_k}^k\rangle\\&\quad=\frac{1}{m}\langle\nabla f(\hat{\mathbf{x}}^k),\mathbf{x}^k-\hat{\mathbf{x}}^k\rangle\\&\quad=\frac{1}{m}\sum_{j=0}^{\tau_k-1}\langle\nabla f(\hat{\mathbf{x}}^k),\hat{\mathbf{x}}^{k,j+1}-\hat{\mathbf{x}}^{k,j}\rangle\end{aligned}$$

$$= \frac{1}{m}\sum_{j=0}^{\tau_k-1}\Big[\langle\nabla f(\hat{\mathbf{x}}^{k,j}), \hat{\mathbf{x}}^{k,j+1}-\hat{\mathbf{x}}^{k,j}\rangle - \langle\nabla f(\hat{\mathbf{x}}^{k,j})-\nabla f(\hat{\mathbf{x}}^{k}), \hat{\mathbf{x}}^{k,j+1}-\hat{\mathbf{x}}^{k,j}\rangle\Big]. \tag{38}$$

Since $\hat{\mathbf{x}}^{k,j+1}-\hat{\mathbf{x}}^{k,j}=\mathbf{x}^{d_{j+1}+1}-\mathbf{x}^{d_{j+1}}$, it follows from (10) that

$$\langle\nabla f(\hat{\mathbf{x}}^{k,j}), \hat{\mathbf{x}}^{k,j+1}-\hat{\mathbf{x}}^{k,j}\rangle \geq f(\hat{\mathbf{x}}^{k,j+1})-f(\hat{\mathbf{x}}^{k,j})-\frac{1}{2}\|\mathbf{x}^{d_{j+1}+1}-\mathbf{x}^{d_{j+1}}\|_{\mathbf{L}}^2. \tag{39}$$

Note $\nabla f(\hat{\mathbf{x}}^{k,j})-\nabla f(\hat{\mathbf{x}}^{k})=\sum_{t=0}^{j-1}(\nabla f(\hat{\mathbf{x}}^{k,t+1})-\nabla f(\hat{\mathbf{x}}^{k,t}))$. Thus, by the Cauchy–Schwarz inequality and the Young's inequality, we have

$$\begin{aligned}
&\langle\nabla f(\hat{\mathbf{x}}^{k,j})-\nabla f(\hat{\mathbf{x}}^{k}), \hat{\mathbf{x}}^{k,j+1}-\hat{\mathbf{x}}^{k,j}\rangle\\
&\leq \sum_{t=0}^{j-1}\|\nabla f(\hat{\mathbf{x}}^{k,t+1})-\nabla f(\hat{\mathbf{x}}^{k,t})\|\cdot\|\hat{\mathbf{x}}^{k,j+1}-\hat{\mathbf{x}}^{k,j}\|\\
&\leq L_r\sum_{t=0}^{j-1}\|\hat{\mathbf{x}}^{k,t+1}-\hat{\mathbf{x}}^{k,t}\|\cdot\|\hat{\mathbf{x}}^{k,j+1}-\hat{\mathbf{x}}^{k,j}\|\\
&\leq \frac{L_r}{2}\sum_{t=0}^{j-1}\left(\|\hat{\mathbf{x}}^{k,t+1}-\hat{\mathbf{x}}^{k,t}\|^2+\|\hat{\mathbf{x}}^{k,j+1}-\hat{\mathbf{x}}^{k,j}\|^2\right).
\end{aligned} \tag{40}$$

Plugging (39) and (40) into (38) gives

$$\begin{aligned}
&\mathbb{E}_{i_k}\langle\nabla_{i_k} f(\hat{\mathbf{x}}^{k}), \mathbf{x}_{i_k}^k-\hat{\mathbf{x}}_{i_k}^k\rangle\\
&\geq \frac{1}{m}\left[f(\mathbf{x}^k)-f(\hat{\mathbf{x}}^k)-\frac{1}{2}\sum_{d\in J(k)}\|\mathbf{x}^{d+1}-\mathbf{x}^d\|_{\mathbf{L}}^2\right]\\
&\quad-\frac{L_r}{2m}\sum_{j=0}^{\tau_k-1}\left(\sum_{t=0}^{j-1}\|\hat{\mathbf{x}}^{k,t+1}-\hat{\mathbf{x}}^{k,t}\|^2+j\|\hat{\mathbf{x}}^{k,j+1}-\hat{\mathbf{x}}^{k,j}\|^2\right)
\end{aligned} \tag{41}$$

Noting $\tau_k\leq\tau$, we have the desired result by plugging (11), (35), (37), and (41) into (34). □

From Lemmas 3.2, 3.3, and 3.7, and also the Eq. (17), we can easily have the following result.

