AROCK: AN ALGORITHMIC FRAMEWORK FOR ASYNCHRONOUS PARALLEL COORDINATE UPDATES

ZHIMIN PENG[*], YANGYANG XU[†], MING YAN[‡], AND WOTAO YIN[*]

Abstract. Finding a fixed point to a nonexpansive operator, i.e., $x^* = Tx^*$, abstracts many problems in numerical linear algebra, optimization, and other areas of data sciences. To solve fixed-point problems, we propose ARock, an algorithmic framework in which multiple agents (machines, processors, or cores) update x in an asynchronous parallel fashion. Asynchrony is crucial to parallel computing since it reduces synchronization wait, relaxes communication bottleneck, and thus speeds up computing significantly. At each step of ARock, an agent updates a randomly selected coordinate x_i based on possibly out-of-date information on x. The agents share x through either global memory or communication. If writing x_i is atomic, the agents can read and write x without memory locks.

We prove that if the nonexpansive operator T has a fixed point, then with probability one, ARock generates a sequence that converges to a fixed point of T. Our conditions on T and step sizes are weaker than comparable work. Linear convergence is obtained under suitable assumptions.

We propose special cases of ARock for linear systems, convex optimization, machine learning, as well as distributed and decentralized consensus problems. Numerical experiments of solving sparse logistic regression problems are presented.

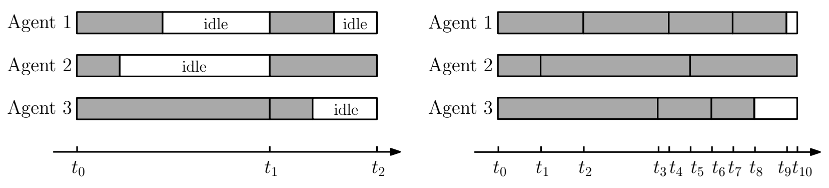
Key words. asynchronous, parallel, coordinate update, nonexpansive operator, ADMM

AMS subject classifications.

1. Introduction.

Technological advances in data gathering and storage have led to a rapid proliferation of big data in diverse areas such as climate studies, cosmology, medicine, the Internet, and engineering [30]. The data involved in many of these modern applications are large and grow quickly. Therefore, parallel computational approaches are needed. This paper introduces a new approach to asynchronous parallel computing with convergence guarantees.

In a synchronous(sync) parallel iterative algorithm, the agents must wait for the slowest agent to finish an iteration before they can all proceed to the next one (Figure 1a). Hence, the slowest agent may cripple the system. In contract, the agents in an asynchronous(async) parallel iterative algorithm run continuously with little idling (Figure 1b). However, the iterations are disordered, and an agent may carry out an iteration without the newest information from other agents.



(a) Sync-parallel computing (b) Async-parallel computing

Fig. 1: Sync-parallel computing (left) versus async-parallel computing (right).

Asynchrony has other advantages [9]: the system is more tolerant to computing faults and communication glitches; it is also easy to incorporate new agents.

[*]Department of Mathematics, University of California, Los Angeles, CA 90095, USA, zhimin.peng/wotaoyin@math.ucla.edu. The research is supported in part by NSF grants DMS-1317602 and ECCS-1462398 and ONR grant N000141410683.

[†]Department of Mathematics, University of Alabama, Tuscaloosa, AL 35487, USA, yxu76@ua.edu

[‡]Department of Computational Mathematics, Science and Engineering and Department of Mathematics, Michigan State University, East Lansing, MI 48824, USA, yanm@math.msu.edu

On the other hand, it is more difficult to analyze asynchronous algorithms and ensure their convergence. It becomes impossible to find a sequence of iterates that one completely determines the next. Nonetheless, we let any update be a new iteration and propose an async-parallel algorithm (ARock) for the generic fixed-point iteration. It converges if the fixed-point operator is nonexpansive (Def. 1.2) and has a fixed point.

Let $\mathcal{H}_1, \dots, \mathcal{H}_m$ be Hilbert spaces and $\mathcal{H} := \mathcal{H}_1 \times \cdots \times \mathcal{H}_m$ be their Cartesian product. For a *nonexpansive operator* $T : \mathcal{H} \to \mathcal{H}$, our problem is to

$$\text{find } x^* \in \mathcal{H} \qquad \text{such that} \qquad x^* = Tx^*. \tag{1.1}$$

Finding a fixed point to T is equivalent to finding a zero of $S \equiv I - T$, denoted by x^* such that $0 = Sx^*$. Hereafter, we will use both S and T for convenience.

Problem (1.1) is widely applicable in linear and nonlinear equations, statistical regression, machine learning, convex optimization, and optimal control. A generic framework for problem (1.1) is the Krasnosel'skiĭ–Mann (KM) iteration [33]:

$$x^{k+1} = x^k + \alpha\,(Tx^k - x^k), \quad \text{or equivalently,} \quad x^{k+1} = x^k - \alpha S x^k, \tag{1.2}$$

where $\alpha \in (0,1)$ is the step size. If $\operatorname{Fix} T$ — the set of fixed points of T (zeros of S) — is nonempty, then the sequence $(x^k)_{k\geq 0}$ converges weakly to a point in $\operatorname{Fix} T$ and $(Tx^k - x^k)_{k\geq 0}$ converges strongly to 0. The KM iteration generalizes algorithms in convex optimization, linear algebra, differential equations, and monotone inclusions. Its special cases include the following iterations: alternating projection, gradient descent, projected gradient descent, proximal-point algorithm, Forward-Backward Splitting (FBS) [45], Douglas-Rachford Splitting (DRS) [37], a three-operator splitting [17], and the Alternating Direction Method of Multipliers (ADMM) [37, 28].

In ARock, a set of p agents, $p \geq 1$, solve problem (1.1) by updating the coordinates $x_i \in \mathcal{H}_i$, $i = 1, \dots, m$, in a random and asynchronous fashion. Algorithm 1 describes the framework. Its special forms for several applications are given in Section 2 below.

Algorithm 1: ARock: a framework for async-parallel coordinate updates

```
Input : $x^0 \in \mathcal{H}$, $K > 0$, a distribution $(p_1, \dots, p_m) > 0$ with $\sum_{i=1}^m p_i = 1$;
global iteration counter $k \leftarrow 0$;
while $k < K$, every agent asynchronously do
    select $i_k \in \{1, \dots, m\}$ with $\text{Prob}(i_k = i) = p_i$;
    perform an update to $x_{i_k}$ according to (1.3);
    update the global counter $k \leftarrow k + 1$;
```

Whenever an agent finishes updating a coordinate, the global iteration counter k increases by one. The kth update is applied to $x_{i_k} \in \mathcal{H}_{i_k}$, where $i_k \in \{1, \dots, m\}$ is an independent random variable. Each coordinate update has the form:

$$x^{k+1} = x^k - \tfrac{\eta_k}{m p_{i_k}} S_{i_k} \hat{x}^k, \tag{1.3}$$

where $\eta_k > 0$ is a scalar whose range will be set later, $S_{i_k} x := (0, ..., 0, (Sx)_{i_k}, 0, ..., 0)$, and $m p_{i_k}$ is used to normalize nonuniform selection probabilities. In the uniform case, namely, $p_i \equiv \frac{1}{m}$ for all i, we have $m p_{i_k} \equiv 1$, which simplifies the update (1.3) to

$$x^{k+1} = x^k - \eta_k S_{i_k} \hat{x}^k. \tag{1.4}$$

Here, the point $\hat{x}^k$ is what an agent reads from global memory to its local cache and to which S_{i_k} is applied, and x^k denotes the state of x in global memory just before the update (1.3) is applied. In a sync-parallel algorithm, we have $\hat{x}^k = x^k$, but in ARock, due to possible updates to x by other agents, $\hat{x}^k$ can be different from x^k. This is a key difference between sync-parallel and async-parallel algorithms. In Subsection 1.2 below, we will establish the relationship between $\hat{x}^k$ and x^k as

$$\hat{x}^k = x^k + \textstyle\sum_{d\in J(k)}(x^d - x^{d+1}), \tag{1.5}$$

where $J(k) \subseteq \{k-1, \ldots, k-\tau\}$ and $\tau \in \mathbb{Z}^+$ is the maximum number of other updates to x during the computation of (1.3). Equation (1.5) has appeared in [38].

The update (1.3) is only computationally worthy if $S_i x$ is much cheaper to compute than Sx. Otherwise, it is more preferable to apply the full KM update (1.2). In Section 2, we will present several applications that have the favorable structures for ARock. Our recent work [46] studies coordinate friendly structures more thoroughly.

The convergence of ARock (Algorithm 1) is stated in Theorems 3.7 and 3.9. Here we include a shortened version, leaving the bounds on η_k to the full theorems:

THEOREM 1.1 (Global and linear convergence). *Let $T : \mathcal{H} \to \mathcal{H}$ be a nonexpansive operator with a fixed point. Let $(x^k)_{k\geq 0}$ be the sequence generated by Algorithm 1 with properly bounded step sizes η_k. Then, with probability one, $(x^k)_{k\geq 0}$ converges weakly to a fixed point of T. This convergence becomes strong if $\mathcal{H}$ has a finite dimension.*

In addition, if T is demicompact (see Definition 1.3 below), then with probability one, $(x^k)_{k\geq 0}$ converges strongly to a fixed point of T .

Furthermore, if $S \equiv I - T$ is quasi-strongly monotone (see Definition 1.2 below), then T has a unique fixed-point x^*, $(x^k)_{k\geq 0}$ converges strongly to x^* with probability one, and $\mathbb{E}\|x^k - x^*\|^2$ converges to 0 at a linear rate.

In the theorem, the weak convergence result only requires T to be nonexpansive and has a fixed point. In addition, the computation requires: (a) bounded step sizes; (b) random coordinate selection; and (c) a finite maximal delay τ. Assumption (a) is standard, and we will see the bound can be $O(1)$. Assumption (b) is essential to both the analysis and the numerical performance of our algorithms. Assumption (c) is *not* essential; an infinite delay with a light tail is allowed (but we leave it to future work). The strong convergence result applies to all the examples in Section 2, and the linear convergence result applies to Examples 2.2 and 2.4 when the corresponding operator S is quasi-strongly monotone. Step sizes η_k are discussed in Remarks 2 and 4.

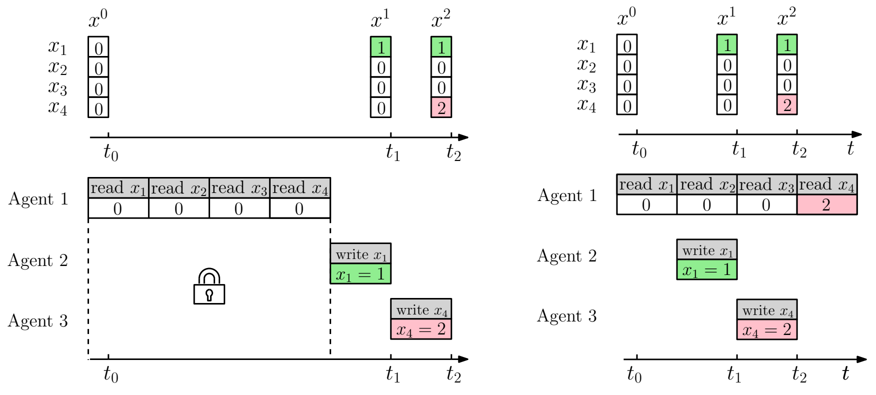
1.1. On random coordinate selection. ARock employs *random coordinate selection*. This subsection discusses its advantages and disadvantages.

Its main disadvantage is that an agent cannot cache the data associated with a coordinate. The variable x and its related data must be either stored in global memory or passed through communication. A secondary disadvantage is that pseudo-random number generations take time, which becomes relatively significant if each coordinate update is cheap. (The network optimization examples in Subsections 2.3 and 2.6.2 are exceptions, where data are naturally stored in a distributed fashion and random coordinate assignments are the results of Poisson processes.)

There are several advantages of random coordinate selection. It realizes the user-specified update frequency p_i for every component x_i, $i = 1, \ldots, m$, even when different agents have different computing powers and different coordinate updates cost different amounts of computation. Therefore, random assignment ensures load balance. The algorithm is also fault tolerant in the sense that if one or more agents fail, it will still converge to a fixed-point of T. In addition, it has been observed numerically on certain problems [12] that random coordinate selection accelerates convergence.

1.2. Uncoordinated memory access. In ARock, since multiple agents simultaneously read and update x in global memory, $\hat{x}^k$ — the result of x that is read from global memory by an agent to its local cache for computation — may not equal x^j for any $j \leq k$, that is, $\hat{x}^k$ may never be consistent with a state of x in global memory. This is known as *inconsistent read*. In contrast, *consistent read* means that $\hat{x}^k = x^j$ for some $j \leq k$, i.e., $\hat{x}^k$ is consistent with a state of x that existed in global memory.

We illustrate inconsistent read and consistent read in the following example, which is depicted in Figure 2. Consider $x = [x_1, x_2, x_3, x_4]^T \in \mathbb{R}^4$ and $x^0 = [0, 0, 0, 0]^T$ initially, at time t_0. Suppose at time t_1, agent 2 updates x_1 from 0 to 1, yielding $x^1 = [1, 0, 0, 0]^T$; then, at time t_2, agent 3 updates x_4 from 0 to 2, further yielding $x^2 = [1, 0, 0, 2]^T$. Suppose that agent 1 starts reading x from the first component x_1 at t_0. For consistent read (Figure 2a), agent 1 acquires a memory lock and only releases the lock after finishing reading all of x_1, x_2, x_3, and x_4. Therefore, agent 1 will read in $[0, 0, 0, 0]^T$. Inconsistent read, however, allows agent 1 to proceed without a memory lock: agent 1 starts reading x_1 at t_0 (Figure 2b) and reaches the last component, x_4, after t_2; since x_4 is updated by agent 3 prior to it is read by agent 1, agent 1 has read $[0, 0, 0, 2]^T$, which is different from any of x^0, x^1, and x^2.



(a) Consistent read. While agent 1 reads x in memory, it acquires a global lock.

(b) Inconsistent read. No lock. Agent 1 reads $(0, 0, 0, 2)^T$, a non-existing state of x.

Fig. 2: Consistent read versus inconsistent read: A demonstration.

Even with inconsistent read, each component is consistent under the *atomic coordinate update* assumption, which will be defined below. Therefore, we can express what has been read in terms of the changes of individual coordinates. In the above example, the first change is $x_1^1 - x_1^0 = 1$, which is added to x_1 just before time t_1 by agent 2, and the second change is $x_4^2 - x_4^1 = 2$, added to x_4 just before time t_2 by agent 3. The inconsistent read by agent 1, which gives the result $[0, 0, 0, 2]^T$, equals $x^0 + 0 \times (x^1 - x^0) + 1 \times (x^2 - x^1)$.

We have demonstrated that $\hat{x}^k$ can be inconsistent, but each of its coordinates is consistent, that is, for each i, $\hat{x}_i^k$ is an ever-existed state of x_i among $x_i^k, \ldots, x_i^{k-\tau}$. Suppose that $\hat{x}_i^k = x_i^{\underline{d}}$, where $\underline{d} \in \{k, k-1, \ldots, k-\tau\}$. Therefore, $\hat{x}_i^k$ can be related to x_i^k through the *interim changes* applied to x_i. Let $J_i(k) \subset \{k-1, \ldots, k-\tau\}$ be the index set of these interim changes. If $J_i(k) \neq \emptyset$, then $\underline{d} = \min\{d \in J_i(k)\}$; otherwise, $\underline{d} = k$. In addition, we have $\hat{x}_i^k = x_i^{\underline{d}} = x_i^k + \sum_{d \in J_i(k)} (x_i^d - x_i^{d+1})$. Since the global counter k is increased after each coordinate update, updates to x_i and x_j, $i \neq j$, must occur at different k's and thus $J_i(k) \cap J_j(k) = \emptyset$, $\forall i \neq j$. Therefore, by letting

$J(k) := \cup_i J_i(k) \subset \{k-1,\dots,k-\tau\}$ and noticing $(x_i^d - x_i^{d+1}) = 0$ for $d \in J_j(k)$ where $i \neq j$, we have $\hat{x}_i^k = x_i^k + \sum_{d\in J(k)}(x_i^d - x_i^{d+1}), \forall i = 1,\dots,m$, which is equivalent to (1.5). Here, we have made two assumptions:

- *atomic coordinate update*: a coordinate is not further broken to smaller components during an update; they are all updated at once.
- *bounded maximal delay* τ: during any update cycle of an agent, x in global memory is updated at most τ times by other agents.

