

Asynchronous Parallel Algorithms for Nonconvex Optimization

Loris Cannelli · Francisco Facchinei ·
Vyacheslav Kungurtsev · Gesualdo
Scutari

April 2, 2018

Abstract We propose a new asynchronous parallel block-descent algorithmic framework for the minimization of the sum of a smooth nonconvex function and a nonsmooth convex one, subject to both convex and nonconvex constraints. The proposed framework hinges on successive convex approximation techniques and a novel probabilistic model that captures key elements of modern computational architectures and asynchronous implementations in a more faithful way than current state-of-the-art models. Other key features of the framework are: i) it covers in a unified way several specific solution methods; ii) it accommodates a variety of possible parallel computing architectures; and iii) it can deal with nonconvex constraints. Almost sure convergence to stationary solutions is proved, and theoretical complexity results are provided, showing nearly ideal linear speedup when the number of workers is not too large.

Keywords Asynchronous algorithms · nonconvex constraints · parallel methods · probabilistic model.

Loris Cannelli and Gesualdo Scutari, School of Industrial Engineering, Purdue University, USA; E-mail: <lcannell, gscutari>@purdue.edu · Francisco Facchinei, Department of Computer, Control, and Management Engineering Antonio Ruberti, University of Rome La Sapienza, Roma, Italy; E-mail: francisco.facchinei@uniroma1.it · Vyacheslav Kungurtsev Dept. of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech E-mail: vyacheslav.kungurtsev@fel.cvut.cz. The work of Cannelli and Scutari was supported by the USA NSF under Grants CIF 1564044, CAREER Award No. 1555850, and CIF 1719205; and the Office of Naval Research (ONR) Grant N00014-16-1-2244. Facchinei was partially supported by the Italian Ministry of Education, Research and University, under the PLATINO (PLATform for INnOvative services in future internet) PON project, Grant Agreement no. PON01_01007. Kungurtsev was supported by the Czech Science Foundation project 17-26999S.

Part of this work has been presented to the *50th Asilomar Conference on Signals, Systems, and Computers* [6] and the *42nd IEEE International Conference on Acoustics, Speech, and Signal Processing* [5]. A two-part preliminary technical report was posted on arxiv on July 2016 [3] and January 2017 [4].

1 Introduction

We study asynchronous parallel block-descent methods for the following class of nonconvex nonsmooth minimization problems:

$$\begin{aligned} \min_{\mathbf{x} \triangleq (\mathbf{x}_1, \dots, \mathbf{x}_N)} \quad & F(\mathbf{x}) \triangleq f(\mathbf{x}) + \sum_{i=1}^N g_i(\mathbf{x}_i) \\ & \mathbf{x}_i \in \mathcal{X}_i, \quad i = 1, \dots, N, \end{aligned} \quad (\text{P})$$

where f is a smooth, possibly nonconvex function, g_i are possibly nonsmooth, convex functions, and $\mathcal{X}_i \subseteq \mathbb{R}^{n_i}$ is a closed, possibly nonconvex set.

Instances of Problem (P) arise in many fields, including compressed sensing, machine learning, data mining, and genomics, just to name a few. Typically, in data-driven applications f might measure the misfit between the observations and the postulated model, parametrized on \mathbf{x} , while the regularizers g_i encode structural constraints on the solution, such as sparsity.

Many of the aforementioned applications give rise to extremely large-scale problems, which naturally call for *asynchronous, parallel* solution methods. In fact, well suited to modern computational architectures, asynchronous methods reduce the idle times of workers, mitigate communication and/or memory-access congestion, and make algorithms more fault-tolerant. In this paper, we introduce a general asynchronous block-descent algorithm for finding stationary solutions of Problem (P).