$$\begin{aligned}
&\mathbb{E}_{i_k}\left[F(\mathbf{x}^{k+1})-F(\mathbf{x})-\langle\boldsymbol{\lambda}^{k+1},\mathbf{r}^{k+1}\rangle+(\beta-\rho)\|\mathbf{r}^{k+1}\|^2-\frac{\beta}{2}\|\mathbf{r}^{k+1}\|^2+\frac{1}{2}\|\mathbf{x}^{k+1}-\mathbf{x}\|_{\mathbf{P}}^2\right]\\
&+\frac{1}{2}\mathbb{E}_{i_k}\|\mathbf{x}^{k+1}-\mathbf{x}^k\|^2_{\mathbf{P}-\mathbf{L}-\alpha L_c\mathbf{I}-\beta\mathbf{A}^\top\mathbf{A}}-\frac{\kappa L_r\tau/\alpha+2L_r\tau}{2m}\sum_{d=k-\tau}^{k-1}\|\mathbf{x}^{d+1}-\mathbf{x}^d\|^2
\end{aligned}$$

$$- \frac{1}{2m} \sum_{d=k-\tau}^{k-1} \|\mathbf{x}^{d+1} - \mathbf{x}^d\|_{\mathbf{L}}^2$$
$$\leq \left(1 - \frac{1}{m}\right) \left[F(\mathbf{x}^k) - F(\mathbf{x}) - \langle \boldsymbol{\lambda}^k, \mathbf{r}^k \rangle + \beta \|\mathbf{r}^k\|^2 \right] - \frac{\beta}{2} \|\mathbf{r}^k\|^2 + \frac{1}{2} \|\mathbf{x}^k - \mathbf{x}\|_{\mathbf{P}}^2. \tag{42}$$

Regard $\mathbf{x}^k = \mathbf{x}^0, \ \forall k < 0$. Hence,

$$\sum_{k=0}^{K} \sum_{d=k-\tau}^{k-1} \|\mathbf{x}^{d+1} - \mathbf{x}^d\|^2 \leq \tau \sum_{k=0}^{K-1} \|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2.$$

Using (42) and following the same arguments in the proofs of Theorems 3.5 and 3.6, we obtain the two theorems below.

Theorem 3.8 (Global convergence in probability) *Let* $\{(\mathbf{x}^k, \mathbf{r}^k, \boldsymbol{\lambda}^k)\}$ *be the sequence generated from Algorithm* 2 *with* $0 < \rho \leq \frac{\beta}{m}$ *and* $\mathbf{P}_i$*'s satisfying*

$$\mathbf{P}_i \succeq \left(L_i + \alpha L_c + \frac{\tau L_i}{m} + \frac{(\kappa/\alpha + 2) L_r \tau^2}{m} \right) \mathbf{I} + \beta \mathbf{A}_i^\top \mathbf{A}_i, \ i = 1, \ldots, m, \tag{43}$$

for $\alpha > 0$*, then*

$$F(\mathbf{x}^k) \xrightarrow{p} F(\mathbf{x}^*), \quad \|\mathbf{r}^k\| \xrightarrow{p} 0.$$

Theorem 3.9 (Ergodic convergence rate) *Under the assumptions of Theorem* 3.8*, let* $\bar{\mathbf{x}}^{K+1} = \frac{\mathbf{x}^{K+1} + \sum_{k=1}^{K} \mathbf{x}^{k+1}/m}{1 + K/m}$ *and*

$$C_0 = \left(1 - \frac{1}{m}\right) \left[F(\mathbf{x}^0) - F(\mathbf{x}^*) \right] + \frac{1}{2} \|\mathbf{x}^0 - \mathbf{x}^*\|_{\mathbf{P}}^2 + \left(\frac{\beta}{2} - \frac{\beta}{m} \right) \|\mathbf{r}^0\|^2.$$

Then we have the same results as those in (24) *and* (25)*. In addition, given any* $\epsilon > 0$ *and* $0 < \sigma < 1$*, if* K *satisfies* (26)*, then* $\bar{\mathbf{x}}^{K+1}$ *is an* (ϵ, σ)*-solution to* (1).

Remark 3.1 Comparing the settings of $\mathbf{P}_i$'s in Theorems 3.5 and 3.8, we see that they are only weakly affected by the delay if $\tau = o(\sqrt{m})$, which holds for problems involving extremely many variables. If all p nodes compute at the same rate, τ is in the same order of p [33], and thus Theorem 3.8 indicates that nearly linear speed-up can be achieved on $O(\sqrt{m})$ nodes. Even without the nonseparable affine constraint, this quantity is better than that required in [27]. In addition, as $\tau = 0$, Algorithm 2 reduces to Algorithm 1, and their convergence results coincide.

4 Numerical experiments

In this section, we test the proposed methods on the basis pursuit problem (2), the nonnegativity constrained quadratic programming, and also the dual SVM (4). We demonstrate their efficacy by comparing to several other existing algorithms.