When each coordinate is a single scalar, updating the scalar is a single atomic instruction on most modern hardware, so the first assumption naturally holds, and our algorithm is *lock-free*. The case where a coordinate is a block that includes multiple scalars is discussed in the next subsection.

1.2.1. Block coordinate. In the "block coordinate" case (updating a block of several coordinates each time), the atomic coordinate update assumption can be met by *either* employing a per-coordinate memory lock *or* taking the following dual-memory approach: Store *two* copies of each coordinate $x_i \in \mathcal{H}_i$ in global memory, denoting them as $x_i^{(0)}$ and $x_i^{(1)}$; let a bit $\alpha_i \in \{0,1\}$ point to the active copy; an agent will only read x_i from the active copy $x_i^{(\alpha_i)}$; before an agent updates the components of x_i, it obtains a memory lock to the *inactive* copy $x_i^{(1-\alpha_i)}$ to prevent other agents from simultaneously updating it; then after it finishes updating $x_i^{(1-\alpha_i)}$, flip the bit α_i so that other agents will begin reading from the updated copy. This approach never blocks any read of x_i, yet it eliminates inconsistency.

1.3. Straightforward generalization. Our async-parallel coordinate update scheme (1.3) can be generalized to (overlapping) block coordinate updates after a change to the step size. Specifically, the scheme (1.3) can be generalized to

$$x^{k+1} = x^k - \tfrac{\eta_k}{np_{i_k}}(U_{i_k} \circ S)\hat{x}^k, \tag{1.6}$$

where U_{i_k} is randomly drawn from a set of operators $\{U_1,\dots,U_n\}$ $(n \le m)$, $U_i : \mathcal{H} \to \mathcal{H}$, following the probability $P(i_k = i) = p_i$, $i = 1,\dots,n$ ($p_i > 0$, and $\sum_{i=1}^n p_i = 1$). The operators must satisfy $\sum_{i=1}^n U_i = I_{\mathcal{H}}$ and $\sum_{i=1}^n \|U_i x\|^2 \le C\|x\|^2$ for some $C > 0$.

Let $U_i : x \mapsto (0,\dots,0,x_i,0,\dots,0)$, $i = 1,\dots,m$, which has $C = 1$; then (1.6) reduces to (1.3). If $\mathcal{H}$ is endowed with a metric M such that $\rho_1\|x\|^2 \le \|x\|_M^2 \le \rho_2\|x\|^2$ (e.g., the metric in the Condat-Vũ primal-dual splitting [16, 57]), then we have

$$\sum_{i=1}^m \|U_i x\|_M^2 \le \rho_2 \sum_{i=1}^m \|U_i x\|^2 = \rho_2 C\|x\|^2 \le \frac{\rho_2 C}{\rho_1}\|x\|_M^2.$$

In general, multiple coordinates can be updated in (1.6). Consider linear $U_i : x \mapsto (a_{i1}x_1,\cdots,a_{im}x_m)$, $i = 1,\dots,m$, where $\sum_{i=1}^n a_{ij} = 1$ for each j. Then, for $C := \max\left\{\sum_{i=1}^n a_{i1}^2,\cdots,\sum_{i=1}^n a_{im}^2\right\}$, we have

$$\sum_{i=1}^n \|U_i x\|^2 = \sum_{i=1}^n\sum_{j=1}^m a_{ij}^2\|x_j\|^2 = \sum_{j=1}^m\sum_{i=1}^n a_{ij}^2\|x_j\|^2 \le C\|x\|^2.$$

This generalization is useful for applying ARock to primal-dual splitting schemes (see Appendix D of [46] for detailed discussion).

1.4. Special cases. If there is only one agent ($p = 1$), ARock (Algorithm 1) reduces to randomized coordinate update, which includes the special case of randomized coordinate descent [42] for convex optimization. Sync-parallel coordinate update is another special case of ARock corresponding to $\hat{x}^k \equiv x^k$. In both cases, there is no delay, i.e., $\tau = 0$ and $J(k) = \emptyset$. In addition, the step size η_k can be more relaxed. In particular, if $p_i = \frac{1}{m}$, $\forall i$, then we can let $\eta_k = \eta$, $\forall k$, for any $\eta < 1$, or $\eta < 1/\alpha$ when T is α-averaged (see Definition 1.3 for the definition of an α-averaged operator).

1.5. An alternative way of counting iterations. Throughout the paper, the iteration counter k increments, i.e., a new iteration is defined, once an agent finishes making an update. Alternatively, the paper [36] increments k right after an agent finishes reading $\hat{x}^k$. The two approaches define new iterations in different ways, and they have both pros and cons. In the former approach, $\hat{x}^k$ is read before the kth iteration is defined, and thus $J(k)$ in (1.5) depends on how long it takes to compute (1.4). Because it may take longer to compute for some coordinates than the others, $J(k)$ depends on i_k; however, their statistical independence is assumed in analysis. In the latter approach, once $\hat{x}^k$ is read, a new iteration is defined. If we further define $x^k := \sum_{j=0}^{k-1} \eta_j S_{i_j} \hat{x}^j$, then $J(k)$ in (1.5) does not depends on how long it takes to compute (1.4) (instead, $J(k)$ depends only on the previous iterations $k-1, \ldots, k-\tau$ that overlap with the reading of $\hat{x}^k$); hence, this gives the statistical independence between $J(k)$ and i_k required by analysis. However, the latter approaches bring new issues: (i) the assumptive x^k becomes a "halo sequence", instead of the status of x in the global memory; (ii) the independence would hold only if every agent reads the entire $\hat{x}^k$ even if computing $S_{i_k}\hat{x}^k$ does not require so. In practice, we always sample i_k then read $\hat{x}^k$ and only read its useful components, so there is no way to make them independent just by redefining iterations. Anyway, no matter which definition is used, the same line of analysis in this paper is applicable.

1.6. Related work. Chazan and Miranker [14] proposed the first async-parallel method in 1969. The method was designed for solving linear systems. Later, async-parallel methods have been successful applied in many fields, e.g., linear systems [3, 10, 25, 53], nonlinear problems [4, 5], differential equations [1, 2, 13, 20], consensus problems [35, 24], and optimization [31, 38, 39, 54, 61]. We review the theory for async-parallel fixed-point iterations and its applications.

General fixed point problems. *Totally async-parallel*[*] iterative methods for a fixed-point problem go back as early as to Baudet [5], where the operator was assumed to be *P-contraction*.[†] Later, Bertsekas [7] generalized the P-contraction assumption and showed convergence. Frommer and Szyld [26] reviewed the theory and applications of totally async-parallel iterations prior to 2000. This review summarized convergence results under the conditions in [7]. However, ARock can be applied to solve many more problems since our nonexpansive assumption, though not strictly weaker than P-contraction, is more pervasive. As opposed to totally asynchronous methods, Tseng, Bertsekas, and Tsitsiklis [8, 56] assumed *quasi-nonexpansiveness*[‡] and proposed an async-parallel method, converging under an additional assumption,

[*] "Totally asynchronous" means no upper bound on the delays; however, other conditions are required, for example: each coordinate must be updated infinitely many times. By default, "asynchronous" in this paper assumes a finite maximum delay.

[†] An operator $T : \mathbb{R}^n \to \mathbb{R}^n$ is P-contraction if $|T(x) - T(y)| \le P|x - y|$, component-wise, where $|x|$ denotes the vector with components $|x_i|$, $i = 1, ..., n$, and $P \in \mathbb{R}^{n\times n}$ is a nonnegative matrix with a spectral radius strictly less than 1.

[‡] An operator $T : \mathcal{H} \to \mathcal{H}$ is quasi-nonexpansive if $\|Tx - x^*\| \le \|x - x^*\|$, $\forall x \in \mathcal{H}$, $x^* \in \operatorname{Fix} T$.

which is difficult to justify in general but can be established for problems such as linear systems and strictly convex network flow problems [8, 56].

The above works assign coordinates in a deterministic manner. Different from them, ARock is stochastic, works for nonexpansive operators, and is more applicable.

Linear, nonlinear, and differential equations. The first async-parallel method for solving linear equations was introduced by Chazan and Miranker in [14]. They proved that on solving linear systems, P-contraction was necessary and sufficient for convergence. The performance of the algorithm was studied by Iain et al. [10, 53] on different High Performance Computing (HPC) architectures. Recently, Avron et al. [3] revisited the async-parallel coordinate update and showed its linear convergence for solving positive-definite linear systems. Tarazi and Nabih [23] extended the poineering work [14] to solving nonlinear equations, and the async-parallel methods have also been applied for solving differential equations, e.g., in [1, 2, 13, 20]. Except for [3], all these methods are totally async-parallel with the P-contraction condition or its variants. On solving a positive-definite linear system, [3] made assumptions similar to ours, and it obtained better linear convergence rate on that special problem.

Optimization. The first async-parallel coordinate update gradient-projection method was due to Bertsekas and Tsitsiklis [8]. The method solves constrained optimization problems with a smooth objective and simple constraints. It was shown that the objective gradient sequence converges to zero. Tseng [55] further analyzed the convergence rate and obtained local linear convergence based on the assumptions of isocost surface separation and a local Lipschitz error bound. Recently, Liu et al. [39] developed an async-parallel stochastic coordinate descent algorithm for minimizing convex smooth functions. Later, Liu and Wright [38] suggested an async-parallel stochastic proximal coordinate descent algorithm for minimizing convex composite objective functions. They established the convergence of the expected objective-error sequence for convex functions. Hsieh et al. [31] proposed an async-parallel dual coordinate descent method for solving ℓ_2 regularized empirical risk minimization problems. Other async-parallel approaches include asynchronous ADMM [29, 58, 61, 32]. Among them, [58, 32] use an asynchronous clock, and [29, 61] use a central node to update the dual variable; they do not deal with delay or inconsistency. Async-parallel stochastic gradient descent methods have also been considered in [40, 50].

Our framework differs from the recent surge of the aforementioned sync-parallel and async-parallel coordinate descent algorithms (e.g., [47, 34, 39, 38, 31, 51]). While they apply to convex function minimization, ARock covers more cases (such as ADMM, primal-dual, and decentralized methods) and also provides sequence convergence. In Section 2, we will show that some of the existing async-parallel coordinate descent algorithms are special cases of ARock, through relating their optimality conditions to nonexpansive operators. Another difference is that the convergence of ARock only requires a nonexpansive operator with a fixed point, whereas properties such as strong convexity, bounded feasible set, and bounded sequence, which are seen in some of the recent literature for async-parallel convex minimization, are unnecessary.

Others. Besides solving equations and optimization problems, there are also applications of async-parallel algorithms to optimal control problems [35], network flow problems [22], and consensus problems of multi-agent systems [24].

1.7. Contributions. Our contributions and techniques are summarized below:

- ARock is the first async-parallel coordinate update framework for finding a fixed point to a nonexpansive operator.
- By introducing a new metric and establishing stochastic Fejér monotonicity,

we show that, with probability one, ARock converges to a point in the solution set; linear convergence is obtained for *quasi-strongly monotone* operators.
- Based on ARock, we introduce an async-parallel algorithm for linear systems, async-parallel ADMM algorithms for distributed or decentralized computing problems, as well as async-parallel operator-splitting algorithms for nonsmooth minimization problems. Some problems are treated in they async-parallel fashion for the first time in history. The developed algorithms are *not* straightforward modifications to their serial versions because their underlying nonexpansive operators must be identified before applying ARock.

1.8. Notation, definitions, background of monotone operators. Throughout this paper, $\mathcal{H}$ denotes a separable Hilbert space equipped with the inner product $\langle \cdot, \cdot \rangle$ and norm $\|\cdot\|$, and $(\Omega, \mathcal{F}, P)$ denotes the underlying probability space, where Ω, $\mathcal{F}$, and P are the sample space, σ-algebra, and probability measure, respectively. The map $x : (\Omega, \mathcal{F}) \to (\mathcal{H}, \mathcal{B})$, where $\mathcal{B}$ is the Borel σ-algebra, is an $\mathcal{H}$-valued random variable. Let $(x^k)_{k\geq 0}$ denote *either* a sequence of deterministic points in $\mathcal{H}$ *or* a sequence of $\mathcal{H}$-valued random variables, which will be clear from the context, and let $x_i \in \mathcal{H}_i$ denote the ith coordinate of x. In addition, we let $\mathcal{X}^k := \sigma(x^0, \hat{x}^1, x^1, ..., \hat{x}^k, x^k)$ denote the smallest σ-algebra generated by $x^0, \hat{x}^1, x^1, ..., \hat{x}^k, x^k$. "Almost surely" is abbreviated as "a.s.", and the n product space of $\mathcal{H}$ is denoted by $\mathcal{H}^n$. We use $\to$ and $\rightharpoonup$ for strong convergence and weak convergence, respectively.

We define $\operatorname{Fix} T := \{x \in \mathcal{H} \mid Tx = x\}$ as the set of fixed points of operator T, and, in the product space, we let $\mathbf{X}^* := \{(x^*, x^*, ..., x^*) \mid x^* \in \operatorname{Fix} T\} \subseteq \mathcal{H}^{\tau+1}$.

Definition 1.2. *An operator $T : \mathcal{H} \to \mathcal{H}$ is* c-Lipschitz, *where $c \geq 0$, if it satisfies $\|Tx - Ty\| \leq c\|x - y\|$, $\forall x, y \in \mathcal{H}$. In particular, T is* nonexpansive *if $c \leq 1$, and* contractive *if $c < 1$.*

Definition 1.3. *Consider an operator $T : \mathcal{H} \to \mathcal{H}$.*
- *T is* α-averaged *with $\alpha \in (0, 1)$, if there is a nonexpansive operator $R : \mathcal{H} \to \mathcal{H}$ such that $T = (1 - \alpha)I_{\mathcal{H}} + \alpha R$, where $I_{\mathcal{H}} : \mathcal{H} \to \mathcal{H}$ is the identity operator.*
- *T is* β-cocoercive *with $\beta > 0$, if $\langle x - y, Tx - Ty\rangle \geq \beta\|Tx - Ty\|^2$, $\forall x, y \in \mathcal{H}$.*
- *T is* μ-strongly monotone, *where $\mu > 0$, if it satisfies $\langle x - y, Tx - Ty\rangle \geq \mu\|x - y\|^2$, $\forall x, y \in \mathcal{H}$. When the inequality holds for $\mu = 0$, T is* monotone.
- *T is* quasi-μ-strongly monotone, *where $\mu > 0$, if it satisfies $\langle x - y, Tx\rangle \geq \mu\|x - y\|^2$, $\forall x \in \mathcal{H}, y \in \operatorname{zer} T := \{y \in \mathcal{H} \mid Ty = 0\}$. When the inequality holds for $\mu = 0$, T is* quasi-monotone.
- *T is* demicompact [48] at *$x \in \mathcal{H}$ if for every bounded sequence $(x^k)_{k\geq 0}$ in $\mathcal{H}$ such that $Tx^k - x^k \to x$, there exists a strongly convergent subsequence.*

Averaged operators are nonexpansive. By the Cauchy-Schwarz inequality, a β-cocoercive operator is $\frac{1}{\beta}$-Lipschitz; the converse is generally untrue, but true for the gradients of convex differentiable functions. Examples are given in the next section.

2. Applications.

In this section, we provide some applications that are special cases of the fixed-point problem (1.1). For each application, we identify its nonexpansive operator T (or the corresponding operator S) and implement the conditions in Theorem 1.1. For simplicity, we use the uniform distribution, $p_1 = \cdots = p_m = 1/m$, and apply the simpler update (1.4) instead of (1.3).