We consider a generic multi-worker architecture (e.g., shared memory system, message passing-based system, cluster computer, cloud federation) wherein multiple workers, continuously and without coordination with each other, update a block-variable by solving a strongly convex block-model of Problem (P). More specifically, at iteration k , a worker updates a block-variable $\mathbf{x}_{i^k}^k$ of \mathbf{x}^k to $\mathbf{x}_{i^k}^{k+1}$, with i^k in the set $\mathcal{N} \triangleq \{1, \dots, N\}$, thus generating the vector \mathbf{x}^{k+1} . When updating block i^k , in general, the worker does not have access to the current vector \mathbf{x}^k , but it will use instead the local estimate $\mathbf{x}^{k-\mathbf{d}^k} \triangleq (x_1^{k-d_1^k}, x_2^{k-d_2^k}, \dots, x_N^{k-d_N^k})$, where $\mathbf{d}^k \triangleq (d_1^k, d_2^k, \dots, d_N^k)$ is the ‘‘vector of delays’’, whose components d_i^k are nonnegative integers. Note that $\mathbf{x}^{k-\mathbf{d}^k}$ is nothing else but a combination of delayed, block-variables. The way each worker forms its own estimate $\mathbf{x}^{k-\mathbf{d}^k}$ depends on the particular architecture under consideration and it is immaterial to the analysis of the algorithm. We only observe here that if all delays d_i^k are zeros, the model reduces to a standard synchronous one.

Given $\mathbf{x}^{k-\mathbf{d}^k}$ and i^k , block $\mathbf{x}_{i^k}^k$ is updated by solving the following *strongly convex* block-approximation of Problem (P):

$$\hat{\mathbf{x}}_{i^k}(\mathbf{x}^{k-\mathbf{d}^k}) \triangleq \underset{\mathbf{x}_{i^k} \in \tilde{\mathcal{X}}_{i^k}(\mathbf{x}^{k-\mathbf{d}^k})}{\operatorname{argmin}} \quad \tilde{f}_{i^k}(\mathbf{x}_{i^k}; \mathbf{x}^{k-\mathbf{d}^k}) + g_{i^k}(\mathbf{x}_{i^k}), \quad (1)$$

and then setting

$$\mathbf{x}_{i^k}^{k+1} = \mathbf{x}_{i^k}^k + \gamma \left(\hat{\mathbf{x}}_{i^k}(\mathbf{x}^{k-\mathbf{d}^k}) - \mathbf{x}_{i^k}^k \right). \quad (2)$$

In (1), \tilde{f}_{i^k} and $\tilde{\mathcal{X}}_{i^k}$ represent a strongly convex surrogate of f and a convex set obtained replacing the nonconvex functions defining \mathcal{X}_{i^k} by suitably chosen

upper convex approximations, respectively; both \tilde{f}_{i^k} and $\tilde{\mathcal{X}}_{i^k}$ are built using the out-of-sync information $\mathbf{x}^{k-\mathbf{d}^k}$. If the set \mathcal{X}_{i^k} is convex, then we will always take $\tilde{\mathcal{X}}_{i^k} = \mathcal{X}_{i^k}$. In (2), $\gamma \in (0, 1]$ is the stepsize. Note that, in the above asynchronous model, the worker that is in charge of the computation (1) and the consequent update (2) is immaterial.

Major contributions: Our main contributions are:

1. *A new probabilistic model for asynchrony fixing some unresolved issues:* Almost all modern asynchronous algorithms for convex and nonconvex problems are modeled in a probabilistic way. We put forth a novel probabilistic model describing the statistics of (i^k, \mathbf{d}^k) that differs markedly from existing ones. This new model allows us not only to fix some important theoretical issues that mar most of the papers in the field (see discussion below on related work), but it also lets us analyze for the first time in a sound way several practically used and effective computing settings and new asynchronous algorithms. For instance, it is widely accepted that in shared-memory systems, the best performance are obtained by first partitioning the variables among cores, and then letting each core update in an asynchronous fashion their own block-variables, according to some randomized cyclic rule. To the best of our knowledge, this is the first work proving convergence of such practically effective methods in an asynchronous setting.

2. *The ability to effectively deal with nonconvex constraints:* All the works in the literature but [9, 10] can deal only with unconstrained or convex constrained problems. On the other hand, the algorithms in [9, 10] require at each iteration the computation of the global optimal solution of nonconvex subproblems, which, except in few special cases, can be as difficult as solving the original nonconvex problem. Our method is the first asynchronous method that allows one to deal (under adequate assumptions) with nonconvex constraints while solving only strongly convex subproblems.