4.1 Basis pursuit

The tests in this subsection compare Algorithm 1 to the linearized augmented Lagrangian method (LALM) and the open-source solver YALL1 [49] on the basis pursuit problem (2). Putting all variables into a single block, we can regard LALM as a special case of Algorithm 1 with $m = 1$, and YALL1 is a linearized ADMM with penalty parameter adaptively updated based on primal and dual residuals.

The matrix $\mathbf{A} \in \mathbb{R}^{q\times 1000}$ in (2) was randomly generated with q varying among $\{200, 300, 400\}$, and its entries independently follow standard Gaussian distribution. We normalized each row of $\mathbf{A}$. A sparse vector $\mathbf{x}^o$ was then generated with 30 nonzero entries that follow standard Gaussian distribution and whose locations are chosen uniformly at random. The vector $\mathbf{b} = \mathbf{A}\mathbf{x}^o$. We evenly partitioned the variable $\mathbf{x}$ into 100 blocks, and we set $\rho = \frac{\beta}{100}$ and $\mathbf{P}_i = \beta\|\mathbf{A}_i\|^2\mathbf{I}$, $i = 1, \ldots, 100$, where $\|\mathbf{A}_i\|$ denotes the spectral norm of $\mathbf{A}_i$. For LALM, we treated it as a special case of Algorithm 1 with a single block and set $\rho = \beta$ and $\mathbf{P} = \beta\|\mathbf{A}\|^2\mathbf{I}$. The same values of β were used for both Algorithm 1 and LALM. The parameters of YALL1 were set to the default values.

To compare the performance of the three algorithms, we plot their values of $|F(\mathbf{x}^t) - F(\mathbf{x}^*)|$ and $\|\mathbf{A}\mathbf{x}^t - \mathbf{b}\|$ with respect to t, where t denotes the epoch number.[2] Since the three algorithms have roughly the same per-epoch complexity, the plot in terms of running time will be similar. In Fig. 1, we fixed $q = 300$ and varied β among $\{1, 10, 100\}$. From the results, we see that the proposed algorithm performed significantly better than LALM and comparably as well as YALL1. In addition, the parameter β affected both Algorithm 1 and LALM but the former was only weakly affected. In Fig. 2, we set $\beta = \sqrt{q}$ and varied q among $\{200, 300, 400\}$. Again we see that the proposed algorithm is significantly better than LALM. For $q = 200$, Algorithm 1 is slightly better than YALL1, and for $q = 300$ and 400, they perform equally well.

4.2 Quadratic programming

In this subsection, we simulate the performance of Algorithm 2 with different delays on solving the nonnegativity constrained quadratic programming (NCQP):

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^\top\mathbf{Q}\mathbf{x} + \mathbf{c}^\top\mathbf{x}, \text{ s.t. } \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0, \tag{44}$$

where $\mathbf{Q}$ is a positive semidefinite matrix. We set $\mathbf{Q} = \mathbf{H}\mathbf{H}^\top$ with $\mathbf{H} \in \mathbb{R}^{2000\times 2000}$ randomly generated from standard Gaussian distribution, and the vector $\mathbf{c}$ was generated from Gaussian distribution. The matrix $\mathbf{A} = [\mathbf{B}, \mathbf{I}] \in \mathbb{R}^{200\times 2000}$ with the entries of $\mathbf{B}$ independently following standard Gaussian distribution, and $\mathbf{b}$ was generated from uniform distribution on $[0, 1]$. This way, we guarantee the feasibility of (44).

We partitioned $\mathbf{x}$ into 2000 blocks, namely, every coordinate was treated as one block. To see how the algorithm is affected by delayed block gradients, $\tau + 1$ most recent iterates were kept, and $\hat{\mathbf{x}}^k$ was set to one of these iterates that was chosen

[2] Each epoch is equivalent to updating m $\mathbf{x}$-blocks.