2.1. Solving linear equations. Consider the linear system $Ax = b$, where $A \in \mathbb{R}^{m\times m}$ is a nonsingular matrix with nonzero diagonal entries. Let $A = D + R$, where D and R are the diagonal and off-diagonal parts of A, respectively. Let $M := -D^{-1}R$ and $T(x) := Mx + D^{-1}b$. Then the system $Ax = b$ is equivalent to the fixed-point problem $x = D^{-1}(b - Rx) =: T(x)$, where T is nonexpansive if the spectral norm $\|M\|_2$ satisfies $\|M\|_2 \leq 1$. The iteration $x^{k+1} = T(x^k)$ is widely known as the Jacobi algorithm. Let $S = I - T$. Each update $S_{i_k}\hat{x}^k$ involves multiplying just the i_kth row of M to x and adding the i_kth entry of $D^{-1}b$, so we arrive at the following algorithm.

Algorithm 2: ARock for linear equations

```
Input : $x^0 \in \mathbb{R}^n$, $K > 0$.
set the global iteration counter $k = 0$;
while $k < K$, every agent asynchronously do
    select $i_k \in \{1, \dots, m\}$ uniformly at random;
    subtract $\frac{\eta_k}{a_{i_k i_k}}(\sum_j a_{i_k j}\, \hat{x}_j^k - b_{i_k})$ from the component $x_{i_k}$ of the variable $x$;
    update the global counter $k \leftarrow k + 1$;
```

Proposition 2.1. *[6, Example 22.5] Suppose that T is c-Lipschitz continuous with $c \in [0, 1)$. Then, $I - T$ is $(1 - c)$-strongly monotone.*

Suppose $\|M\|_2 < 1$. Since T is $\|M\|_2$-Lipschitz continuous, by Proposition 2.1, S is $(1 - \|M\|_2)$-strongly monotone. By Theorem 3.9, Algorithm 2 converges linearly.

2.2. Minimize convex smooth function. Consider the optimization problem

$$\operatorname*{minimize}_{x \in \mathcal{H}} f(x), \tag{2.1}$$

where f is a closed proper convex differentiable function and ∇f is L-Lipschitz continuous, $L > 0$. Let $S := \frac{2}{L}\nabla f$. As f is convex and differentiable, x is a minimizer of f if and only if x is a zero of S. Note that S is $\frac{1}{2}$-cocoercive. By Lemma 3.1, $T \equiv I - S$ is nonexpansive. Applying ARock, we have the following iteration:

$$x^{k+1} = x^k - \eta_k S_{i_k}\hat{x}^k, \tag{2.2}$$

where $S_{i_k}x = \frac{2}{L}(0, ..., 0, \nabla_{i_k} f(x), 0, ..., 0)^T$. Note that ∇f needs a structure that makes it cheap to compute $\nabla_{i_k} f(\hat{x}^k)$. Let us give two such examples: (i) quadratic programming: $f(x) = \frac{1}{2}x^T Ax - b^T x$, where $\nabla f(x) = Ax - b$ and $\nabla_{i_k} f(\hat{x}^k)$ only depends on a part of A and b; (ii) sum of sparsely supported functions: $f = \sum_{j=1}^N f_j$ and $\nabla f = \sum_{j=1}^N \nabla f_j$, where each f_j depends on just a few variables.

Theorem 3.7 below guarantees the convergence of $(x^k)_{k\geq 0}$ if $\eta_k \in [\eta_{\min}, \frac{1}{2\tau/\sqrt{m}+1})$. In addition, If $f(x)$ is *restricted strongly convex*, namely, for any $x \in \mathcal{H}$ and $x^* \in X^*$, where X^* is the solution set to (2.1), we have $\langle x - x^*, \nabla f(x)\rangle \geq \mu\|x - x^*\|^2$ for some $\mu > 0$, then S is quasi-strongly monotone with modulus μ. According to Theorem 3.9, iteration (2.2) converges at a linear rate if the step size meets the condition therein.

Our convergence and rates are given in term of the distance to the solution set X^*. In comparison, the results in the work [39] are given in terms of objective error under the assumption of a uniformly bounded $(x^k)_{k\geq 0}$. In addition, their step size decays like $O(\frac{1}{\tau\rho^\tau})$ for some $\rho > 1$ depending on τ, and our $O(\frac{1}{\tau})$ is better. Under similar assumptions, Bertsekas and Tsitsiklis [8, Section 7.5] also describes an algorithm for (2.1) and proves only subsequence convergence [8, Proposition 5.3] in $\mathbb{R}^n$.

2.3. Decentralized consensus optimization. Consider that m agents in a connected network solve the consensus problem of minimizing $\sum_{i=1}^m f_i(x)$, where $x \in \mathbb{R}^d$ is the shared variable and the convex differentiable function f_i is held privately by agent i. We assume that ∇f_i is L_i-Lipschitz continuous for all i. A decentralized gradient descent algorithm [41] can be developed based on the equivalent formulation

$$\underset{x_1,\ldots,x_m\in\mathbb{R}^d}{\text{minimize}}\ f(\mathbf{x}) := \textstyle\sum_{i=1}^m f_i(x_i), \quad \text{subject to } W\mathbf{x}=\mathbf{x}, \tag{2.3}$$

where $\mathbf{x} = (x_1, \ldots, x_m)^T \in \mathbb{R}^{m\times d}$ and $W \in \mathbb{R}^{m\times m}$ is the so-called mixing matrix satisfying: $W\mathbf{x} = \mathbf{x}$ if and only if $x_1 = \cdots = x_m$. For $i \neq j$, if $w_{i,j} \neq 0$, then agent i can communicate with agent j; otherwise they cannot. We assume that W is symmetric and doubly stochastic. Then, the decentralized consensus algorithm [41] can be expressed as $\mathbf{x}^{k+1} = W\mathbf{x}^k - \gamma\nabla f(\mathbf{x}^k) = \mathbf{x}^k - \gamma(\nabla f(\mathbf{x}^k) + \frac{1}{\gamma}(I-W)\mathbf{x}^k)$, where $\nabla f(\mathbf{x}) \in \mathbb{R}^{m\times d}$ is a matrix with its ith row equal to $(\nabla f_i(x_i))^T$; see [60]. The computation of $W\mathbf{x}^k$ involves communication between agents, and $\nabla f_i(x_i)$ is independently computed by each agent i. The iteration is equivalent to the gradient descent iteration applied to $\min_{\mathbf{x}} \sum_{i=1}^m f_i(x_i) + \frac{1}{2\gamma}\mathbf{x}^T(I-W)\mathbf{x}$. To apply our algorithm, we let $S := \frac{2}{L}\nabla F = \frac{2}{L}(\nabla f + \frac{1}{\gamma}(I-W))$ with $L = \max_i L_i + (1-\lambda_{\min}(W))/\gamma$, where $\lambda_{\min}(A)$ is the smallest eigenvalue of W. Computing $S_i\hat{\mathbf{x}}^k$ reduces to computing $\nabla f_i(\hat{x}_i^k)$ and the ith entry of $W\hat{\mathbf{x}}^k$ or $\sum_j w_{i,j}\hat{x}_j^k$, which involves only $\hat{x}_i^k$ and $\hat{x}_j^k$ from the neighbors of agent i. Note that since each agent i can store its own x_i locally, we have $\hat{x}_i^k \equiv x_i^k$.

If the agents are p independent Poisson processes and that each agent i has activation rate λ_i, then the probability that agent i activates before other agents is equal to $\frac{\lambda_i}{\sum_{i=1}^p \lambda_i}$ [43] and therefore our random sample scheme holds and ARock applies naturally. The algorithm is summarized as follows:

Algorithm 3: ARock for decentralized optimization (2.3)

```
Input : Each agent $i$ sets $x_i^0 \in \mathbb{R}^d$, $K > 0$.
while $k < K$ do
    when an agent $i$ is activated,
     $x_i^{k+1} = x_i^k - \frac{\eta_k}{L}(\nabla f_i(x_i^k) + \frac{1}{\gamma}(x_i^k - \sum_j w_{i,j}\hat{x}_j^k))$;
    increase the global counter $k \leftarrow k+1$;
```

2.4. Minimize smooth + nonsmooth functions. Consider the problem

$$\underset{x\in\mathcal{H}}{\text{minimize}}\, f(x) + g(x), \tag{2.4}$$

where f is closed proper convex and g is convex and L-Lipschitz differentiable with $L > 0$. Problems in the form of (2.4) arise in statistical regression, machine learning, and signal processing and include well-known problems such as the support vector machine, regularized least-squares, and regularized logistic regression. For any $x \in \mathcal{H}$ and scalar $\gamma \in (0, \frac{2}{L})$, define the proximal operator $\mathbf{prox}_f : \mathcal{H} \to \mathcal{H}$ and the reflective-proximal operator $\mathbf{refl}_f : \mathcal{H} \to \mathcal{H}$ as

$$\mathbf{prox}_{\gamma f}(x) := \underset{y\in\mathcal{H}}{\arg\min}\, f(y) + \frac{1}{2\gamma}\|y-x\|^2 \quad \text{and} \quad \mathbf{refl}_{\gamma f} := 2\mathbf{prox}_{\gamma f} - I_{\mathcal{H}}, \tag{2.5}$$

respectively, and define the following forward-backward operator $T_{\text{FBS}} := \mathbf{prox}_{\gamma f} \circ (I - \gamma\nabla g)$. Because $\mathbf{prox}_{\gamma f}$ is $\frac{1}{2}$-averaged and $(I - \gamma\nabla g)$ is $\frac{\gamma L}{2}$-averaged, T_{FBS} is α-averaged for $\alpha \in [\frac{2}{3}, 1)$ [6, Propositions 4.32 and 4.33]. Define $S := I - T_{\text{FBS}} = I - \mathbf{prox}_{\gamma f} \circ (I - \gamma\nabla g)$. When we apply Algorithm 1 to $T = T_{\text{FBS}}$ to solve (2.4), and assume f is separable in all coordinates, that is, $f(x) = \sum_{i=1}^m f_i(x_i)$, the update for the i_kth selected coordinate is

$$x_{i_k}^{k+1} = x_{i_k}^k - \eta_k\left(\hat{x}_{i_k}^k - \mathbf{prox}_{\gamma f_{i_k}}(\hat{x}_{i_k}^k - \gamma\nabla_{i_k} g(\hat{x}^k))\right), \tag{2.6}$$

Examples of separable functions include ℓ_1 norm, ℓ_2 norm square, the Huber function, and the indicator function of box constraints, i.e., $\{x|a_i \le x_i \le b_i,\ \forall i\}$. They all have simple **prox** maps. If $\eta_k \in [\eta_{\min}, \frac{1}{2\tau/\sqrt{m}+1})$, then the convergence is guaranteed by Theorem 3.7. To show linear convergence, we need to assume that $g(x)$ is strongly convex. Then, Proposition 2.2 below shows that $\mathbf{prox}_{\gamma f}\circ(I-\gamma\nabla g)$ is a quasi-contractive operator, and by Proposition 2.1, operator $I-\mathbf{prox}_{\gamma f}\circ(I-\gamma\nabla g)$ is quasi-strongly monotone. Finally, linear convergence and its rate follow from Theorem 3.9.

Proposition 2.2. *Assume that f is a closed proper convex function, and g is L-Lipschitz differentiable and strongly convex with modulus $\mu > 0$. Let $\gamma \in (0, \frac{2}{L})$. Then, both $I-\gamma\nabla g$ and $\mathbf{prox}_{\gamma f}\circ(I-\gamma\nabla g)$ are quasi-contractive operators.*

Proof. We first show that $I-\gamma\nabla g$ is a quasi-contractive operator. Note

$$\begin{aligned}
&\|(x-\gamma\nabla g(x))-(x^*-\gamma\nabla g(x^*))\|^2\\
=&\|x-x^*\|^2-2\gamma\langle x-x^*, \nabla g(x)-\nabla g(x^*)\rangle+\gamma^2\|\nabla g(x)-\nabla g(x^*)\|^2\\
\le&\|x-x^*\|^2-\gamma(2-\gamma L)\langle x-x^*, \nabla g(x)-\nabla g(x^*)\rangle\\
\le&(1-2\gamma\mu+\mu\gamma^2 L)\|x-x^*\|^2,
\end{aligned}$$

where the first inequality follows from the Baillon-Haddad theorem[§] and the second one from the strong convexity of g. Hence, $I-\gamma\nabla g$ is quasi-contractive if $0<\gamma<2/L$. Since f is convex, $\mathbf{prox}_{\gamma f}$ is firmly nonexpansive, and thus we immediately have the quasi-contractiveness of $\mathbf{prox}_{\gamma f}\circ(I-\gamma\nabla g)$ from that of $I-\gamma\nabla g$. ∎

2.5. Minimize nonsmooth + nonsmooth functions.

Consider

$$\underset{x\in\mathcal{H}}{\text{minimize}}\, f(x)+g(x), \tag{2.7}$$

where both $f(x)$ and $g(x)$ are closed proper convex and their **prox** maps are easy to compute. Define the Peaceman-Rachford [37] operator:

$$T_{\text{PRS}} := \mathbf{refl}_{\gamma f}\circ\mathbf{refl}_{\gamma g}.$$

Since both $\mathbf{refl}_{\gamma f}$ and $\mathbf{refl}_{\gamma g}$ are nonexpansive, their composition T_{PRS} is also nonexpansive. Let $S := I - T_{\text{PRS}}$. When applying ARock to $T = T_{\text{PRS}}$ to solve problem (2.7), the update (1.6) reduces to:

$$z^{k+1} = z^k - \eta_k U_{i_k}\circ\left(I-\mathbf{refl}_{\gamma f}\circ\mathbf{refl}_{\gamma g}\right)\hat{z}^k, \tag{2.8}$$

where we use z instead of x since the limit z^* of $(z^k)_{k\ge0}$ is not a solution to (2.7); instead, a solution must be recovered via $x^* = \mathbf{prox}_{\gamma g} z^*$. The convergence follows from Theorem 3.7 and that T_{PRS} is nonexpansive. If either f or g is strongly convex, then T_{PRS} is contractive and thus by Theorem 3.9, ARock converges linearly. Finer convergence rates follow from [18, 19]. A naive implementation of (2.8) is

$$\hat{x}^k = \mathbf{prox}_{\gamma g}(\hat{z}^k), \tag{2.9a}$$
$$\hat{y}^k = \mathbf{prox}_{\gamma f}(2\hat{x}^k-\hat{z}^k), \tag{2.9b}$$
$$z^{k+1} = z^k + 2\eta_k U_{i_k}(\hat{y}^k-\hat{x}^k), \tag{2.9c}$$

[§]Let g be a convex differentiable function. Then, ∇g is L-Lipschitz if and only if it is $\frac{1}{L}$-cocoercive.

where $\hat{x}^k$ and $\hat{y}^k$ are intermediate variables. Note that the order in which the proximal operators are applied to f and g affects both z^k [59] and whether coordinate-wise updates can be efficiently computed. Next, we present two special cases of (2.7) in Subsections 2.5.1 and 2.6 and discuss how to efficiently implement the update (2.9).