3. *The possibility to leverage potentially complex, but effective subproblems* (1): Asynchronous methods so far are all built around a proximal linearization method, which corresponds, in our framework, to setting

$$\tilde{f}_i(\mathbf{x}_{i^k}; \mathbf{x}^{k-\mathbf{d}^k}) = \nabla_{\mathbf{x}_{i^k}} f(\mathbf{x}^{k-\mathbf{d}^k})^T \left(\mathbf{x}_{i^k} - \mathbf{x}_{i^k}^{k-d_{i^k}^k} \right) + \beta \|\mathbf{x}_{i^k} - \mathbf{x}_{i^k}^{k-d_{i^k}^k}\|_2^2,$$

for some constant $\beta > 0$. This choice often leads to efficient solution methods and, in some cases, even to subproblems that admit a solution in closed-form. For instance, it has been shown to be very efficient on composite quadratic problems, like LASSO. However, moving to more nonlinear problems, one may want to use more complex/higher order models. In fact, the more sophisticated the subproblem (1), the better the overall behavior of the algorithm (at least in terms of iterations) is. This happens at the price of computationally more expensive subproblems. But in asynchronous and distributed methods, the bottleneck is often given by the communication cost. In these cases, it might be desirable to reduce the communication overhead at the price of more complex subproblems to solve. Furthermore, there are many application for which

one can define subproblems that, while not being proximal linearizations, still admit closed-form solutions (see, e.g., [8, 32]). Overall, the ability to use more complex subproblems is an additional degree of freedom that one may want to exploit to improve the performance of the algorithm.

4. *Almost sure convergence and complexity analysis:* We prove i) almost sure convergence to stationary solutions of Problem (P); and ii) convergence to ϵ -stationary solutions in an $\mathcal{O}(\epsilon^{-1})$ number of iterations. We remark that our convergence results match similar ones in the literature [9, 10, 21, 22, 27], which however were obtained in a simplified setting (e.g., only for unconstrained or convex constrained problems) and under unrealistic probabilistic assumptions on the pair index-delay (i^k, \mathbf{d}^k) (see discussion on related work). Our analysis builds on an induction technique based on our probabilistic model and a novel Lyapunov function that properly combines variable dynamics and their delayed versions.

5. *A theoretical almost linear speed-up for a wide range of number of cores:* The holy grail of asynchronous methods is the ideal linear speed-up (with respect to the number of workers). This theoretical limit is not achievable in practice; in fact, as the number of workers increases, the effective speedup is always limited by associated overheads (communication costs, conflicts, etc.), which make the linear growth impossible to achieve for arbitrarily large number of workers. By using the number of iterations needed to achieve an ϵ -stationary solution as a proxy for the computational time and leveraging our new Lyapunov function, we are able to show almost linear speed-up in many settings of practical interest. This is the first theoretical result on speedup, based on a realistic probabilistic model for asynchrony (see discussion in contribution 1).

Related work. Although asynchronous block-methods have a long history (see, e.g., [1, 2, 7, 14, 34]), their revival and probabilistic analysis have taken place only in recent years; this is mainly due to the current trend towards huge scale optimization and the availability of ever more complex computational architectures that call for efficient and resilient algorithms. Indeed, asynchronous parallelism has been applied to many state-of-the-art optimization algorithms, including stochastic gradient methods [16, 19, 20, 23–26] and ADMM-like schemes [15, 17, 35]. Block-Coordinate Descent (BCD) methods are part of the folklore in optimization; more recently, they have been proven to be particularly effective in solving very large-scale problems arising, e.g., from data-intensive applications. Their asynchronous counterpart has been introduced and studied in the seminal work [21], which motivated and oriented much of subsequent research in the field, see e.g. [9, 10, 22, 27, 28]. We refer the interested reader to [36] and references therein for a detailed overview of BCD methods. There are several differences between the above methods and the framework proposed in this paper, as detailed next.

- *On the probabilistic model:* All current probabilistic models for asynchronous BCD methods are based on the (implicit or explicit) assumption that the random variables i^k and \mathbf{d}^k are *independent*; this greatly simplifies the convergence analysis. However, in reality there is a strong dependence of the delays \mathbf{d}^k on