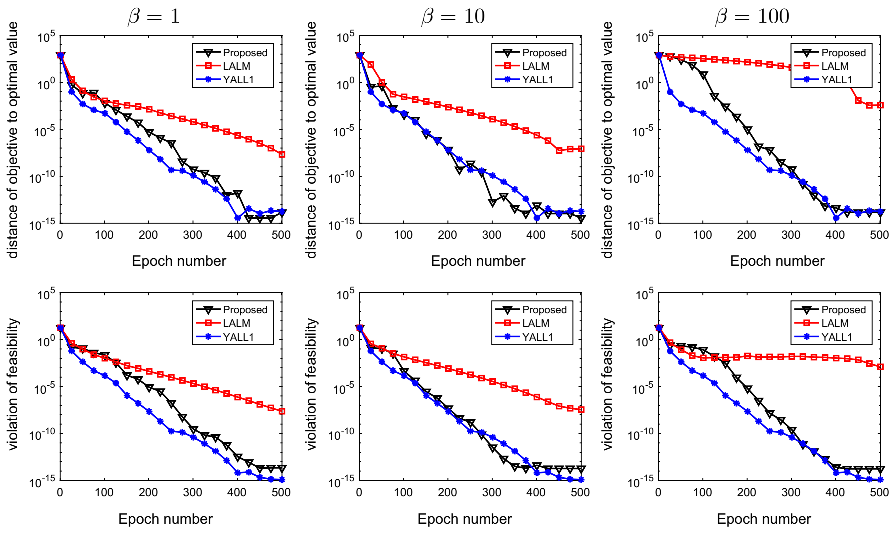

Fig. 1 Results by three different algorithms on solving the basis pursuit problem (2) with $\mathbf{A} \in \mathbb{R}^{300\times1000}$. The parameter β varies among {1, 10, 100} for Algorithm 1 and LALM

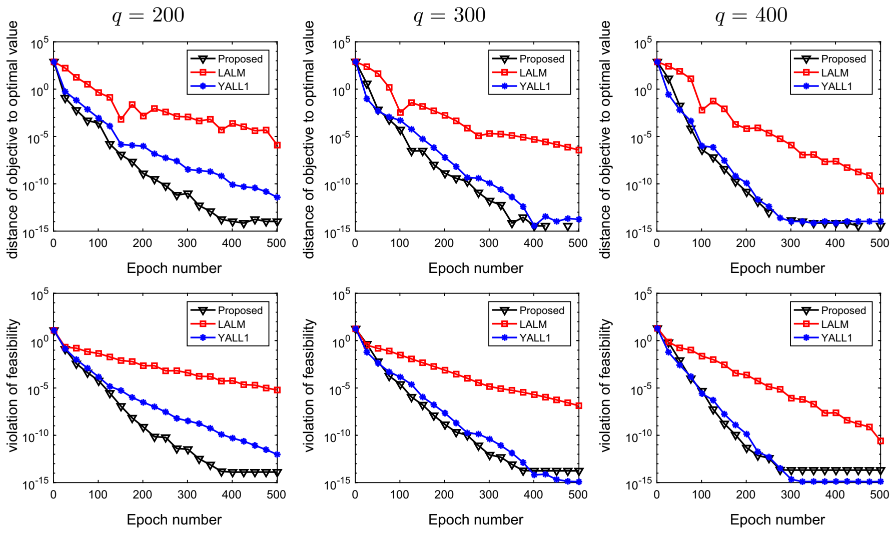

Fig. 2 Results by three different algorithms on solving the basis pursuit problem (2) with $\mathbf{A} \in \mathbb{R}^{q\times1000}$ and q varying among {200, 300, 400}. The parameter β was set to $\sqrt{q}$ for Algorithm 1 and LALM

uniformly at random. We varied τ among {0, 5, 10, 20, 40}. β was tuned to $\sqrt{2}$, $\rho = \frac{\beta}{2000}$ was used, and $\mathbf{P}_i$'s were set in two different ways. Figure 3 plots the results by Algorithm 2 with $\mathbf{P}_i$'s set according to (43) with $\alpha = 1$. Note that for this instance, we have $L_i = Q_{ii}$, i.e., the ith diagonal entry of $\mathbf{Q}$ for each i, and $L_r = \max_i \|\mathbf{q}_i\|$ where $\mathbf{q}_i$ denotes the ith column of $\mathbf{Q}$. From the figure, we see that the convergence speed of the algorithm is affected by the delays. Larger τ gives smaller stepsize and leads

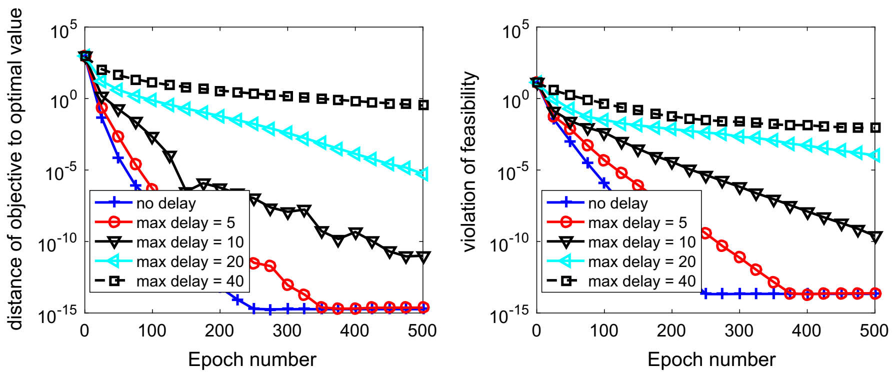

Fig. 3 Results by Algorithm 2 on solving the quadratic programming (44). The matrices $\mathbf{P}_i$'s are set according to (43) with $\alpha = 1$