2.5.1. Feasibility problem. Suppose that $C_1, ..., C_m$ are closed convex subsets of $\mathcal{H}$ with a nonempty intersection. The problem is to find a point in the intersection. Let $\mathcal{I}_{C_i}$ be the indicator function of the set C_i, that is, $\mathcal{I}_{C_i}(x) = 0$ if $x \in C_i$ and ∞ otherwise. The feasibility problem can be formulated as the following

$$\underset{x=(x_1,\ldots,x_m)\in\mathcal{H}^m}{\text{minimize}} \quad \sum_{i=1}^m \mathcal{I}_{C_i}(x_i) + \mathcal{I}_{\{x_1=\cdots=x_m\}}(x).$$

Let $z^k = (z_1^k, \ldots, z_m^k) \in \mathcal{H}^m$, $\hat{z}^k = (\hat{z}_1^k, \ldots, \hat{z}_m^k) \in \mathcal{H}^m$, and $\hat{\bar{z}}^k \in \mathcal{H}$. We can implement (2.9) as follows (see Appendix A for the step-by-step derivation):

$$\hat{\bar{z}}^k = \tfrac{1}{m}\textstyle\sum_{i=1}^m \hat{z}_i^k, \tag{2.10a}$$

$$\hat{y}_{i_k}^k = \mathrm{Proj}_{C_{i_k}}\big(2\hat{\bar{z}}^k - \hat{z}_{i_k}^k\big), \tag{2.10b}$$

$$z_{i_k}^{k+1} = z_{i_k}^k + 2\eta_k(\hat{y}_{i_k}^k - \hat{\bar{z}}^k). \tag{2.10c}$$

The update (2.10) can be implemented as follows. Let global memory hold $z_1, \ldots, z_m$, as well as $\bar{z} = \frac{1}{m}\sum_{i=1}^m z_i$. At the kth update, an agent independently generates a random number $i_k \in \{1, \ldots, m\}$, then reads z_{i_k} as $\hat{z}_{i_k}^k$ and $\bar{z}$ as $\hat{\bar{z}}^k$, and finally computes $\hat{y}_{i_k}$ and updates z_{i_k} in global memory according to (2.10). Since $\bar{z}$ is maintained in global memory, the agent updates $\bar{z}$ according to $\bar{z}^{k+1} = \bar{z}^k + \frac{1}{m}(z_{i_k}^{k+1} - z_{i_k}^k)$. This implementation saves each agent from computing (2.10a) or reading all $z_1, \ldots, z_m$. Each agent only reads z_{i_k} and $\bar{z}$, executes (2.10b), and updates z_{i_k} (2.10c) and $\bar{z}$.

2.6. Async-parallel ADMM. This is another application of (2.9). Consider

$$\underset{x\in\mathcal{H}_1,\ y\in\mathcal{H}_2}{\text{minimize}} \quad f(x) + g(y) \quad \text{subject to } Ax + By = b, \tag{2.11}$$

where $\mathcal{H}_1$ and $\mathcal{H}_2$ are Hilbert spaces, A and B are bounded linear operators. We apply the update (2.9) to the Lagrange dual of (2.11) (see [27] for the derivation):

$$\underset{w\in\mathcal{G}}{\text{minimize}}\ d_f(w) + d_g(w), \tag{2.12}$$

where $d_f(w) := f^*(A^*w)$, $d_g(w) := g^*(B^*w) - \langle w, b\rangle$, and f^* and g^* denote the convex conjugates of f and g, respectively. The proximal maps induced by d_f and d_g can be computed via solving subproblems that involve only the original terms in (2.11): $z^+ = \mathbf{prox}_{\gamma d_f}(z)$ can be computed by (see Appendix A for the derivation)

$$\begin{cases} x^+ \in \arg\min_x f(x) - \langle z, Ax\rangle + \frac{\gamma}{2}\|Ax\|^2, \\ z^+ = z - \gamma Ax^+, \end{cases} \tag{2.13}$$

and $z^+ = \mathbf{prox}_{\gamma d_g}(z)$ by

$$\begin{cases} y^+ \in \arg\min_y g(y) - \langle z, By - b\rangle + \frac{\gamma}{2}\|By - b\|^2, \\ z^+ = z - \gamma(By^+ - b). \end{cases} \tag{2.14}$$

Plugging (2.13) and (2.14) into (2.9) yields the following naive implementation

$$\hat{y}^k \in \arg\min_y g(y) - \langle \hat{z}^k, By - b\rangle + \frac{\gamma}{2}\|By-b\|^2, \tag{2.15a}$$

$$\hat{w}_g^k = \hat{z}^k - \gamma(B\hat{y}^k - b), \tag{2.15b}$$

$$\hat{x}^k \in \arg\min_x f(x) - \langle 2\hat{w}_g^k - \hat{z}^k, Ax\rangle + \frac{\gamma}{2}\|Ax\|^2, \tag{2.15c}$$

$$\hat{w}_f^k = 2\hat{w}_g^k - \hat{z}^k - \gamma A\hat{x}^k, \tag{2.15d}$$

$$z_{i_k}^{k+1} = z_{i_k}^k + \eta_k(\hat{w}_{f,i_k}^k - \hat{w}_{g,i_k}^k). \tag{2.15e}$$

Note that $2\eta_k$ in (2.9c) becomes η_k in (2.15e) because ADMM is equivalent to the Douglas-Rachford operator, which is the average of the Peaceman-Rachford operator and the identity operator [37]. Under favorable structures, (2.15) can be implemented efficiently. For instance, when A and B are block diagonal matrices and f, g are corresponding block separable functions, steps (2.15a)–(2.15d) reduce to independent computation for each i. Since only $\hat{w}_{f,i_k}^k$ and $\hat{w}_{g,i_k}^k$ are needed to update the main variable z^k, we only need to compute (2.15a)–(2.15d) for the i_kth block. This is exploited in distributed and decentralized ADMM in the next two subsections.

2.6.1. Async-parallel ADMM for consensus optimization. Consider the consensus optimization problem:

$$\underset{x_i, y\in\mathcal{H}}{\text{minimize}} \ \sum_{i=1}^m f_i(x_i) \quad \text{subject to } x_i - y = 0, \quad \forall i = 1,...,m, \tag{2.16}$$

where $f_i(x_i)$ are proper close convex functions. Rewrite (2.16) to the ADMM form:

$$\begin{aligned} &\underset{x_i, y\in\mathcal{H}}{\text{minimize}} \quad \sum_{i=1}^m f_i(x_i) + g(y) \\ &\text{subject to} \quad \begin{bmatrix} I_{\mathcal{H}} & 0 & \cdots & 0 \\ 0 & I_{\mathcal{H}} & \cdots & 0 \\ & & \ddots & \\ 0 & 0 & \cdots & I_{\mathcal{H}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} - \begin{bmatrix} I_{\mathcal{H}} \\ I_{\mathcal{H}} \\ \vdots \\ I_{\mathcal{H}} \end{bmatrix} y = 0, \end{aligned} \tag{2.17}$$

where $g = 0$. Now apply the async-parallel ADMM (2.15) to (2.17) with dual variables $z_1, ..., z_m \in \mathcal{H}$. In particular, the update (2.15a), (2.15b), (2.15c), (2.15d) reduce to

$$\begin{aligned} \hat{y}^k &= \arg\min_y \left\{ \textstyle\sum_{i=1}^m \langle \hat{z}_i^k, y\rangle + \frac{\gamma m}{2}\|y\|^2 \right\} = -\frac{1}{\gamma m}\textstyle\sum_{i=1}^m \hat{z}_i^k \\ (\hat{w}_{d_g}^k)_i &= \hat{z}_i^k + \gamma\, \hat{y}^k \\ \hat{x}_i^k &= \arg\min_{x_i} \left\{ f_i(x_i) - \langle 2(\hat{w}_{d_g}^k)_i - \hat{z}_i^k, x_i\rangle + \tfrac{\gamma}{2}\|x_i\|^2 \right\}, \\ (\hat{w}_{d_f}^k)_i &= 2(\hat{w}_{d_g}^k)_i - \hat{z}_i^k - \gamma\, \hat{x}_i^k \end{aligned} \tag{2.18}$$

Therefore, we obtain the following async-parallel ADMM algorithm for the problem (2.16). This algorithm applies to all the distributed applications in [11].

Algorithm 4: ARock for consensus optimization

```
Input : set shared variables $y^0, z_i^0$, $\forall i$, and $K > 0$.
while $k < K$ every agent asynchronously do
    choose $i_k$ from $\{1, ..., m\}$ with equal probability;
    evaluate $(\hat{w}_{d_g}^k)_{i_k}$, $\hat{x}_{i_k}^k$, and $(\hat{w}_{d_f}^k)_{i_k}$ following (2.18);
    update $z_{i_k}^{k+1} = z_{i_k}^k + \eta_k\,((\hat{w}_{d_f}^k)_{i_k} - (\hat{w}_{d_g}^k)_{i_k})$;
    update $y^{k+1} = y^k + \frac{1}{\gamma m}(z_{i_k}^k - z_{i_k}^{k+1})$;
    update the global counter $k \leftarrow k+1$;
```

2.6.2. Async-parallel ADMM for decentralized optimization. Let $V = \{1, ..., m\}$ be a set of agents and $E = \{(i,j) \mid \text{if agent } i \text{ connects to agent } j, i < j\}$ be the set of undirected links between the agents. Consider the following decentralized consensus optimization problem on the graph $G = (V, E)$:

$$\underset{x_1,\dots,x_m\in\mathbb{R}^d}{\text{minimize}}\ f(x_1,\dots,x_m) := \textstyle\sum_{i=1}^m f_i(x_i), \quad \text{subject to } x_i = x_j,\ \forall (i,j)\in E, \tag{2.19}$$

where $x_1, ..., x_m \in \mathbb{R}^d$ are the local variables and each agent can only communicate with its neighbors in G. By introducing the auxiliary variable y_{ij} associated with each edge $(i,j) \in E$, the problem (2.19) can be reformulated as:

$$\underset{x_i, y_{ij}}{\text{minimize}}\ \textstyle\sum_{i=1}^m f_i(x_i), \quad \text{subject to } x_i = y_{ij},\ x_j = y_{ij}, \quad \forall (i,j)\in E. \tag{2.20}$$

Define $x = (x_1, ..., x_m)^T$ and $y = (y_{ij})_{(i,j)\in E} \in \mathbb{R}^{|E|d}$ to rewrite (2.20) as

$$\underset{x,y}{\text{minimize}}\ \textstyle\sum_{i=1}^m f_i(x_i), \quad \text{subject to } Ax + By = 0, \tag{2.21}$$

for proper matrices A and B. Applying the async-parallel ADMM (2.15) to (2.21) gives rise to the following simplified update: Let $E(i)$ be the set of edges connected with agent i and $|E(i)|$ be its cardinality. Let $L(i) = \{j \mid (j,i) \in E(i), j < i\}$ and $R(i) = \{j \mid (i,j) \in E(i), j > i\}$. To every pair of constraints $x_i = y_{ij}$ and $x_j = y_{ij}$, $(i,j) \in E$, we associate the dual variables $z_{ij,i}$ and $z_{ij,j}$, respectively. Whenever some agent i is activated, it calculates

$$\hat{x}_i^k = \arg\min_{x_i} f_i(x_i) + \Big(\sum_{l\in L(i)} \hat{z}_{li,l}^k + \sum_{r\in R(i)} \hat{z}_{ir,r}^k\Big)x_i + \frac{\gamma}{2}|E(i)|\cdot\|x_i\|^2, \tag{2.22a}$$

$$z_{li,i}^{k+1} = z_{li,i}^k - \eta_k((\hat{z}_{li,i}^k + \hat{z}_{li,l})/2 + \gamma\hat{x}_i^k), \quad \forall l \in L(i), \tag{2.22b}$$

$$z_{ir,i}^{k+1} = z_{ir,i}^k - \eta_k((\hat{z}_{ir,i}^k + \hat{z}_{ir,r})/2 + \gamma\hat{x}_i^k), \quad \forall r \in R(i). \tag{2.22c}$$

We present the algorithm based on (2.22) for problem (2.19) in Algorithm 5.

Algorithm 5: ARock for the decentralized problem (2.20)

```
Input : Each agent $i$ sets the dual variables $z_{e,i}^0 = 0$ for $e \in E(i)$, $K > 0$.
while $k < K$, any activated agent $i$ do
    (previously received $\hat{z}_{li,l}^k$ from neighbors $l \in L(i)$ and $\hat{z}_{ir,r}^k$ from $r \in R(i)$);
    update $\hat{x}_i^k$ according to (2.22a);
    update $z_{li,i}^{k+1}$ and $z_{ir,i}^{k+1}$ according to (2.22b) and (2.22c), respectively;
    send $z_{li,i}^{k+1}$ to neighbors $l \in L(i)$ and $z_{ir,i}^{k+1}$ to neighbors $r \in R(i)$;
```

Algorithm 5 activates one agent at each iteration and updates all the dual variables associated with the agent. In this case, only one-sided communication is needed, for sending the updated dual variables in the last step. We allow this communication to be delayed in the sense that agent i's neighbors may be activated and start their computation before receiving the latest dual variables from agent i.

Our algorithm is different from the asynchronous ADMM algorithm by Wei and Ozdaglar [58]. Their algorithm activates an edge and its two associated agents at each iteration and thus requires two-sided communication at each activation. We can recover their algorithm as a special case by activating an edge $(i,j)\in E$ and its associated agents i and j at each iteration, updating the dual variables $z_{ij,i}$ and $z_{ij,j}$ associated with the edge, as well as computing the intermediate variables x_i, x_j, and y_{ij}. The updates are derived from (2.21) with the orders of x and y swapped. Note that [58] does not consider the situation that adjacent edges are activated in a short period of time, which may cause overlapped computation and delay communication. Indeed, their algorithm corresponds to $\tau=0$ and the corresponding stepsize $\eta_k\equiv 1$. Appendix B presents the steps to derive the algorithms in this subsection.

3. Convergence.

We establish weak and strong convergence in Subsection 3.1 and linear convergence in Subsection 3.2. Step size selection is also discussed.

3.1. Almost sure convergence.

ASSUMPTION 1. *Throughout our analysis, we assume* $p_{\min}:=\min_i p_i>0$ *and*

$$\mathrm{Prob}(i_k=i\,|\,\mathcal{X}^k)=\mathrm{Prob}(i_k=i)=p_i,\quad \forall i,k. \tag{3.1}$$

We let $|J(k)|$ be the number of elements in $J(k)$ (see Subsection 1.2). Only for the purpose of analysis, we define the (never computed) full update at kth iteration:

$$\bar{x}^{k+1}:=x^k-\eta_k S\hat{x}^k. \tag{3.2}$$

Lemma 3.1 below shows that T is nonexpansive if and only if S is 1/2-cocoercive.

LEMMA 3.1. *Operator* $T:\mathcal{H}\to\mathcal{H}$ *is nonexpansive if and only if* $S=I-T$ *is 1/2-cocoercive, i.e.,* $\langle x-y,Sx-Sy\rangle\geq\frac{1}{2}\|Sx-Sy\|^2,\forall\, x,y\in\mathcal{H}$.

Proof. See textbook [6, Proposition 4.33] for the proof of the "if" part, and the "only if" part, though missing there, follows by just reversing the proof. ☐

The lemma below develops an upper bound for the expected distance between x^{k+1} and any $x^*\in\operatorname{Fix}T$.