the updated block i^k . Consider the setting where the variables are partitioned among two workers and each worker updates only its own block-variables; let \mathcal{N}_1 and \mathcal{N}_2 be the index set of the blocks controlled by worker 1 and 2, respectively. It is clear that in the updates of worker 1 it will always be $d_i^k = 0$, for all $i \in \mathcal{N}_1$ and k , while (at least some) delays d_i^k associated with the blocks $i \in \mathcal{N}_2$ will be positive; the opposite happens to worker two. The independence assumption is unrealistic also in settings where all the workers share all the variables. Blocks that are updated less frequently than others, when updated, will have larger associated delays. This happens, for instance, in problems where i) some blocks are more expensive to update than others, because they are larger, bear more nonzero entries, or data retrieval requires longer times; or ii) the updates are carried by heterogeneous workers (e.g., some are faster or busier than others). We tested this assumption, performing an asynchronous algorithm on two different architectures and measuring the average delay corresponding to different blocks updated. The experiments were performed on a shared-memory system with 10 cores of an Intel E5-2699Av4 processor. An asynchronous algorithm was applied to a LASSO problem [33] with 10000 variables, partitioned uniformly into 100 contiguous blocks; the Hessian matrix was generated with high sparsity on several rows. All the cores can update any block, selected uniformly at random. We found that blocks associated with the sparse rows of the Hessian have delays \mathbf{d}^k with components between 0 and 3, while the delays of the other blocks were all bigger than 20. Even when the computing environment is homogeneous and/or the block updates have the same cost, the aforementioned dependence persists. We simulated a message-passing system on Purdue Community Cluster Snyder; we used two nodes of the cluster, each of them equipped with 10 cores of an Intel Xeon-E5 processors and its own shared memory. Every node can update every block, selected uniformly at random. We ran an asynchronous algorithm on the same LASSO problem described above but now with a dense Hessian matrix. The blocks updated by node 1 have an average delay of 12 while those updated by node 2 experience an average delay of 22. This can be due to several uncontrollable factors, like operation delay of 22. This can be due to several uncontrollable factors, like operation system and memory schedulers, buses controllers, etc., which are hard to rigorously model and analyze.

Another unrealistic assumption often made in the literature [9, 21, 22, 25] is that the block-indices i^k are selected *uniformly* at random. While this assumption simplifies the convergence analysis, it limits the applicability of the model; see Examples 4 and 5 in Section 3.1. In a nutshell, this assumption may be satisfied only if all workers have the same computational power and have access to all variables.

We conclude the discussion on probabilistic models underlying asynchronous algorithms mentioning the line of work dealing with stochastic gradient methods. Stochastic gradient methods are similar to block-descent approaches in that at each iteration sampling is performed to determine the nature of the update, but sampling is done among functions in an optimization problem minimizing the sum of functions, as opposed to block variables. A related, albeit different, issue of independence in the probabilistic models used in stochastic

gradient methods was first noted in the technical report [23], see also [19, 26] for further developments. These papers circumvent the issue by enforcing independence (a) using a particular manner of labeling iterations as well as (b) reading the entire vector of variables regardless of the sparsity pattern among the summand functions in the objective. However, the analysis in [19, 23, 26] is (c) only performed in the context of strongly convex unconstrained problems, (d) involves uniform sampling and (e) is only applicable for the shared memory setting. Thus, while the analysis and procedures described in the references above are interesting, on the whole requirements (b)-(e) make these proposals of marginal interest in the context of block-descent methods (even assuming they can actually be adapted to our setting).

Differently from the aforementioned works, our more general and sophisticated probabilistic model neither postulates the independence between i^k and \mathbf{d}^k nor requires artificial changes in the algorithm [e.g., costly unnecessary readings, as in (b)] to enforce it; it handles instead the potential dependency among variables directly. By doing so, one can establish convergence without requiring any of the restrictive conditions (b)-(e), and significantly enlarge the class of computational architecture falling within the model [e.g., going beyond (d) and (e)]—see Section 3.1 for several examples. The necessity of a new probabilistic model of asynchrony in BCD methods was first observed in our conference works [5, 6] while the foundations of our approach were presented in our technical reports [3, 4] along with some numerical results. Here we improve the analysis of [3, 4] by relaxing considerably the assumptions for convergence and tightening the complexity bounds.