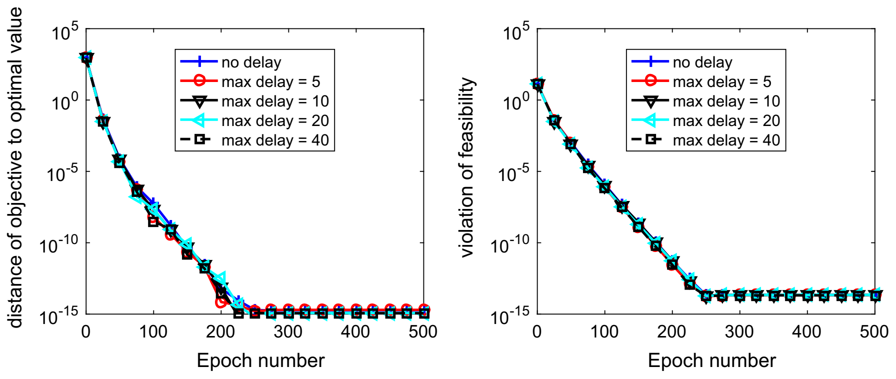

Fig. 4 Results by Algorithm 2 on solving the quadratic programming (44). The same matrices $\mathbf{P}_i$'s are used for different delays, i.e., $\mathbf{P}_i = Q_{ii} + \beta\|\mathbf{a}_i\|^2$, $\forall i$ according to Theorem 3.5, where $\mathbf{a}_i$ is the ith column of $\mathbf{A}$

to slower convergence. However, the algorithm is hardly affected by delayed block gradient if the same $\mathbf{P}_i$'s were used, as shown in Fig. 4. Practically, the maximum delay τ is unknown, but the results in Fig. 4 indicate that we can simply set $\mathbf{P}_i$'s according to Theorem 3.5 regardless of the delay. This implies that our analysis may not be tight.

4.3 Support vector machine

In this subsection, we compare the performance of the async-parallel Algorithm 2 and its sync-parallel counterpart on solving the dual SVM (4). Another way of parallel computing on solving (4) is to directly distribute computation of an algorithm (that may not be BCU type) over multiple nodes, such as the method in [47]. In the test, we used two LIBSVM datasets[3]: `rcv1` and `news20`, whose characteristics are listed in Table 1.

[3] The data can be downloaded from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

Table 1 Characteristics of two LIBSVM datasets

Name	#samples	#features	#nonzeros
rcv1	20,242	47,236	1,498,952
news20	19,996	1,355,191	9,097,916

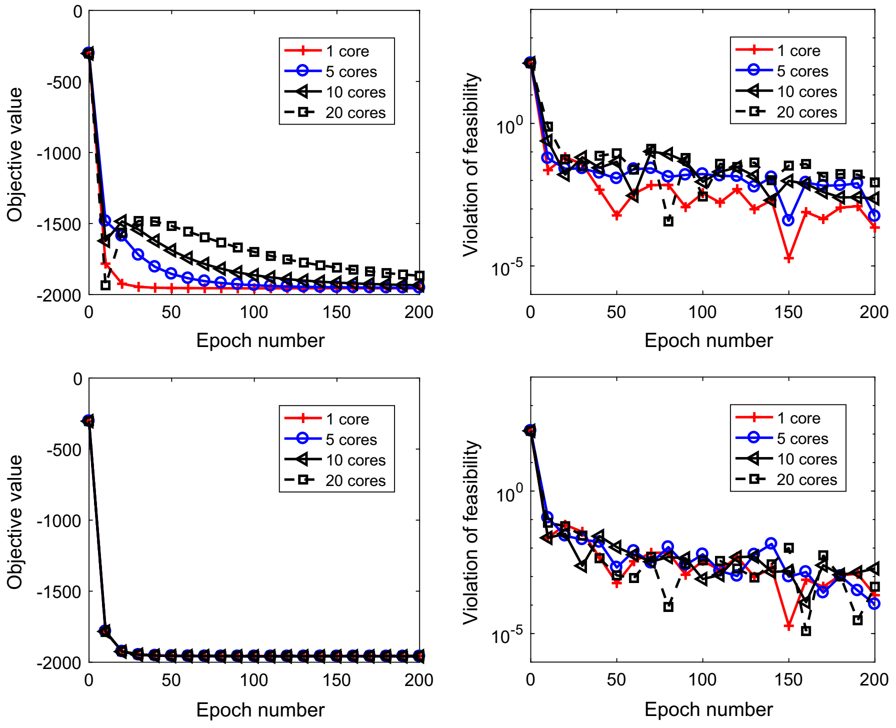

Fig. 5 Results by the sync-parallel (top) and async-parallel (bottom) algorithms on solving the dual SVM (4). The dataset rcv1 is used