LEMMA 3.2. *Let* $(x^k)_{k\geq 0}$ *be the sequence generated by Algorithm 1. Then for any* $x^*\in\operatorname{Fix}T$ *and* $\gamma>0$ *(to be optimized later), we have*

$$\begin{aligned}\mathbb{E}\big(\|x^{k+1}-x^*\|^2\,\big|\,\mathcal{X}^k\big)\leq&\|x^k-x^*\|^2+\tfrac{\gamma}{m}\textstyle\sum_{d\in J(k)}\|x^d-x^{d+1}\|^2\\&+\tfrac{1}{m}\left(\tfrac{|J(k)|}{\gamma}+\tfrac{1}{mp_{\min}}-\tfrac{1}{\eta_k}\right)\|x^k-\bar{x}^{k+1}\|^2.\end{aligned} \tag{3.3}$$

Proof. Recall $\mathrm{Prob}(i_k=i\,|\,\mathcal{X}^k)=p_i$. Then we have

$$\begin{aligned}&\mathbb{E}\left(\|x^{k+1}-x^*\|^2\,|\,\mathcal{X}^k\right)\\ \overset{(1.3)}{=}\;&\mathbb{E}\left(\|x^k-\tfrac{\eta_k}{mp_{i_k}}S_{i_k}\hat{x}^k-x^*\|^2\,|\,\mathcal{X}^k\right)\\ =\;&\|x^k-x^*\|^2+\mathbb{E}\left(\tfrac{2\eta_k}{mp_{i_k}}\left\langle S_{i_k}\hat{x}^k,x^*-x^k\right\rangle+\tfrac{\eta_k^2}{m^2p_{i_k}^2}\|S_{i_k}\hat{x}^k\|^2\,\Big|\,\mathcal{X}^k\right)\\ =\;&\|x^k-x^*\|^2+\tfrac{2\eta_k}{m}\textstyle\sum_{i=1}^m\left\langle S_i\hat{x}^k,x^*-x^k\right\rangle+\tfrac{\eta_k^2}{m^2}\sum_{i=1}^m\tfrac{1}{p_i}\|S_i\hat{x}^k\|^2\\ =\;&\|x^k-x^*\|^2+\tfrac{2\eta_k}{m}\left\langle S\hat{x}^k,x^*-x^k\right\rangle+\tfrac{\eta_k^2}{m^2}\textstyle\sum_{i=1}^m\tfrac{1}{p_i}\|S_i\hat{x}^k\|^2.\end{aligned} \tag{3.4}$$

Note that

$$\sum_{i=1}^{m}\frac{1}{p_i}\|S_i\hat{x}^k\|^2 \le \frac{1}{p_{\min}}\sum_{i=1}^{m}\|S_i\hat{x}^k\|^2 = \frac{1}{p_{\min}}\|S\hat{x}^k\|^2 \overset{(3.2)}{=} \frac{1}{\eta_k^2 p_{\min}}\|x^k-\bar{x}^{k+1}\|^2, \tag{3.5}$$

and

$$\begin{aligned}
&\langle S\hat{x}^k, x^*-x^k\rangle\\
\overset{(1.5)}{=}&\langle S\hat{x}^k, x^*-\hat{x}^k+\textstyle\sum_{d\in J(k)}(x^d-x^{d+1})\rangle\\
\overset{(3.2)}{=}&\langle S\hat{x}^k, x^*-\hat{x}^k\rangle+\tfrac{1}{\eta_k}\textstyle\sum_{d\in J(k)}\langle x^k-\bar{x}^{k+1}, x^d-x^{d+1}\rangle\\
\le&\langle S\hat{x}^k-Sx^*, x^*-\hat{x}^k\rangle+\tfrac{1}{2\eta_k}\textstyle\sum_{d\in J(k)}\left(\tfrac{1}{\gamma}\|x^k-\bar{x}^{k+1}\|^2+\gamma\|x^d-x^{d+1}\|^2\right)\\
\le&-\tfrac{1}{2}\|S\hat{x}^k\|^2+\tfrac{1}{2\eta_k}\textstyle\sum_{d\in J(k)}\left(\tfrac{1}{\gamma}\|x^k-\bar{x}^{k+1}\|^2+\gamma\|x^d-x^{d+1}\|^2\right)\\
\overset{(3.2)}{=}&-\tfrac{1}{2\eta_k^2}\|x^k-\bar{x}^{k+1}\|^2+\tfrac{|J(k)|}{2\gamma\eta_k}\|x^k-\bar{x}^{k+1}\|^2+\tfrac{\gamma}{2\eta_k}\textstyle\sum_{d\in J(k)}\|x^d-x^{d+1}\|^2,
\end{aligned} \tag{3.6}$$

where the first inequality follows from the Young's inequality. Plugging (3.5) and (3.6) into (3.4) gives the desired result. ∎

We need the following lemma on *nonnegative almost supermartingales* [52].

Lemma 3.3 ([52, Theorem 1]). *Let $\mathscr{F}=(\mathcal{F}^k)_{k\ge 0}$ be a sequence of sub-sigma algebras of $\mathcal{F}$ such that $\forall k\ge 0$, $\mathcal{F}^k\subset\mathcal{F}^{k+1}$. Define $\ell_+(\mathscr{F})$ as the set of sequences of $[0,+\infty)$-valued random variables $(\xi_k)_{k\ge0}$, where ξ_k is $\mathcal{F}^k$ measurable, and $\ell_+^1(\mathscr{F}):=\{(\xi_k)_{k\ge0}\in\ell_+(\mathscr{F})\,|\,\sum_k\xi_k<+\infty \text{ a.s.}\}$. Let $(\alpha_k)_{k\ge0},(v_k)_{k\ge0}\in\ell_+(\mathscr{F})$, and $(\eta_k)_{k\ge0},(\xi_k)_{k\ge0}\in\ell_+^1(\mathscr{F})$ be such that*

$$\mathbb{E}(\alpha_{k+1}|\mathcal{F}^k)+v_k\le(1+\xi_k)\alpha_k+\eta_k.$$

Then $(v_k)_{k\ge0}\in\ell_+^1(\mathscr{F})$ and α_k converges to a $[0,+\infty)$-valued random variable a.s..

Let $\mathcal{H}^{\tau+1}=\prod_{i=0}^{\tau}\mathcal{H}$ be a product space and $\langle\cdot\,|\,\cdot\rangle$ be the induced inner product:

$$\big\langle(z^0,\dots,z^\tau)\,|\,(y^0,\dots,y^\tau)\big\rangle=\sum_{i=0}^{\tau}\langle z^i,y^i\rangle,\quad\forall(z^0,\dots,z^\tau),(y^0,\dots,y^\tau)\in\mathcal{H}^{\tau+1}.$$

Let M' be a symmetric $(\tau+1)\times(\tau+1)$ tri-diagonal matrix with its main diagonal as $\sqrt{p_{\min}}[\frac{1}{\sqrt{p_{\min}}}+\tau,2\tau-1,2\tau-3,\dots,1]$ and first off-diagonal as $-\sqrt{p_{\min}}[\tau,\tau-1,\dots,1]$, and let $M=M'\otimes I_{\mathcal{H}}$. Here $\otimes$ represents the Kronecker product. For a given $(y^0,\cdots,y^\tau)\in\mathcal{H}^{\tau+1}$, $(z^0,\cdots,z^\tau)=M(y^0,\cdots,y^\tau)$ is given by:

$$\begin{aligned}
z^0&=y^0+\sqrt{p_{\min}}(y^0-y^1),\\
z^i&=\sqrt{p_{\min}}\left((i-\tau-1)y^{i-1}+(2\tau-2i+1)y^i+(i-\tau)y^{i+1}\right),\ \text{ if } 1\le i\le\tau-1,\\
z^\tau&=\sqrt{p_{\min}}(y^\tau-y^{\tau-1}).
\end{aligned}$$

Then M is a self-adjoint and positive definite linear operator since M' is symmetric and positive definite, and we define $\langle\cdot\,|\,\cdot\rangle_M=\langle\cdot\,|\,M\cdot\rangle$ as the M-weighted inner product and $\|\cdot\|_M$ the induced norm. Let

$$\mathbf{x}^k=(x^k,x^{k-1},\dots,x^{k-\tau})\in\mathcal{H}^{\tau+1},\ k\ge0,\text{ and }\mathbf{x}^*=(x^*,x^*,\dots,x^*)\in\mathbf{X}^*\subseteq\mathcal{H}^{\tau+1},$$

where we set $x^k=x^0$ for $k<0$. With

$$\xi_k(\mathbf{x}^*):=\|\mathbf{x}^k-\mathbf{x}^*\|_M^2=\|x^k-x^*\|^2+\sqrt{p_{\min}}\textstyle\sum_{i=k-\tau}^{k-1}\left(i-(k-\tau)+1\right)\|x^i-x^{i+1}\|^2, \tag{3.7}$$

we have the following fundamental inequality:

THEOREM 3.4 (Fundamental inequality). *Let $(x^k)_{k\geq 0}$ be the sequence generated by ARock. Then for any $\mathbf{x}^* \in \mathbf{X}^*$, it holds that*

$$\mathbb{E}\left(\xi_{k+1}(\mathbf{x}^*) \,\middle|\, \mathcal{X}^k\right) + \frac{1}{m}\left(\frac{1}{\eta_k} - \frac{2\tau}{m\sqrt{p_{\min}}} - \frac{1}{mp_{\min}}\right)\|\bar{x}^{k+1} - x^k\|^2 \leq \xi_k(\mathbf{x}^*). \tag{3.8}$$

Proof. Let $\gamma = m\sqrt{p_{\min}}$. Since $J(k) \subset \{k-1, \cdots, k-\tau\}$, then (3.3) indicates

$$\begin{aligned}\mathbb{E}\big(\|x^{k+1} - x^*\|^2 \,\big|\, \mathcal{X}^k\big) \leq &\|x^k - x^*\|^2 + \tfrac{1}{\sqrt{p_{\min}}}\textstyle\sum_{i=k-\tau}^{k-1}\|x^i - x^{i+1}\|^2 \\ &+ \tfrac{1}{m}\left(\tfrac{\tau}{m\sqrt{p_{\min}}} + \tfrac{1}{mp_{\min}} - \tfrac{1}{\eta_k}\right)\|x^k - \bar{x}^{k+1}\|^2.\end{aligned} \tag{3.9}$$

From (1.3) and (3.2), it is easy to have $\mathbb{E}(\|x^k - x^{k+1}\|^2|\mathcal{X}^k) \leq \frac{1}{m^2p_{\min}}\|x^k - \bar{x}^{k+1}\|^2$, which together with (3.9) implies (3.8) by using the definition of $\xi_k(\mathbf{x}^*)$. ∎

REMARK 1 (Stochastic Fejér monotonicity). *From (3.8), if $0 < \eta_k \leq \frac{mp_{\min}}{2\tau\sqrt{p_{\min}}+1}$, then we have $\mathbb{E}(\|\mathbf{x}^{k+1} - \mathbf{x}^*\|_M^2|\mathcal{X}^k) \leq \|\mathbf{x}^k - \mathbf{x}^*\|_M^2$, $\forall \mathbf{x}^* \in \mathbf{X}^*$.*

REMARK 2. *Let us check our step size bound $\frac{mp_{\min}}{2\tau\sqrt{p_{\min}}+1}$. Consider the uniform case: $p_{\min} \equiv p_i \equiv \frac{1}{m}$. Then, the bound simplifies to $\frac{1}{1+2\tau/\sqrt{m}}$. If the maximal delay is no more than the square root of the number of coordinates, i.e., $\tau = O(\sqrt{m})$, then the bound is $O(1)$. In general, τ depends on several factors such as problem structure, system architecture, load balance, etc. If all updates and agents are identical, then τ is proportional to p, the number of agents. Hence, ARock takes an $O(1)$ step size for solving a problem with m coordinates by $p = \sqrt{m}$ agents under balanced loads.*

The next lemma is a direct consequence of the invertibility of the metric M.

LEMMA 3.5. *A sequence $(\mathbf{z}^k)_{k\geq 0} \subset \mathcal{H}^{\tau+1}$ (weakly) converges to $\mathbf{z} \in \mathcal{H}^{\tau+1}$ under the metric $\langle\cdot\,|\,\cdot\rangle$ if and only if it does so under the metric $\langle\cdot\,|\,\cdot\rangle_M$.*

In light of Lemma 3.5, the metric of the inner product for weak convergence in the next lemma is not specified. The lemma and its proof are adapted from [15].

LEMMA 3.6. *Let $(x^k)_{k\geq 0} \subset \mathcal{H}$ be the sequence generated by ARock with $\eta_k \in [\eta_{\min}, \frac{cmp_{\min}}{2\tau\sqrt{p_{\min}}+1}]$ for any $\eta_{\min} > 0$ and $0 < c < 1$. Then we have:*

(i) *$\sum_{k=0}^{\infty}\|x^k - \bar{x}^{k+1}\|^2 < \infty$ a.s..*

(ii) *$x^k - x^{k+1} \to 0$ a.s. and $\hat{x}^k - x^{k+1} \to 0$ a.s..*

(iii) *The sequence $(\mathbf{x}^k)_{k\geq 0} \subset \mathcal{H}^{\tau+1}$ is bounded a.s..*

(iv) *There exists $\tilde{\Omega} \in \mathcal{F}$ such that $P(\tilde{\Omega}) = 1$ and, for every $\omega \in \tilde{\Omega}$ and every $\mathbf{x}^* \in \mathbf{X}^*$, $(\|\mathbf{x}^k(\omega) - \mathbf{x}^*\|_M)_{k\geq 0}$ converges.*

(v) *Let $\mathscr{Z}(\mathbf{x}^k)$ be the set of weakly convergent cluster points of $(\mathbf{x}^k)_{k\geq 0}$. Then, $\mathscr{Z}(\mathbf{x}^k) \subseteq \mathbf{X}^*$ a.s..*

Proof. (i): Applying Lemma 3.3 with $\xi_k = \eta_k = 0$ and $\alpha_k = \xi_k(\mathbf{x}^*)$, $\forall k$ to (3.8) and noting $\inf_k\left(\frac{1}{\eta_k} - \frac{2\tau}{m\sqrt{p_{\min}}} - \frac{1}{mp_{\min}}\right) > 0$ gives this result directly.

(ii) From (i), we have $x^k - \bar{x}^{k+1} \to 0$ a.s.. Since $\|x^k - x^{k+1}\| \leq \frac{1}{mp_{\min}}\|x^k - \bar{x}^{k+1}\|$, we have $x^k - x^{k+1} \to 0$ a.s.. Then from (1.5), we have $\hat{x}^k - x^k \to 0$ a.s..

(iii): From Lemma 3.3, we have that $(\|\mathbf{x}^k - \mathbf{x}^*\|_M^2)_{k\geq 0}$ converges a.s. and so does $(\|\mathbf{x}^k - \mathbf{x}^*\|_M)_{k\geq 0}$, i.e., $\lim_{k\to\infty}\|\mathbf{x}^k - \mathbf{x}^*\|_M = \gamma$ a.s., where γ is a $[0, +\infty)$-valued random variable. Hence, $(\|\mathbf{x}^k - \mathbf{x}^*\|_M)_{k\geq 0}$ must be bounded a.s. and so is $(\mathbf{x}^k)_{k\geq 0}$.

(iv): The proof follows directly from [15, Proposition 2.3 (iii)]. It is worth noting that $\tilde{\Omega}$ in the statement works for all $\mathbf{x}^* \in \mathbf{X}^*$, namely, $\tilde{\Omega}$ does not depend on $\mathbf{x}^*$.

(v): By (ii), there exists $\hat{\Omega} \in \mathcal{F}$ such that $P(\hat{\Omega}) = 1$ and

$$x^k(w) - x^{k+1}(w) \to 0, \quad \forall w \in \hat{\Omega}. \tag{3.10}$$

For any $\omega \in \hat{\Omega}$, let $(\mathbf{x}^{k_n}(\omega))_{n\geq 0}$ be a weakly convergent subsequence of $(\mathbf{x}^k(\omega))_{k\geq 0}$, i.e., $\mathbf{x}^{k_n}(\omega) \rightharpoonup \mathbf{x}$, where $\mathbf{x}^{k_n}(\omega) = (x^{k_n}(\omega), x^{k_n-1}(\omega)..., x^{k_n-\tau}(\omega))$ and $\mathbf{x} = (u^0, ..., u^\tau)$. Note that $\mathbf{x}^{k_n}(\omega) \rightharpoonup \mathbf{x}$ implies $x^{k_n-j}(\omega) \rightharpoonup u^j$, $\forall j$. Therefore, $u^i = u^j$, for any $i, j \in \{0, \cdots, \tau\}$ because $x^{k_n-i}(\omega) - x^{k_n-j}(\omega) \to 0$.