- *Nonconvex constraints:* Another important feature of our algorithm is the ability to handle nonconvex objective functions and nonconvex constraints by an algorithm that only needs to solve, at each iteration, a strongly convex optimization subproblem. Almost all asynchronous methods cited above can handle only convex optimization problems or, in the case of fixed point problems, nonexpansive mappings. The exceptions are [20, 38] and, more relevant to our setting, [9, 10] that study unconstrained and constrained nonconvex optimization problems, respectively. However, the papers dealing with constrained problems, i.e. [9, 10], propose algorithms that require, at each iteration, the global solution of nonconvex subproblems. Except for few cases, the subproblems could be hard to solve and potentially as difficult as the original one.

- *Successive Convex Approximation:* All the asynchronous algorithms described so far use proximal linearization to define subproblems. As already pointed out, this is the first paper where subproblem models able to capture more structure of the objective functions are considered. This offers more freedom and flexibility to tailor the minimization algorithm to the problem structure, in order to obtain more efficient solution methods.

Notation: We use the following notation for random variables and their realizations: underlined symbols denote random variables, e.g., $\underline{\mathbf{x}}^k$, $\underline{\mathbf{x}}^{k-\underline{\mathbf{d}}^k}$, whereas the same symbols with no underline are the corresponding realizations.

2 Asynchronous Algorithmic Framework

In this section we introduce the assumptions on Problem (P) along with the formal description of the proposed algorithm. For simplicity of presentation, we begin studying (P) assuming that there are only convex constraints, i.e., all \mathcal{X}_i are convex. This unnecessary assumption will be removed in Section 5.

Assumption A (On Problem (P)).

- (A1) Each set $\mathcal{X}_i \subseteq \mathbb{R}^{n_i}$ is nonempty, closed, and convex;
- (A2) $f : \mathcal{O} \rightarrow \mathbb{R}$ is C^1 , where \mathcal{O} is an open set containing $\mathcal{X} \triangleq \mathcal{X}_1 \times \cdots \times \mathcal{X}_N$;
- (A3) $\nabla_{\mathbf{x}_i} f$ is L_f -Lipschitz continuous on \mathcal{X} ;
- (A4) Each $g_i : \mathcal{O}_i \rightarrow \mathbb{R}$ is convex, possibly nonsmooth, and L_g -Lipschitz continuous on \mathcal{X}_i , where \mathcal{O}_i is an open set containing \mathcal{X}_i ;
- (A5) F is coercive on \mathcal{X} , i.e., $\lim_{\mathbf{x} \in \mathcal{X}, \|\mathbf{x}\| \rightarrow \infty} F(\mathbf{x}) = +\infty$.

These assumptions are rather standard. For example, A3 holds trivially if \mathcal{X} is bounded and ∇f is locally Lipschitz. We remark that in most practical cases the g_i 's are norms or polyhedral functions and A4 is readily satisfied. Finally, A5 guarantees the existence of a solution.

We introduce now our algorithmic asynchronous framework. The asynchronous iterations performed by the workers are given in (1) and (2) [cf. Section 1]. However, the analysis of the algorithm based directly on (1)-(2) is not a simple task. The key idea is then to introduce a “global view” of (1)-(2) that captures through a unified, general, probabilistic model several specific computational architectures/systems and asynchronous modus operandi. The iteration $k \rightarrow k + 1$ is triggered when a block-component i^k of the current \mathbf{x}^k is updated by some worker using (possibly) delayed information $\mathbf{x}^{k-\mathbf{d}^k}$, thus generating the new vector \mathbf{x}^{k+1} . Note that, in this model, the worker that performs the update is immaterial. Given (1) and (2), it is clear that the update $\mathbf{x}^k \rightarrow \mathbf{x}^{k+1}$ is fully determined once i^k and \mathbf{d}^k are specified. In several asynchronous methods, the index i^k is chosen randomly. Even when this is not the case, the values of i^k and \mathbf{d}^k are difficult to preview beforehand, because they depend on several factors which are hard to model mathematically, such as the computational architecture, the specific hardware, the communication protocol employed by the workers, possible hardware failures, etc.. Therefore, we model the sequence of pairs $\{(i^k, \mathbf{d}^k)\}_{k \in \mathbb{N}_+}$ generated by the algorithmic process as a realization of a stochastic process; the probabilistic space associated to this stochastic process is formally introduced in Section 3. The proposed general asynchronous model is summarized in Algorithm 1, which we term Asynchronous FLEXible Parallel Algorithm (AsyFLEXA).

Discussion on Algorithm 1. Several comments are in order.