We partitioned the variable into blocks of size 50 or 51. For both sync and async-parallel methods, $\beta = 0.1$ and $\rho = \frac{\beta}{m}$ were set, where m is the number of blocks. As suggested in Sect. 4.2, for the async-parallel method, we set $\mathbf{P}_i = (L_i + \beta\|\mathbf{A}_i\|^2)\mathbf{I}$, $\forall i$ according to Theorem 3.5. For the sync-parallel method, if there are p cores, we selected a set S of p blocks at every iteration and set $\mathbf{P}_i = \sum_{j\in S}(L_j + \beta\|\mathbf{A}_j\|^2)\mathbf{I}$ for all $i \in S$. We also used $\mathbf{P}_i$'s the same as those by the async-parallel method but noticed that the sync-parallel method diverged. The larger weight matrices are also suggested in [14] to be proportional to the number of blocks. Note that in the dual SVM (4), if we let $\mathbf{X}_i$ and $\mathbf{y}_i$ contain the data points and labels corresponding to the ith block variable, then L_i equals the spectral norm of the matrix $\text{diag}(\mathbf{y}_i)\mathbf{X}_i^\top\mathbf{X}_i\text{diag}(\mathbf{y}_i)$. Since every block only has 50 or 51 coordinates, it is easy to compute L_i's.

We ran the tests on a machine with 20 cores. Figure 5 plots the results by the sync and async-parallel algorithms on the `rcv1` dataset. From the figure, we see that in

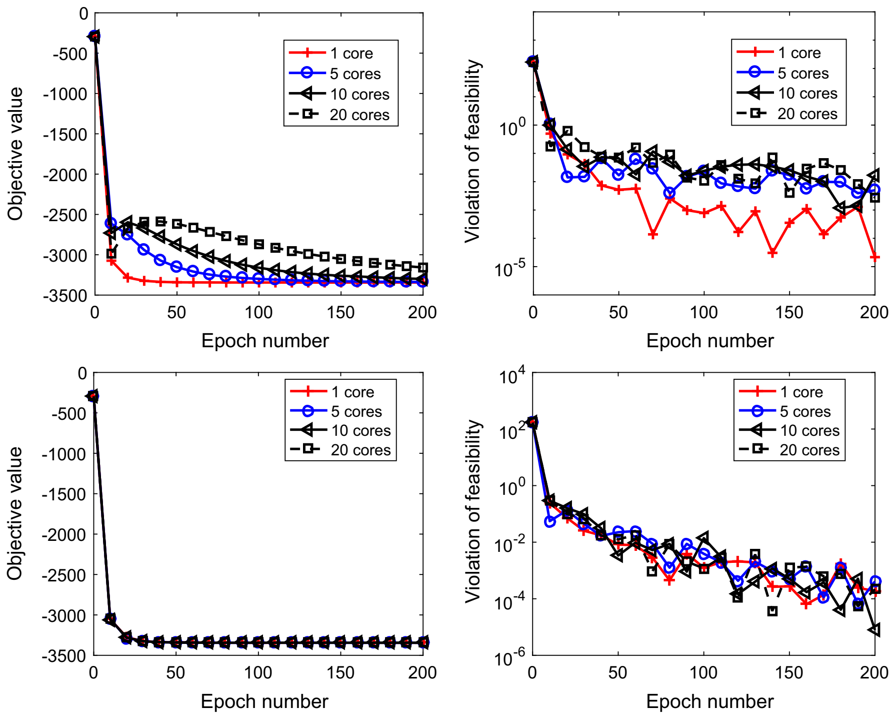

Fig. 6 Results by the sync-parallel (top) and async-parallel (bottom) algorithms on solving the dual SVM (4). The dataset news20 is used

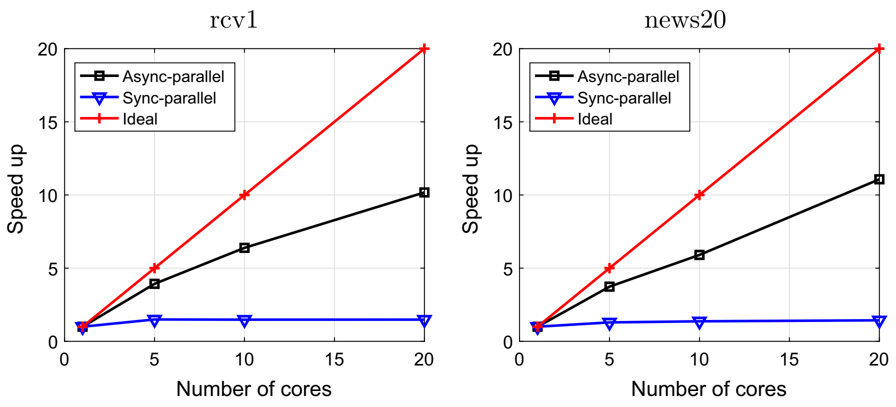

Fig. 7 Speed up of the sync and async-parallel algorithms for solving the dual SVM (4) on different number of cores

terms of epoch number, the sync-parallel method converges slower if more cores are used, while the async-parallel one converges almost the same with different number of cores. As shown in Fig. 6, similar results were observed for the `news20` dataset. We also measured the speed-up of the two parallel methods in terms of running time. The results are plotted in Fig. 7. From the results, we see that the async-parallel method achieves significantly better speed-up than the sync-parallel one, and that is because synchronization at every iteration wastes much waiting time.