Furthermore, observing $\eta_k \geq \eta_{\min} > 0$, we have

$$\lim_{n\to\infty} \hat{x}^{k_n}(\omega) - T\hat{x}^{k_n}(\omega) = \lim_{n\to\infty} S\hat{x}^{k_n}(\omega) = \lim_{n\to\infty} \frac{1}{\eta_{k_n}}(x^{k_n}(\omega) - \bar{x}^{k_n+1}(\omega)) = 0. \tag{3.11}$$

From the triangle inequality and the nonexpansiveness of T, it follows that

$$\begin{aligned}
&\|x^{k_n}(\omega) - Tx^{k_n}(\omega)\| \\
=&\|x^{k_n}(\omega) - \hat{x}^{k_n}(\omega) + \hat{x}^{k_n}(\omega) - T\hat{x}^{k_n}(\omega) + T\hat{x}^{k_n}(\omega) - Tx^{k_n}(\omega)\| \\
\leq&\|x^{k_n}(\omega) - \hat{x}^{k_n}(\omega)\| + \|\hat{x}^{k_n}(\omega) - T\hat{x}^{k_n}(\omega)\| + \|T\hat{x}^{k_n}(\omega) - Tx^{k_n}(\omega)\| \\
\leq&2\|x^{k_n}(\omega) - \hat{x}^{k_n}(\omega)\| + \|\hat{x}^{k_n}(\omega) - T\hat{x}^{k_n}(\omega)\| \\
\leq&2\textstyle\sum_{d\in J(k_n)} \|x^d(\omega) - x^{d+1}(\omega)\| + \|\hat{x}^{k_n}(\omega) - T\hat{x}^{k_n}(\omega)\|.
\end{aligned}$$

From (3.10), (3.11), and the above inequality, it follows $\lim_{n\to\infty} x^{k_n}(\omega) - Tx^{k_n}(\omega) = 0$. Finally, the demiclosedness principle [6, Theorem 4.17] implies $u^0 \in \operatorname{Fix} T$. □

Theorem 3.7. *Under the assumptions of Lemma 3.6, the sequence $(\mathbf{x}^k)_{k\geq 0}$ weakly converges to an $\mathbf{X}^*$-valued random variable a.s.. In addition, if T is demicompact at 0, $(\mathbf{x}^k)_{k\geq 0}$ strongly converges to an $\mathbf{X}^*$-valued random variable a.s..*

Proof. The proof for a.s. weak convergence follows from Opial's Lemma [49, 44] and Lemma 3.6 (iv)-(v). Next we assume that T is demicompact at 0. From the proof of Lemma 3.6 (v), there is $\hat{\Omega} \in \mathcal{F}$ such that $P(\hat{\Omega}) = 1$ and, for any $w \in \hat{\Omega}$ and any weakly convergent subsequence of $(\mathbf{x}^{k_n}(w))_{n\geq 0}$, $\lim_{n\to\infty} x^{k_n}(w) - Tx^{k_n}(w) = 0$. Since T is demicompact, $(x^{k_n}(w))_{n\geq 0}$ has a strongly convergent subsequence, for which we still use $(x^{k_n}(w))_{n\geq 0}$. Hence, $x^{k_n}(w) \to \bar{x}(w) \in \operatorname{Fix} T$. Lemma 3.6 (ii) yields $\mathbf{x}^{k_n}(w) \to \bar{\mathbf{x}}(w) \in \mathbf{X}^*$. Then by Lemma 3.6 (iv), there is $\tilde{\Omega} \in \mathcal{F}$ such that $P(\tilde{\Omega}) = 1$ and, for every $w \in \tilde{\Omega}$ and every $\mathbf{x}^* \in \mathbf{X}^*$, $(\|\mathbf{x}^k(w) - \mathbf{x}^*\|_M)_{k\geq 0}$ converges. Thus, for any $w \in \hat{\Omega} \cap \tilde{\Omega}$, we have $\lim_{k\to\infty} \|\mathbf{x}^k(w) - \bar{\mathbf{x}}(w)\|_M = 0$. Because $P(\hat{\Omega} \cap \tilde{\Omega}) = 1$, we conclude that $(\mathbf{x}^k)_{k\geq 0}$ strongly converges to an $\mathbf{X}^*$-valued random variable a.s.. □

Remark 3. *For the generalization in Section 1.3, we need to replace (3.5) by*

$$\textstyle\sum_{i=1}^m \frac{1}{p_i}\|U_i \circ S\hat{x}^k\|^2 \leq \frac{1}{p_{\min}} \sum_{i=1}^m \|U_i \circ S\hat{x}^k\|^2 \leq \frac{C}{p_{\min}}\|S\hat{x}^k\|^2 = \frac{C}{\eta_k^2 p_{\min}}\|x^k - \bar{x}^{k+1}\|^2,$$

and update the step size condition to $\eta_k \in [\eta_{\min}, \frac{cmp_{\min}}{2\tau\sqrt{p_{\min}}+C}]$. Then the proofs of Theorem 3.7 and Lemma 3.6 will go through and yield the same convergence result.

3.2. Linear convergence.

In this section, we establish linear convergence under the assumption that S is quasi-strongly monotone. We first present a key lemma.

Lemma 3.8. *Assume that the step size is fixed, i.e., $\eta_k = \eta$, and satisfies*

$$0 < \eta \leq \underline{\eta}_1 := (1 - \tfrac{1}{\rho})\tfrac{m\sqrt{p_{\min}}}{8} \tfrac{\rho^{1/2}-1}{\rho^{(\tau+1)/2}-1} \tag{3.12}$$

for some $\rho > 1$. Then we have, for all $k \geq 1$,

$$\mathbb{E}\|\bar{x}^k - x^{k-1}\|^2 \leq \rho\mathbb{E}\|\bar{x}^{k+1} - x^k\|^2. \tag{3.13}$$

Proof. We prove (3.13) by induction. First, based on the inequality $\|a\|^2-\|b\|^2 \le 2\|a\|\|b-a\|$ we observe that, for any $k\ge 1$,

$$\begin{aligned}\|\bar{x}^k-x^{k-1}\|^2-\|\bar{x}^{k+1}-x^k\|^2 &\le 2\|\bar{x}^k-x^{k-1}\|\|\bar{x}^{k+1}-x^k-\bar{x}^k+x^{k-1}\|\\ &=2\|\bar{x}^k-x^{k-1}\|\|\eta S(\hat{x}^k)-\eta S(\hat{x}^{k-1})\|\\ &\le 4\eta\|\bar{x}^k-x^{k-1}\|\|\hat{x}^k-\hat{x}^{k-1}\|. \end{aligned}\tag{3.14}$$

Applying the triangle inequality and (1.5) yields

$$\begin{aligned}\|\hat{x}^k-\hat{x}^{k-1}\| &\le \|x^k-\hat{x}^k\|+\|x^k-x^{k-1}\|+\|x^{k-1}-\hat{x}^{k-1}\|\\ &\le \sum_{d\in J(k)}\|x^d-x^{d+1}\|+\|x^k-x^{k-1}\|+\sum_{d\in J(k-1)}\|x^d-x^{d+1}\|\\ &\le 2\textstyle\sum_{t=0}^{\tau}\|x^{k-t}-x^{k-t-1}\|.\end{aligned}\tag{3.15}$$

For the basic case, we have $\hat{x}^0=x^0$, $\hat{x}^1\in\{x^0,x^1\}$. Letting $k=1$ in (3.14) gets us

$$\begin{aligned}\mathbb{E}\|\bar{x}^1-x^0\|^2-\mathbb{E}\|\bar{x}^2-x^1\|^2 &\le 4\eta\mathbb{E}\|\bar{x}^1-x^0\|\|x^1-x^0\|\\ &\le 2\eta\big(\tfrac{1}{m\sqrt{p_{\min}}}\mathbb{E}\|\bar{x}^1-x^0\|^2+m\sqrt{p_{\min}}\mathbb{E}\|x^1-x^0\|^2\big)\\ &=2\eta\big(\tfrac{1}{m\sqrt{p_{\min}}}\mathbb{E}\|\bar{x}^1-x^0\|^2+m\sqrt{p_{\min}}\textstyle\sum_{i=1}^m p_i\tfrac{\eta^2}{m^2p_i^2}(S_ix^0)^2\big)\\ &\le 2\eta\big(\tfrac{1}{m\sqrt{p_{\min}}}\mathbb{E}\|\bar{x}^1-x^0\|^2+\tfrac{1}{m\sqrt{p_{\min}}}\mathbb{E}\|\bar{x}^1-x^0\|^2\big)\\ &=\tfrac{4\eta}{m\sqrt{p_{\min}}}\mathbb{E}\|\bar{x}^1-x^0\|^2.\end{aligned}$$

Rearranging the above inequality yields $\mathbb{E}\|\bar{x}^1-x^0\|^2\le \frac{1}{1-\frac{4\eta}{m\sqrt{p_{\min}}}}\mathbb{E}\|\bar{x}^2-x^1\|^2$. By (3.12) and $\rho>1$, it holds that $0<\eta\le(1-\frac{1}{\rho})\frac{m\sqrt{p_{\min}}}{8}\frac{\rho^{1/2}-1}{\rho^{(\tau+1)/2}-1}\le(1-\frac{1}{\rho})\frac{m\sqrt{p_{\min}}}{4}$. Hence, $\mathbb{E}\|\bar{x}^1-x^0\|^2\le\rho\mathbb{E}\|\bar{x}^2-x^1\|^2$.

For the induction step, applying Young's inequality gives us

$$\begin{aligned}\mathbb{E}\|\bar{x}^k-x^{k-1}\|\|x^{k-t}-x^{k-t-1}\| &\le \tfrac12\mathbb{E}\big\{a\|x^{k-t}-x^{k-t-1}\|^2+\tfrac1a\|\bar{x}^k-x^{k-1}\|^2\big\}\\ &\le \tfrac12\mathbb{E}\big\{\tfrac{a}{m^2p_{\min}}\|\bar{x}^{k-t}-x^{k-t-1}\|^2+\tfrac1a\|\bar{x}^k-x^{k-1}\|^2\big\}\\ &\le \tfrac12\big\{\tfrac{a\rho^t}{m^2p_{\min}}+\tfrac1a\big\}\mathbb{E}\|\bar{x}^k-x^{k-1}\|^2\\ &=\tfrac{\rho^{t/2}}{m\sqrt{p_{\min}}}\mathbb{E}\|\bar{x}^k-x^{k-1}\|^2. \qquad (\text{letting } a=m\sqrt{p_{\min}}\rho^{-t/2})\end{aligned}$$

Taking the expectation on (3.15) and combining it with (3.14) yield

$$\begin{aligned}\mathbb{E}\|\bar{x}^k-x^{k-1}\|^2-\mathbb{E}\|\bar{x}^{k+1}-x^k\|^2&\le 8\eta\textstyle\sum_{t=0}^{\tau}\mathbb{E}\|\bar{x}^k-x^{k-1}\|\|x^{k-t}-x^{k-t-1}\|\\ \le\tfrac{8\eta}{m\sqrt{p_{\min}}}\textstyle\sum_{t=0}^{\tau}\rho^{t/2}\mathbb{E}\|\bar{x}^k-x^{k-1}\|^2 &\le \tfrac{8\eta}{m\sqrt{p_{\min}}}\tfrac{1-\rho^{(\tau+1)/2}}{1-\rho^{1/2}}\mathbb{E}\|\bar{x}^k-x^{k-1}\|^2.\end{aligned}$$

Finally, rearranging the above inequality and using (3.12) lead to $\mathbb{E}\|\bar{x}^k-x^{k-1}\|^2\le\rho\mathbb{E}\|\bar{x}^{k+1}-x^k\|^2$. This completes the proof. □

With this lemma, we are ready to derive the linear convergence rate of ARock.

THEOREM 3.9 (Linear convergence). *Assume that S is quasi-μ-strongly monotone with $\mu>0$. Let $\beta\in(0,1)$ and $(x^k)_{k\ge0}$ be the sequence generated by ARock with a constant stepsize $\eta\in(0,\min\{\underline{\eta}_1,\underline{\eta}_2\}]$, where $\underline{\eta}_1$ is given in (3.12) and*

$$\underline{\eta}_2=\frac{-b+\sqrt{b^2+4(1-\beta)a}}{2a},\quad a=\frac{2\beta\mu\tau}{m^2p_{\min}}\frac{\rho(\rho^\tau-1)}{\rho-1},\quad b=\frac{1}{mp_{\min}}+\frac{2}{m}\sqrt{\frac{\rho(\rho^\tau-1)\tau}{(\rho-1)p_{\min}}}.\tag{3.16}$$

Then

$$\mathbb{E}\left(\|x^k - x^*\|^2\right) \leq \left(1 - \frac{\beta\mu\eta}{m}\right)^k \|x^0 - x^*\|^2. \tag{3.17}$$

Proof. Following the proof of Lemma 3.2 and starting from (3.6), we have

$$\begin{aligned}
&\langle S\hat{x}^k, x^* - x^k\rangle \\
\leq&\langle S\hat{x}^k - Sx^*, x^* - \hat{x}^k\rangle + \tfrac{1}{2\eta}\textstyle\sum_{d\in J(k)}\left(\tfrac{1}{\gamma}\|x^k - \bar{x}^{k+1}\|^2 + \gamma\|x^d - x^{d+1}\|^2\right) \\
\leq& -\beta\mu\|\hat{x}^k - x^*\|^2 - \tfrac{1-\beta}{2}\|S\hat{x}^k\|^2 + \tfrac{1}{2\eta}\textstyle\sum_{d\in J(k)}\left(\tfrac{1}{\gamma}\|x^k - \bar{x}^{k+1}\|^2 + \gamma\|x^d - x^{d+1}\|^2\right) \\
=& -\beta\mu\|x^k - x^* + \textstyle\sum_{d\in J(k)}(x^d - x^{d+1})\|^2 - \tfrac{1-\beta}{2\eta^2}\|x^k - \bar{x}^{k+1}\|^2 \\
&+ \tfrac{|J(k)|}{2\gamma\eta}\|x^k - \bar{x}^{k+1}\|^2 + \tfrac{\gamma}{2\eta}\textstyle\sum_{d\in J(k)}\|x^d - x^{d+1}\|^2 \\
\leq& -\tfrac{\beta\mu}{2}\|x^k - x^*\|^2 + \beta\mu\|\textstyle\sum_{d\in J(k)}(x^d - x^{d+1})\|^2 - \tfrac{1-\beta}{2\eta^2}\|x^k - \bar{x}^{k+1}\|^2 \\
&+ \tfrac{|J(k)|}{2\gamma\eta}\|x^k - \bar{x}^{k+1}\|^2 + \tfrac{\gamma}{2\eta}\textstyle\sum_{d\in J(k)}\|x^d - x^{d+1}\|^2 \\
\leq& -\tfrac{\beta\mu}{2}\|x^k - x^*\|^2 + \beta\mu|J(k)|\textstyle\sum_{d\in J(k)}\|x^d - x^{d+1}\|^2 - \tfrac{1-\beta}{2\eta^2}\|x^k - \bar{x}^{k+1}\|^2 \\
&+ \tfrac{|J(k)|}{2\gamma\eta}\|x^k - \bar{x}^{k+1}\|^2 + \tfrac{\gamma}{2\eta}\textstyle\sum_{d\in J(k)}\|x^d - x^{d+1}\|^2,
\end{aligned}$$

where the second inequality holds because S is $\frac{1}{2}$-cocoercive and also quasi-μ-strongly monotone, and the last one comes from the Cauchy-Schwartz inequality. Plugging the above inequality and (3.5) into (3.4) and noting $|J(k)| \subset \{k-\tau, \ldots, k-1\}$ gives

$$\begin{aligned}
\mathbb{E}\left(\|x^{k+1} - x^*\|^2 \,\middle|\, \mathcal{X}^k\right) \leq& \left(1 - \tfrac{\beta\mu\eta}{m}\right)\|x^k - x^*\|^2 + \tfrac{1}{m}\left(2\beta\eta\mu\tau + \gamma\right)\textstyle\sum_{d=k-\tau}^{k-1}\|x^d - x^{d+1}\|^2 \\
&+ \tfrac{1}{m}\left(\tfrac{\tau}{\gamma} + \tfrac{1}{mp_{\min}} - \tfrac{1-\beta}{\eta}\right)\|x^k - \bar{x}^{k+1}\|^2.
\end{aligned}$$