1. *On the generality of the model:* Algorithm 1 represents a gamut of asynchronous schemes and architectures, all captured in an abstract and unified way by the stochastic process modeling the specific mechanism of generation of the delay vectors \mathbf{d}^k and indices i^k of the blocks to updates. For concreteness, we show next how Algorithm 1 customizes when modeling asynchrony in shared-memory and message passing-based architectures.

Algorithm 1 Asynchronous FLExible Parallel Algorithm (AsyFLEXA)

Initialization: $k = 0$, $\mathbf{x}^0 \in \mathcal{X}$, $\gamma \in (0; 1]$.

while a termination criterion is not met **do**

(S.1) The random variable $(\underline{i}^k, \underline{\mathbf{d}}^k)$ is realized as (i^k, \mathbf{d}^k) ;

(S.2) $\hat{\mathbf{x}}_{i^k}(\mathbf{x}^{k-\mathbf{d}^k})$ is computed:

$$\hat{\mathbf{x}}_{i^k}(\mathbf{x}^{k-\mathbf{d}^k}) \triangleq \underset{\mathbf{x}_{i^k} \in \mathcal{X}_{i^k}}{\operatorname{argmin}} \tilde{f}_{i^k}(\mathbf{x}_{i^k}; \mathbf{x}^{k-\mathbf{d}^k}) + g_{i^k}(\mathbf{x}_{i^k}), \quad (3)$$

(S.3) $\mathbf{x}_{i^k}^k$ is acquired;

(S.4) The block i^k is updated:

$$\mathbf{x}_i^{k+1} = \begin{cases} \mathbf{x}_i^k + \gamma(\hat{\mathbf{x}}_i(\mathbf{x}^{k-\mathbf{d}^k}) - \mathbf{x}_i^k), & \text{if } i = i^k \\ \mathbf{x}_i^k & \text{if } i \neq i^k \end{cases} \quad (4)$$

(S.5) $k \leftarrow k + 1$;

end while

Example 1: Shared-memory systems. Consider a shared-memory system wherein multiple cores update in an asynchronous fashion blocks of the vector \mathbf{x} , stored in a shared memory. An iteration $k \rightarrow k + 1$ of Algorithm 1 is triggered when a core writes the (block) update $\mathbf{x}_{i^k}^{k+1}$ in the shared memory (Step 4). Note that the cores need not know the global iteration index k . No memory lock is assumed, implying that components of the variables may be written by some cores while other components are simultaneously read by others. This *inconsistent read* produces vectors $\mathbf{x}^{k-\mathbf{d}^k} = (x_1^{k-d_1^k}, x_2^{k-d_2^k}, \dots, x_N^{k-d_N^k})$, to be used in the computation of $\hat{\mathbf{x}}_{i^k}$ (Step 2), whose (block) component $\mathbf{x}_i^{k-d_i^k}$ is a (possibly) delayed version of block i read by the core that is going to perform the update. Note that, while $\mathbf{x}_i^{k-d_i^k}$ existed in the shared memory at some point in time, the entire delayed vector $\mathbf{x}^{k-\mathbf{d}^k}$ might have not at any time. Also, in Step 4, it is tacitly assumed that the update of a block is *atomic* (the block is written in the shared memory as a whole) and while a core is writing that block no other core can modify the same block. This is minor requirement, which can be easily enforced in modern architectures either by a block-coordinate lock or using a dual-memory writing approach, see [27, Section 1.2.1].

Figure 1 shows few iterations of the algorithm dynamics in the asynchronous setting described above. The (continuous) time when operations (reading, writing, computation) are performed is indicated in the top horizontal axes whereas the global (discrete) iteration counter is reported in the bottom axes. The asynchronous updates happen as follows. At iteration $k = 3$, Core 3 writes x_1^3 ; therefore, \mathbf{x}^3 differs from \mathbf{x}^2 in the first component. Core 2 locks x_3^2 to quickly read it and perform the linear combination with $\hat{x}_3(\mathbf{x}^{3-\mathbf{d}^3})$ [cf. (4)], and updates x_3 ; therefore \mathbf{x}^4 differs from \mathbf{x}^3 in just the 3rd component. Note that core 2 reads x_3^2 which is equal to x_3^3 , so $d_3^3 = 0$; this is because between the lock and the writing of core 2, no other cores wrote x_3 (core 3 updates x_1). At iteration $k = 5$, core 3 writes x_2^5 . In this case $\mathbf{x}^{4-\mathbf{d}^4}$, used to compute $x_2^5 = (1 - \gamma)x_2^4 + \gamma\hat{x}_2(\mathbf{x}^{4-\mathbf{d}^4})$, is exactly equal to \mathbf{x}^4 , since core 3 reads the vector entirely after the last update, so $\mathbf{d}^4 = \mathbf{0}$. A different situation happens