5 Conclusions

We have proposed an async-parallel primal–dual BCU method for convex programming with *nonseparable* objective and arbitrary linear constraint. As a special case on a single node, the method reduces to a randomized primal–dual BCU for multi-block linearly constrained problems. Convergence and also rate results in probability have been established under convexity assumption. We have also numerically compared the proposed algorithm to several existing methods. The experimental results demonstrate the superior performance of our algorithm over other ones.

References

1. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and Distributed Computation: Numerical Methods, vol. 23. Prentice Hall, Englewood Cliffs (1989)
2. Bianchi, P., Hachem, W., Iutzeler, F.: A coordinate descent primal–dual algorithm and application to distributed asynchronous optimization. IEEE Trans. Autom. Control **61**(10), 2947–2957 (2016)
3. Cai, X., Han, D., Yuan, X.: On the convergence of the direct extension of ADMM for three-block separable convex minimization models with one strongly convex function. Comput. Optim. Appl. **66**(1), 39–73 (2017)
4. Chang, T.-H., Hong, M., Liao, W.-C., Wang, X.: Asynchronous distributed admm for large-scale optimization—part I: algorithm and convergence analysis. IEEE Trans. Signal Process. **64**(12), 3118–3130 (2016)
5. Chang, T.-H., Liao, W.-C., Hong, M., Wang, X.: Asynchronous distributed admm for large-scale optimization—part II: linear convergence analysis and numerical performance. IEEE Trans. Signal Process. **64**(12), 3131–3144 (2016)
6. Chazan, D., Miranker, W.: Chaotic relaxation. Linear Algebra Appl. **2**(2), 199–222 (1969)
7. Chen, C., He, B., Ye, Y., Yuan, X.: The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent. Math. Programm. **155**(1–2), 57–79 (2016)
8. Chen, S.S., Donoho, D.L., Saunders, M.A.: Atomic decomposition by basis pursuit. SIAM Rev. **43**(1), 129–159 (2001)
9. Combettes, P.L., Eckstein, J.: Asynchronous block-iterative primal–dual decomposition methods for monotone inclusions. Math. Program. **168**(1–2), 645–672 (2018)
10. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
11. Davis, D., Yin, W.: A three-operator splitting scheme and its optimization applications. Set-Valued Var. Anal. **25**(4), 829–858 (2017)
12. Deng, W., Lai, M.-J., Peng, Z., Yin, W.: Parallel multi-block ADMM with $o(1/k)$ convergence. J. Sci. Comput. **71**(2), 712–736 (2017)
13. Feng, J.-K., Zhang, H.-B., Cheng, C.-Z., Pei, H.-M.: Convergence analysis of L-ADMM for multi-block linear-constrained separable convex minimization problem. J. Oper. Res. Soc. China **3**(4), 563–579 (2015)
14. Gao, X., Xu, Y., Zhang, S.: Randomized primal–dual proximal block coordinate updates. arXiv preprint arXiv:1605.05969 (2016)
15. Gao, X., Zhang, S.: First-order algorithms for convex optimization with nonseparable objective and coupled constraints. J. Oper. Res. Soc. China **5**(2), 131–159 (2017)
16. Han, D., Yuan, X.: A note on the alternating direction method of multipliers. J. Optim. Theory Appl. **155**(1), 227–238 (2012)