Taking expectation over both sides of the above inequality, noting $\mathbb{E}\|x^d - x^{d+1}\|^2 \leq \frac{1}{m^2 p_{\min}}\mathbb{E}\|x^d - \bar{x}^{d+1}\|^2$, and using Lemma 3.8, we have

$$\begin{aligned}
&\mathbb{E}\left(\|x^{k+1} - x^*\|^2\right) \\
\leq&\left(1 - \tfrac{\beta\mu\eta}{m}\right)\mathbb{E}\|x^k - x^*\|^2 + \tfrac{1}{m^3 p_{\min}}\left(2\beta\eta\mu\tau + \gamma\right)\textstyle\sum_{d=1}^{\tau}\rho^d\mathbb{E}\|x^k - \bar{x}^{k+1}\|^2 \\
&+ \tfrac{1}{m}\left(\tfrac{\tau}{\gamma} + \tfrac{1}{mp_{\min}} - \tfrac{1-\beta}{\eta}\right)\mathbb{E}\|x^k - \bar{x}^{k+1}\|^2 \\
=&\left(1 - \tfrac{\beta\mu\eta}{m}\right)\mathbb{E}\|x^k - x^*\|^2 + \tfrac{1}{m^3 p_{\min}}\left(2\beta\eta\mu\tau + \gamma\right)\tfrac{\rho(\rho^\tau - 1)}{\rho - 1}\mathbb{E}\|x^k - \bar{x}^{k+1}\|^2 \\
&+ \tfrac{1}{m}\left(\tfrac{\tau}{\gamma} + \tfrac{1}{mp_{\min}} - \tfrac{1-\beta}{\eta}\right)\mathbb{E}\|x^k - \bar{x}^{k+1}\|^2 \\
=&\left(1 - \tfrac{\beta\mu\eta}{m}\right)\mathbb{E}\|x^k - x^*\|^2 \\
&+ \tfrac{1}{m}\left(\tfrac{2\beta\eta\mu\tau}{m^2 p_{\min}}\tfrac{\rho(\rho^\tau - 1)}{\rho - 1} + \tfrac{2}{m}\sqrt{\tfrac{\rho(\rho^\tau - 1)\tau}{(\rho - 1)p_{\min}}} + \tfrac{1}{mp_{\min}} - \tfrac{1-\beta}{\eta}\right)\mathbb{E}\|x^k - \bar{x}^{k+1}\|^2 \\
\leq&\left(1 - \tfrac{\beta\mu\eta}{m}\right)\mathbb{E}\|x^k - x^*\|^2,
\end{aligned}$$

where we have let $\gamma = m\sqrt{\frac{\tau(\rho-1)p_{\min}}{\rho(\rho^\tau - 1)}}$ in the second equality, and the last inequality holds because of the choice of η. Therefore, (3.17) holds. ∎

Remark 4. *Assume i_k is chosen uniformly at random, so $p_{\min} = \frac{1}{m}$. We consider the case when m and τ are large. Let $\sqrt{\rho} = 1 + \frac{1}{\tau}$. Then from the fact that $(1 + \frac{1}{k})^k$ increasingly converges to the natural number e, we have from* (3.12) *that* $\underline{\eta}_1 = O(\frac{\sqrt{m}}{\tau^2})$*. In addition, note from* (3.16) *that* $a = O(b^2) = O(\frac{\tau^2}{m})$*, and thus* $\underline{\eta}_2 = O(\frac{\sqrt{m}}{\tau})$*. Therefore, if* $\tau = O(m^{\frac{1}{4}})$*, then the stepsize in Theorem 3.9 can be* $\eta = O(1)$*. Hence, linear speedup can be achieved.*

4. Experiments.

We illustrate the behavior of ARock for solving the ℓ_1 regularized logistic regression problem. Our primary goal is to show the efficiency of the async-parallel implementation compared to the single-threaded implementation and the sync-parallel implementation.

Our experiments run on 1 to 32 threads on a machine with eight Quad-Core AMD Opteron™ Processors (32 cores in total) and 64 Gigabytes of RAM. All of the experiments were coded in C++ and OpenMP. We use the Eigen library¶ for sparse matrix operations. Our codes [21] as well as numerical results for other applications are publicly available on the authors' website.

The running times and speedup ratios of both sync-parallel and async-parallel algorithms are sensitive to a number of factors, such as the size of each coordinate update (granularity), sparsity of the problem data, compiler optimization flags, and operations that affect cache performance and memory access contention. In addition, since all agents in the sync-parallel implementation must wait for the last agent to finish an iteration, a large load imbalance will significantly degrade the performance. We do not have the space in this paper to present numerical results under all variations of these cases.

4.1. ℓ_1 regularized logistic regression.

In this subsection, we apply ARock with the update (2.6) to the ℓ_1 regularized logistic regression problem:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ \lambda\|x\|_1 + \frac{1}{N}\sum_{i=1}^{N}\log\left(1+\exp(-b_i\cdot a_i^T x)\right), \tag{4.1}$$

where $\{(a_i, b_i)\}_{i=1}^N$ is the set of sample-label pairs with $b_i \in \{1, -1\}$, $\lambda = 0.0001$, and n and N represent the numbers of features and samples, respectively. This test uses the datasets‖: rcv1 and news20, which are summarized in Table 1.

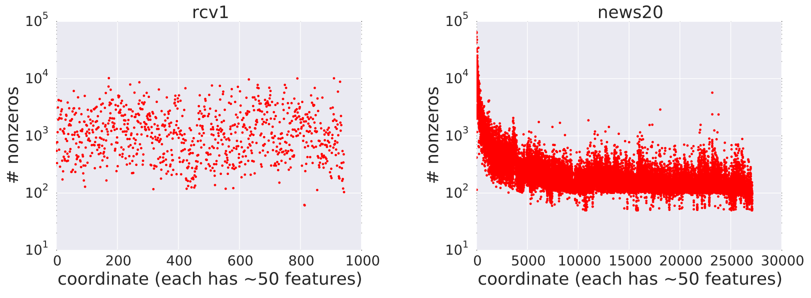
Name	# samples	# features	# nonzeros in $\{a_1, \dots, a_N\}$
rcv1	20, 242	47, 236	1, 498, 952
news20	19, 996	1, 355, 191	9, 097, 916

Table 1: Two datasets for sparse logistic regression.

We let each coordinate hold roughly 50 features. Since the total number of features is not divisible by 50, some coordinates have 51 features. We let each agent draw a coordinate uniformly at random at each iteration. We stop all the tests after 100 epochs since they have nearly identical progress per iteration. The step size is set to $\eta_k = 0.9$, $\forall k$. Let $A = [a_1, \dots, a_N]^T$ and $b = [b_1, ..., b_N]^T$. In global memory, we store A, b, and x. We also store the product Ax in global memory so that the forward step can be efficiently computed. Whenever a coordinate of x gets updated, Ax is immediately updated at a low cost. Note that if Ax is *not* stored in global memory, every coordinate update will have to compute Ax from scratch, which involves the entire x and will be very expensive.

¶http://eigen.tuxfamily.org

‖http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/



(a) dataset rcv1 (b) dataset news20

Fig. 3: The distribution of coordinate sparsity. Each dot represents the total number of nonzeros in the vectors a_i that correspond to each coordinate. The large distribution in (b) is responsible for the large load imbalance and thus the poor sync-parallel performance.

Table 2 gives the running times of the sync-parallel and ARock (async-parallel) implementations on the two datasets. We can observe that ARock achieves almost-linear speedup, but sync-parallel scales very poorly as we explain below.

In the sync-parallel implementation, all the running cores have to wait for the last core to finish an iteration, and therefore if a core has a large load, it slows down the iteration. Although every core is (randomly) assigned to roughly the same number of features (either 50 or 51 components of x) at each iteration, their a_i's have very different numbers of nonzeros (see Figure 3 for the distribution), and the core with the largest number of nonzeros is the slowest (Sparse matrix computation is used for both datasets, which are very large.) As more cores are used, despite that they altogether do more work at each iteration, the per-iteration time increases as the slowest core tends to be slower. The very large imbalance of load explains why the 32 cores only give speedup ratios of 4.0 and 1.3 in Table 2.

On the other hand, being asynchronous, ARock does not suffer from the load imbalance. Its performance grows nearly linear with the number of cores. In theory, a large load imbalance may cause a large τ, and thus a small η_k. However, the uniform $\eta_k = 0.9$ works well in all the tests, possibly because the a_i's are sparse.

Finally, we have observed that the progress toward solving (4.1) is mainly a function of the number of epochs and does not change appreciably when the number of cores increases or between sync-parallel and async-parallel. Therefore, we always stop at 100 epochs.

5. Conclusion. We have proposed an async-parallel framework, ARock, for finding a fixed-point of a nonexpansive operator by coordinate updates. We establish the almost sure weak and strong convergence, linear convergence rate and almost-linear speedup of ARock under certain assumptions. Preliminary numerical results on real data illustrate the high efficiency of the proposed framework compared to the traditional parallel (sync-parallel) algorithms.

6. Acknowledgements. We would like to thank Brent Edmunds for offering invaluable suggestions on the organization and writing of this paper. We would also like to thank Robert Hannah for coming up with the dual-memory approach. The authors are grateful to Kun Yuan for helpful discussions on decentralized optimization.

REFERENCES

# cores	rcv1				news20			
	Time (s)		Speedup		Time (s)		Speedup	
	async	sync	async	sync	async	sync	async	sync
1	122.0	122.0	1.0	1.0	591.1	591.3	1.0	1.0
2	63.4	104.1	1.9	1.2	304.2	590.1	1.9	1.0
4	32.7	83.7	3.7	1.5	150.4	557.0	3.9	1.1
8	16.8	63.4	7.3	1.9	78.3	525.1	7.5	1.1
16	9.1	45.4	13.5	2.7	41.6	493.2	14.2	1.2
32	4.9	30.3	24.6	4.0	22.6	455.2	26.1	1.3

Table 2: Running times of ARock (async-parallel) and sync-parallel FBS implementations for the ℓ_1 regularized logistic regression on two datasets. Sync-parallel has a very poor speedup due to the large distribution of coordinate sparsity (Figure 3) and thus the large load imbalance across cores.

[1] Dan Aharoni and Amnon Barak, *Parallel iterative discontinuous Galerkin finite-element methods*, in Discontinuous Galerkin Methods, Springer, 2000, pp. 247–254.

[2] Dganit Amitai, Amir Averbuch, Moshe Israeli, and Samuel Itzikowitz, *Implicit-explicit parallel asynchronous solver of parabolic pdes*, SIAM Journal on Scientific Computing, 19 (1998), pp. 1366–1404.

[3] Haim Avron, Alex Druinsky, and Anshul Gupta, *Revisiting asynchronous linear solvers: Provable convergence rate through randomization*, in Parallel and Distributed Processing Symposium, 2014 IEEE 28th International, IEEE, 2014, pp. 198–207.

[4] Jacques Bahi, Jean-Claude Miellou, and Karim Rhofir, *Asynchronous multisplitting methods for nonlinear fixed point problems*, Numerical Algorithms, 15 (1997), pp. 315–345.

[5] Gérard M Baudet, *Asynchronous iterative methods for multiprocessors*, Journal of the ACM (JACM), 25 (1978), pp. 226–244.

[6] Heinz H Bauschke and Patrick L Combettes, *Convex analysis and monotone operator theory in Hilbert spaces*, Springer Science & Business Media, 2011.

[7] Dimitri P Bertsekas, *Distributed asynchronous computation of fixed points*, Mathematical Programming, 27 (1983), pp. 107–120.

[8] Dimitri P Bertsekas and John N Tsitsiklis, *Parallel and distributed computation: numerical methods*, vol. 23, Prentice hall Englewood Cliffs, NJ, 1989.

[9] ———, *Some aspects of parallel and distributed iterative algorithmsa survey*, Automatica, 27 (1991), pp. 3–21.

[10] Iain Bethune, J Mark Bull, Nicholas J Dingle, and Nicholas J Higham, *Performance analysis of asynchronous Jacobi's method implemented in mpi, shmem and openmp*, International Journal of High Performance Computing Applications, 28 (2014), pp. 97–111.

[11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends in Machine Learning, 3 (2011), pp. 1–122.

[12] Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin, *Coordinate descent method for large-scale l2-loss linear support vector machines*, The Journal of Machine Learning Research, 9 (2008), pp. 1369–1398.

[13] M. Chau, P. Spiteri, R. Guivarch, and H.C. Boisson, *Parallel asynchronous iterations for the solution of a 3d continuous flow electrophoresis problem*, Computers & Fluids, 37 (2008), pp. 1126 – 1137.

[14] Daniel Chazan and Willard Miranker, *Chaotic relaxation*, Linear algebra and its applications, 2 (1969), pp. 199–222.

[15] Patrick L. Combettes and Jean-Christophe Pesquet, *Stochastic quasi-Fejér block-coordinate fixed point iterations with random sweeping*, SIAM Journal on Optimization, 25 (2015), pp. 1221–1248.

[16] Laurent Condat, *A primal–dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms*, Journal of Optimization Theory and Applications, 158 (2013), pp. 460–479.

[17] Damek Davis and Wotao Yin, *A three-operator splitting scheme and its optimization applications*, arXiv preprint arXiv:1504.01032, (2015).

[18] ———, *Convergence rate analysis of several splitting schemes*, in Splitting Methods in Communication and Imaging, Science and Engineering,, R. Glowinski, S. Osher, and W. Yin,

eds., Springer, New York, 2016.

[19] ———, *Faster convergence rates of relaxed Peaceman-Rachford and ADMM under regularity assumptions*, Mathematics of Operations Research, (to appear, 2016).

[20] Diego A Donzis and Konduri Aditya, *Asynchronous finite-difference schemes for partial differential equations*, Journal of Computational Physics, 274 (2014), pp. 370–392.

[21] Brent Edmunds, Zhimin Peng, and Wotao Yin, *TMAC: A toolbox of modern async-parallel, coordinate, splitting, and stochastic methods*, arXiv preprint arXiv:1606.04551, (2016).

[22] Didier El Baz, Pierre Spiteri, Jean Claude Miellou, and Didier Gazen, *Asynchronous iterative algorithms with flexible communication for nonlinear network flow problems*, Journal of Parallel and Distributed Computing, 38 (1996), pp. 1–15.

[23] Mouhamed Nabih El Tarazi, *Some convergence results for asynchronous algorithms*, Numerische Mathematik, 39 (1982), pp. 325–340.

[24] Lei Fang and Panos J Antsaklis, *Information consensus of asynchronous discrete-time multi-agent systems*, in American Control Conference, 2005. Proceedings of the 2005, IEEE, 2005, pp. 1883–1888.

[25] Andreas Frommer, Hartmut Schwandt, and Daniel B Szyld, *Asynchronous weighted additive schwarz methods*, Electronic Transactions on Numerical Analysis, 5 (1997), pp. 48–61.

[26] Andreas Frommer and Daniel B Szyld, *On asynchronous iterations*, Journal of computational and applied mathematics, 123 (2000), pp. 201–216.

[27] Daniel Gabay, *Chapter ix applications of the method of multipliers to variational inequalities*, Studies in mathematics and its applications, 15 (1983), pp. 299–331.

[28] Roland Glowinski and A Marroco, *Sur l'approximation, par elements finis d'ordre un, et la resolution, par penalisation-dualite d'une classe de problemes de dirichlet non lineaires*, ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique, 9 (1975), pp. 41–76.

[29] Mingyi Hong, *A distributed, asynchronous and incremental algorithm for nonconvex optimization: An ADMM based approach*, arXiv preprint arXiv:1412.6058, (2014).

[30] White House, *Big data: Seizing opportunities, preserving values*, 2014.

[31] Cho-Jui Hsieh, Hsiang-Fu Yu, and Inderjit S Dhillon, *Passcode: Parallel asynchronous stochastic dual co-ordinate descent*, in Proceedings of the 32nd International Conference on Machine Learning (ICML-15), 2015, pp. 2370–2379.

[32] Franck Iutzeler, Pascal Bianchi, Philippe Ciblat, and Walid Hachem, *Asynchronous distributed optimization using a randomized alternating direction method of multipliers*, in Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on, IEEE, 2013, pp. 3671–3676.