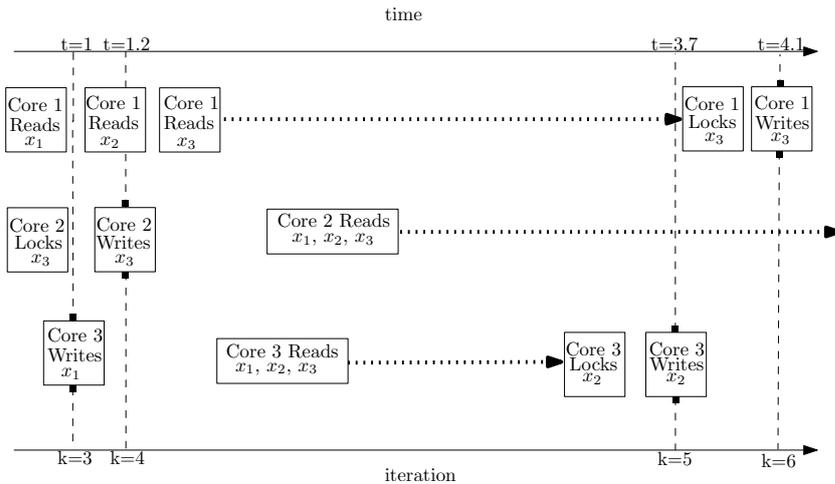


Fig. 1: AsyFLEXA modeling block asynchronous updates in a shared-memory system: three cores, vector variables $\mathbf{x} \in \mathbb{R}^3$, scalar blocks ($n_i = 1$, for all i).

when iteration $k = 6$ is triggered by core 1: the vector used by the core to perform its update is $\mathbf{x}^{5-\mathbf{d}^5}$, which is such that $x_1^{5-\mathbf{d}^5} = x_1^2 \neq x_1^3$, $x_2^{5-\mathbf{d}^5} = x_2^2$, and $x_3^{5-\mathbf{d}^5} = x_3^4 \neq x_3^2$; therefore, $\mathbf{x}^{5-\mathbf{d}^5}$ never existed in the shared memory at any time. It can be seen that the vector of delays at $k = 6$ reads $\mathbf{d}^5 = (3, 1, 0)^T$. Note that the delay vector \mathbf{d}^k used at a given iteration may not be unique: different values for the components d_i^k may produce the same delayed vector $\mathbf{x}^{k-\mathbf{d}^k}$. For instance, in the example above, the vector $(2, 1, 0)^T$ could have been used in place of \mathbf{d}^5 .

Example 2: Message passing systems. Consider a message passing-based system: multiple computational units (e.g., clouds, cluster computers) are connected through a (directed) graph, modeling the communication pattern among the units. Suppose that every worker has in charge the update of a set of block variables, partitioned among all the workers. In this setting, Algorithm 1 still models asynchronous updates and communications. There is no shared memory; every worker updates its own variables writing its own local memory, and then broadcasts its updates to its neighbors, according to a given protocol, which specifies, for instance, how often the communications will happen and with whom. In this setting, $\mathbf{x}^{k-\mathbf{d}^k}$ corresponds to the most recent information a worker has received from the others at the time of its update.

2. *On the surrogate functions \tilde{f}_i .* A degree of freedom offered by the proposed framework is the choice of the surrogate function used in the subproblems (3) solved by the workers at each iteration. We consider the following general class of surrogate functions (we denote by $\nabla \tilde{f}_i$ the partial gradient of \tilde{f}_i with respect to the first argument).