17. He, B., Hou, L., Yuan, X.: On full Jacobian decomposition of the augmented Lagrangian method for separable convex programming. SIAM J. Optim. **25**(4), 2274–2312 (2015)
18. He, B., Tao, M., Yuan, X.: Alternating direction method with Gaussian back substitution for separable convex programming. SIAM J. Optim. **22**(2), 313–340 (2012)
19. He, B., Tao, M., Yuan, X.: Convergence rate analysis for the alternating direction method of multipliers with a substitution procedure for separable convex programming. Math. Oper. Res. **42**(3), 662–691 (2017)
20. Hong, M., Chang, T.-H., Wang, X., Razaviyayn, M., Ma, S., Luo, Z.-Q.: A block successive upper bound minimization method of multipliers for linearly constrained convex optimization. arXiv preprint arXiv:1401.7079 (2014)
21. Hong, M., Wang, X., Razaviyayn, M., Luo, Z.-Q.: Iteration complexity analysis of block coordinate descent methods. Math. Program. **163**(1–2), 85–114 (2017)
22. James, G.M., Paulson, C., Rusmevichientong, P.: Penalized and Constrained Regression. Technical report (2013)
23. Li, M., Sun, D., Toh, K.-C.: A convergent 3-block semi-proximal ADMM for convex minimization problems with one strongly convex block. Asia-Pac. J. Oper. Res. **32**(04), 1550024 (2015)
24. Li, X., Sun, D., Toh, K.-C.: A Schur complement based semi-proximal ADMM for convex quadratic conic programming and extensions. Math. Program. **155**(1–2), 333–373 (2016)
25. Lin, T., Ma, S., Zhang, S.: On the global linear convergence of the ADMM with multiblock variables. SIAM J. Optim. **25**(3), 1478–1497 (2015)
26. Lin, T., Ma, S., Zhang, S.: On the sublinear convergence rate of multi-block ADMM. J. Oper. Res. Soc. China **3**(3), 251–274 (2015)
27. Liu, J., Wright, S.J.: Asynchronous stochastic coordinate descent: parallelism and convergence properties. SIAM J. Optim. **25**(1), 351–376 (2015)
28. Liu, J., Wright, S.J., Ré, C., Bittorf, V., Sridhar, S.: An asynchronous parallel stochastic coordinate descent algorithm. J. Mach. Learn. Res. **16**, 285–322 (2015)
29. Markowitz, H.: Portfolio selection. J. Finance **7**(1), 77–91 (1952)
30. Nesterov, Y.: Efficiency of coordinate descent methods on huge-scale optimization problems. SIAM J. Optim. **22**(2), 341–362 (2012)
31. Peng, Z., Wu, T., Xu, Y., Yan, M., Yin, W.: Coordinate friendly structures, algorithms and applications. Ann. Math. Sci. Appl. **1**(1), 57–119 (2016)
32. Peng, Z., Xu, Y., Yan, M., Yin, W.: ARock: an algorithmic framework for asynchronous parallel coordinate updates. SIAM J. Sci. Comput. **38**(5), A2851–A2879 (2016)
33. Peng, Z., Xu, Y., Yan, M., Yin, W.: On the convergence of asynchronous parallel iteration with arbitrary delays. arXiv preprint arXiv:1612.04425 (2016)
34. Razaviyayn, M., Hong, M., Luo, Z.-Q.: A unified convergence analysis of block successive minimization methods for nonsmooth optimization. SIAM J. Optim. **23**(2), 1126–1153 (2013)
35. Recht, B., Re, C., Wright, S., Niu, F.: Hogwild: a lock-free approach to parallelizing stochastic gradient descent. In: Advances in Neural Information Processing Systems, pp. 693–701 (2011)
36. Richtárik, P., Takáč, M.: Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. Math. Program. **144**(1–2), 1–38 (2014)
37. Sun, D., Toh, K.-C., Yang, L.: A convergent 3-block semiproximal alternating direction method of multipliers for conic programming with 4-type constraints. SIAM J. Optim. **25**(2), 882–915 (2015)
38. Sun, R., Luo, Z.-Q., Ye, Y.: On the expected convergence of randomly permuted ADMM. arXiv preprint arXiv:1503.06387 (2015)
39. Tseng, P.: On the rate of convergence of a partially asynchronous gradient projection algorithm. SIAM J. Optim. **1**(4), 603–619 (1991)
40. Tseng, P.: Convergence of a block coordinate descent method for nondifferentiable minimization. J. Optim. Theory Appl. **109**(3), 475–494 (2001)
41. Tseng, P., Yun, S.: A coordinate gradient descent method for nonsmooth separable minimization. Math. Program. **117**(1–2), 387–423 (2009)
42. Wei, E., Ozdaglar, A.: On the $o(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers. In: IEEE Global Conference on Signal and Information Processing (GlobalSIP), pp. 551–554 (2013)
43. WhiteHouse. Big Data: Seizing Opportunities Preserving Values (2014)
44. Xu, Y.: Hybrid Jacobian and Gauss–Seidel proximal block coordinate update methods for linearly constrained convex programming. SIAM J. Optim. **28**(1), 646–670 (2018)

45. Xu, Y., Yin, W.: A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. SIAM J. Imaging Sci. **6**(3), 1758–1789 (2013)
46. Xu, Y., Yin, W.: A globally convergent algorithm for nonconvex optimization based on block coordinate update. J. Sci. Comput. **72**(2), 700–734 (2017)
47. Zeng, Z.-Q., Yu, H.-B., Xu, H.-R., Xie, Y.-Q., Gao, J.: Fast training support vector machines using parallel sequential minimal optimization. In: Intelligent System and Knowledge Engineering, 2008. ISKE 2008. 3rd International Conference on, vol. 1, pp. 997–1001. IEEE (2008)
48. Zhang, R., Kwok, J.: Asynchronous distributed ADMM for consensus optimization. In: International Conference on Machine Learning, pp. 1701–1709 (2014)
49. Zhang, Y., Yang, J., Yin, W.: YALL1: your algorithms for l_1. Online at yall1.blogs.rice.edu (2011)