[33] Mark Aleksandrovich Krasnosel'skii, *Two remarks on the method of successive approximations*, Uspekhi Matematicheskikh Nauk, 10 (1955), pp. 123–127.

[34] Aapo Kyrola, Daniel Bickson, Carlos Guestrin, and Joseph K Bradley, *Parallel coordinate descent for l1-regularized loss minimization*, in Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 321–328.

[35] B Lang, JC Miellou, and P Spiteri, *Asynchronous relaxation algorithms for optimal control problems*, Mathematics and computers in simulation, 28 (1986), pp. 227–242.

[36] Remi Leblond, Fabian Pedregosa, and Lacoste-Julien Simon, *ASAGA: Asynchronous parallel SAGA*, arXiv preprint arXiv:1606.04809, (2016).

[37] Pierre-Louis Lions and Bertrand Mercier, *Splitting algorithms for the sum of two nonlinear operators*, SIAM Journal on Numerical Analysis, 16 (1979), pp. 964–979.

[38] Ji Liu and Stephen J Wright, *Asynchronous stochastic coordinate descent: Parallelism and convergence properties*, SIAM Journal on Optimization, 25 (2015), pp. 351–376.

[39] Ji Liu, Stephen J Wright, Christopher Ré, Victor Bittorf, and Srikrishna Sridhar, *An asynchronous parallel stochastic coordinate descent algorithm*, Journal of Machine Learning Research, 16 (2015), pp. 285–322.

[40] A Nedić, Dimitri P Bertsekas, and Vivek S Borkar, *Distributed asynchronous incremental subgradient methods*, Studies in Computational Mathematics, 8 (2001), pp. 381–407.

[41] Angelia Nedic and Asuman Ozdaglar, *Distributed subgradient methods for multi-agent optimization*, Automatic Control, IEEE Transactions on, 54 (2009), pp. 48–61.

[42] Yu Nesterov, *Efficiency of coordinate descent methods on huge-scale optimization problems*, SIAM Journal on Optimization, 22 (2012), pp. 341–362.

[43] Amedeo R Odoni and Richard C Larson, *Urban operations research*, Dynamic Ideas, 2nd ed., 2007.

[44] Zdzisław Opial, *Weak convergence of the sequence of successive approximations for nonexpansive mappings*, Bulletin of the American Mathematical Society, 73 (1967), pp. 591–598.

[45] Gregory B Passty, *Ergodic convergence to a zero of the sum of monotone operators in Hilbert*

space, Journal of Mathematical Analysis and Applications, 72 (1979), pp. 383–390.

[46] Zhimin Peng, Tianyu Wu, Yangyang Xu, Ming Yan, and Wotao Yin, *Coordinate friendly structures, algorithms and applications*, Annals of Mathematical Sciences and Applications, 1 (2016), pp. 57–119.

[47] Zhimin Peng, Ming Yan, and Wotao Yin, *Parallel and distributed sparse optimization*, in Signals, Systems and Computers, 2013 Asilomar Conference on, IEEE, 2013, pp. 659–646.

[48] WV Petryshyn, *Construction of fixed points of demicompact mappings in Hilbert space*, Journal of Mathematical Analysis and Applications, 14 (1966), pp. 276–284.

[49] Juan Peypouquet and Sylvain Sorin, *Evolution equations for maximal monotone operators: Asymptotic analysis in continuous and discrete time*, Journal of Convex Analysis, 17 (2010), pp. 1113–1163.

[50] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu, *Hogwild: A lock-free approach to parallelizing stochastic gradient descent*, in Advances in Neural Information Processing Systems, 2011, pp. 693–701.

[51] Peter Richtárik and Martin Takáč, *Parallel coordinate descent methods for big data optimization*, Mathematical Programming, Series A, 156 (2016), pp. 433–484.

[52] Herbert Robbins and David Siegmund, *A convergence theorem for non negative almost supermartingales and some applications*, in Herbert Robbins Selected Papers, Springer, 1985, pp. 111–135.

[53] Jack L Rosenfeld, *A case study in programming for parallel-processors*, Communications of the ACM, 12 (1969), pp. 645–655.

[54] Xue-Cheng Tai and Paul Tseng, *Convergence rate analysis of an asynchronous space decomposition method for convex minimization*, Mathematics of Computation, 71 (2002), pp. 1105–1135.

[55] Paul Tseng, *On the rate of convergence of a partially asynchronous gradient projection algorithm*, SIAM Journal on Optimization, 1 (1991), pp. 603–619.

[56] Paul Tseng, Dimitri P Bertsekas, and John N Tsitsiklis, *Partially asynchronous, parallel algorithms for network flow and other problems*, SIAM Journal on Control and Optimization, 28 (1990), pp. 678–710.

[57] Bng Công Vũ, *A splitting algorithm for dual monotone inclusions involving cocoercive operators*, Advances in Computational Mathematics, 38 (2013), pp. 667–681.

[58] Ermin Wei and Asuman Ozdaglar, *On the o(1/k) convergence of asynchronous distributed alternating direction method of multipliers*, in Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE, IEEE, 2013, pp. 551–554.

[59] Ming Yan and Wotao Yin, *Self equivalence of the alternating direction method of multipliers*, in Splitting Methods in Communication and Imaging, Science and Engineering,, R. Glowinski, S. Osher, and W. Yin, eds., Springer, New York, 2016.

[60] Kun Yuan, Qing Ling, and Wotao Yin, *On the convergence of decentralized gradient descent*, arXiv preprint arXiv:1310.7063, (2013).

[61] Ruiliang Zhang and James Kwok, *Asynchronous distributed ADMM for consensus optimization*, in Proceedings of the 31st International Conference on Machine Learning (ICML-14), 2014, pp. 1701–1709.

Appendix A. Derivation of certain updates.

We show in details how to obtain the updates in (2.10) and (2.15).

A.1. Derivation of updates in (2.10).

Let $x = (x_1, \dots, x_m) \in \mathcal{H}^m$,

$$f(x) := \sum_{i=1}^m \mathcal{I}_{C_i}(x_i), \qquad g(x) := \mathcal{I}_{\{x_1=\cdots=x_m\}}(x),$$

where $g(x)$ equals 0 if $x_1 = \cdots = x_m$ and ∞ otherwise. Then (2.9a) reduces to

$$\begin{aligned}\hat{x}^k &= \operatorname*{arg\,min}_{z\in\mathcal{H}^m} g(z) + \frac{1}{2\gamma}\|z - \hat{z}^k\|^2 = \operatorname*{arg\,min}_{z\in\mathcal{H}^m: z_1=\cdots=z_m} \|z - \hat{z}^k\|^2 \\ &= \operatorname*{arg\,min}_{z\in\mathcal{H}^m: z_1=\cdots=z_m} \sum_{i=1}^m \|z_1 - \hat{z}_i^k\|^2 = \left(\frac{1}{m}\sum_{i=1}^m \hat{z}_i^k, \dots, \frac{1}{m}\sum_{i=1}^m \hat{z}_i^k\right) \in \mathcal{H}^m,\end{aligned}$$

where the last equality is obtained by noting that $z_1 = \frac{1}{m}\sum_{i=1}^m \hat{z}_i^k$ is the unique minimizer of $\sum_{i=1}^m \|z_1 - \hat{z}_i^k\|^2$. Next, (2.9b) reduces to

$$\hat{y}^k = \operatorname*{arg\,min}_{z\in\mathcal{H}^m} f(z) + \frac{1}{2\gamma}\|z - (2\hat{x}^k - \hat{z}^k)\|^2 = \operatorname*{arg\,min}_{z:z_i\in C_i,\forall i} \sum_{i=1}^m \|z_i - (2\hat{x}_i^k - \hat{z}_i^k)\|^2.$$

It is easy to see that $\hat{y}_i^k = \mathrm{Proj}_{C_i}(2\hat{x}_i^k - \hat{z}_i^k)$, $\forall i$.

Since (2.9c) only updates the i_kth coordinate of z, we only need $\hat{x}_{i_k}^k$ and $\hat{y}_{i_k}^k$, and thus in (2.10a) and (2.10b), we only compute $\hat{x}_{i_k}^k$ and $\hat{y}_{i_k}^k$. Plugging the above $\hat{x}^k$ and $\hat{y}^k$ into (2.9c) gives (2.10c) directly.

A.2. Derivation of (2.15). We first show how to get (2.12). The Lagrangian of (2.11) is $L(x,y,w) = f(x) + g(y) - \langle w, Ax + By - b\rangle$, and the Lagrange dual function is

$$\begin{aligned} d(w) =& \min_{x\in\mathcal{H}_1, y\in\mathcal{H}_2} L(x,y,w) \\ =&\big(\min_{x\in\mathcal{H}_1} f(x) - \langle A^*w, x\rangle\big) + \big(\min_{y\in\mathcal{H}_2} g(y) - \langle B^*w, y\rangle\big) + \langle w, b\rangle \\ =&-\big(\max_{x\in\mathcal{H}_1} -f(x) + \langle A^*w, x\rangle\big) - \big(\max_{y\in\mathcal{H}_2} -g(y) + \langle B^*w, y\rangle\big) + \langle w, b\rangle \\ =&-f^*(A^*w) - g^*(B^*w) + \langle w, b\rangle, \end{aligned}$$

where the last equality is from the definition of convex conjugate: $f^*(z) = \max_x \langle z, x\rangle - f(x)$. Hence, the dual problem is $\max_w d(w)$, which is equivalent to (2.12).

Secondly, we show why $z^+ = \mathbf{prox}_{\gamma\cdot d_g}(z)$ is given by (2.14). Note

$$\begin{aligned} \min_s d_g(s) + \tfrac{1}{2\gamma}\|s - z\|^2 &= \min_s g^*(B^*s) - \langle s, b\rangle + \tfrac{1}{2\gamma}\|s - z\|^2 \\ &= \min_s \max_y \langle B^*s, y\rangle - g(y) - \langle s, b\rangle + \tfrac{1}{2\gamma}\|s - z\|^2 \\ &= \max_y \min_s \langle B^*s, y\rangle - g(y) - \langle s, b\rangle + \tfrac{1}{2\gamma}\|s - z\|^2 \\ &= \max_y \min_s \langle s, By - b\rangle - g(y) + \tfrac{1}{2\gamma}\|s - z\|^2 \\ &= \max_y -g(y) + \langle z, By - b\rangle - \tfrac{\gamma}{2}\|By - b\|^2 \\ &= -\min_y g(y) - \langle z, By - b\rangle + \tfrac{\gamma}{2}\|By - b\|^2, \end{aligned}$$

where the fifth equality holds because $s^* = z - \gamma(By - b) = \operatorname{arg\,min}_s \langle s, By - b\rangle + \frac{1}{2\gamma}\|s - z\|^2$. Hence, by the definition of the proximal operator and the above arguments, we have that $z^+ = \mathbf{prox}_{\gamma\cdot d_g}(z)$ can be obtained from (2.14). Then (2.13) is from (2.14) through replacing g to f, B to A, and b to 0.

Finally, it is straightforward to have (2.15) by plugging (2.13) and (2.14) into (2.9).

Appendix B. Derivation of async-parallel ADMM for decentralized optimization. This section describes how to implement the updates (2.15) for the model (2.21).

In (2.21), $g(y)$ and b vanish and, corresponding to the two constraints $x_i = y_{ij}$ and $x_j = y_{ij}$, the two rows of matrices A and B are $\begin{bmatrix} \cdots & 1 & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & 1 & \cdots \end{bmatrix}$ $\begin{bmatrix} \cdots & -1 & \cdots \\ \cdots & -1 & \cdots \end{bmatrix}$, where $\cdots$ are zeros, the two coefficients 1 correspond to x_i and x_j, and the two coefficients -1 correspond to y_{ij}. Then, (2.15a) and (2.15b) can be calculated as

$$\begin{aligned} \hat{y}_{li}^k &= (\hat{z}_{li,l}^k + \hat{z}_{li,i}^k)/(2\gamma) \quad \forall l \in L(i), \\ (\hat{w}_g^k)_{li,i} &= (\hat{z}_{li,i}^k - \hat{z}_{li,r}^k)/2 \quad \forall l \in L(i), \\ \hat{y}_{ir}^k &= (\hat{z}_{ir,i}^k + \hat{z}_{ir,r}^k)/(2\gamma) \quad \forall r \in R(i), \\ (\hat{w}_g^k)_{ir,i} &= (\hat{z}_{ir,i}^k - \hat{z}_{ir,r}^k)/2 \quad \forall r \in R(i). \end{aligned}$$

In addition, $\hat{x}_i^k$ can be obtained by solving (2.22a), and both $z_{li,i}^{k+1}$ and $z_{ir,i}^{k+1}$ can be updated from (2.22b) and (2.22c).

Furthermore, as mentioned in Section 2.6.2, we can derive another version of async-parallel ADMM for decentralized optimization, which reduces to the algorithm in [58], by activating an edge $(i,j) \in E$ instead of an agent i each time. In this version, the agents i and j associated with the edge (i,j) must also be activated. Here we derive the update (2.15) for the model (2.21) with the update order of x and y swapped. Following (2.15) we obtain the following steps whenever an edge $(i,j) \in E$ is activated:

$$\begin{aligned}
\hat{x}_i^k &= \arg\min_{x_i} f_i(x_i) - \Big(\sum_{l\in L(i)} \hat{z}_{li,i}^k + \sum_{r\in R(i)} \hat{z}_{ir,i}^k\Big)x_i + \frac{\gamma}{2}|E(i)|\cdot\|x_i\|^2\\
\hat{x}_j^k &= \arg\min_{x_j} f_j(x_j) - \Big(\sum_{l\in L(j)} \hat{z}_{lj,j}^k + \sum_{r\in R(j)} \hat{z}_{jr,j}^k\Big)x_j + \frac{\gamma}{2}|E(j)|\cdot\|x_j\|^2\\
(\hat{w}_f^k)_{ij,i} &= \hat{z}_{ij,i}^k - \gamma\hat{x}_i^k\\
(\hat{w}_f^k)_{ij,j} &= \hat{z}_{ij,j}^k - \gamma\hat{x}_j^k\\
\hat{y}_{ij}^k &= \arg\min_{y_{ij}} \langle 2(\hat{w}_f^k)_{ij,i} - \hat{z}_{ij,i}^k + 2(\hat{w}_f^k)_{ij,j} - \hat{z}_{ij,j}^k, y_{ij}\rangle + \frac{\gamma}{2}\|y_{ij}\|^2\\
(\hat{w}_g^k)_{ij,i} &= 2(\hat{w}_f^k)_{ij,i} - \hat{z}_{ij,i}^k + \gamma\hat{y}_{ij}^k\\
(\hat{w}_g^k)_{ij,j} &= 2(\hat{w}_f^k)_{ij,j} - \hat{z}_{ij,j}^k + \gamma\hat{y}_{ij}^k\\
z_{ij,i}^{k+1} &= z_{ij,i}^k + \eta_k((\hat{w}_g^k)_{ij,i} - (\hat{w}_f^k)_{ij,i})\\
z_{ij,j}^{k+1} &= z_{ij,j}^k + \eta_k((\hat{w}_g^k)_{ij,j} - (\hat{w}_f^k)_{ij,j}).
\end{aligned}$$

Every agent i in the network maintains the dual variables $z_{li,i}$, $l \in L(i)$, and $z_{ir,i}$, $r \in R(i)$, and the variables x, y, w are intermediate and do not need to be maintained between the activations. When an edge (i,j) is activated, the agents i and j first compute their $\{\hat{x}_i^k, (\hat{w}_f^k)_{ij,i}\}$ and $\{\hat{x}_j^k, (\hat{w}_f^k)_{ij,j}\}$ independently and respectively, then they collaboratively compute $\hat{y}_{ij}^k$, and finally they update their own $z_{ij,i}^k$ and $z_{ij,j}^k$, respectively. We allow adjacent edges (which share agents) to be activated in a short period of time when their updates are possibly overlapped in time. When $\tau = 0$, i.e., there is no simultaneous activation or overlap, it reduces to the algorithm in [58].