Assumption B (On the surrogate functions \tilde{f}_i 's). Given $\tilde{f}_i : \mathcal{X}_i \times \mathcal{X} \rightarrow \mathbb{R}$, with $i \in \mathcal{N}$, we assume:

- (B1) $\tilde{f}_i(\bullet; \mathbf{y})$ is C^1 on an open set containing \mathcal{X}_i , and $c_{\tilde{f}}$ -strongly convex on \mathcal{X}_i , for all $\mathbf{y} \in \mathcal{X}$;
- (B2) $\nabla \tilde{f}_i(\mathbf{y}_i; \mathbf{y}) = \nabla_{\mathbf{y}_i} f(\mathbf{y})$, for all $\mathbf{y} \in \mathcal{X}$;
- (B3) $\nabla \tilde{f}_i(\mathbf{y}_i; \bullet)$ is L_B -Lipschitz continuous on \mathcal{X} , for all $\mathbf{y}_i \in \mathcal{X}_i$;
- (B4) $\nabla \tilde{f}_i(\bullet; \mathbf{y})$ is L_E -Lipschitz continuous on \mathcal{X}_i , for all $\mathbf{y} \in \mathcal{X}$.

The surrogate $\tilde{f}_i(\bullet; \mathbf{x}^k)$ should be regarded as a (simple) strongly convex local approximation of f around $\mathbf{x}^k \in \mathcal{X}$, that preserves the first order properties of f . Finding a surrogate \tilde{f}_i that satisfies Assumption B is in general non difficult; in any case, one can always choose $\tilde{f}_i(\mathbf{x}_i; \mathbf{x}^k) = \nabla_{\mathbf{x}_i} f(\mathbf{x}^k)^\top (\mathbf{x}_i - \mathbf{x}_i^k) + \beta \|\mathbf{x}_i - \mathbf{x}_i^k\|_2^2$, where β is a positive constant, which leads to the classical proximal-gradient update. However, having the possibility to use a different \tilde{f}_i may be useful to exploit some potential structure in the problem; of course, a trade-off is expected: the more complex the \tilde{f}_i , the more information will be retained in $\hat{\mathbf{x}}_i(\mathbf{x}^k)$, but also the more intensive its computation is expected to be. On the other hand, the solution of more complex subproblems will in general decrease the number of information exchanges in the system, which may be a key advantage in many applications. Some valid instances of \tilde{f}_i 's going beyond the proximal-gradient choice are discussed next; we refer the interested reader to [12, 13] for further examples.

- If $f(\mathbf{x}_1, \dots, \mathbf{x}_N)$ is block-wise uniformly convex, instead of linearizing f one can exploit a second-order approximation and set $\tilde{f}_i(\mathbf{x}_i; \mathbf{x}^k) = f(\mathbf{x}^k) + \nabla_{\mathbf{x}_i} f(\mathbf{x}^k)^\top (\mathbf{x}_i - \mathbf{x}_i^k) + \frac{1}{2} (\mathbf{x}_i - \mathbf{x}_i^k)^\top \nabla_{\mathbf{x}_i \mathbf{x}_i}^2 f(\mathbf{x}^k) (\mathbf{x}_i - \mathbf{x}_i^k) + \beta \|\mathbf{x}_i - \mathbf{x}_i^k\|_2^2$;
- In the same setting as above, one can also better preserve the partial convexity of f and set $\tilde{f}_i(\mathbf{x}_i; \mathbf{x}^k) = f(\mathbf{x}_i, \mathbf{x}_{-i}^k) + \beta \|\mathbf{x}_i - \mathbf{x}_i^k\|_2^2$, where $\mathbf{x}_{-i} \triangleq (\mathbf{x}_1, \dots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \dots, \mathbf{x}_N)$;
- As a last example, suppose that f is the difference of two convex functions $f^{(1)}$ and $f^{(2)}$, i.e., $f(\mathbf{x}) = f^{(1)}(\mathbf{x}) - f^{(2)}(\mathbf{x})$, one can preserve the partial convexity in f setting $\tilde{f}_i(\mathbf{x}_i; \mathbf{x}^k) = f^{(1)}(\mathbf{x}_i, \mathbf{x}_{-i}^k) - \nabla_{\mathbf{x}_i} f^{(2)}(\mathbf{x}^k)^\top (\mathbf{x}_i - \mathbf{x}_i^k) + \beta \|\mathbf{x}_i - \mathbf{x}_i^k\|_2^2$.

3 AsyFLEXA: Probabilistic Model

In this section, we complete the description of AsyFLEXA, introducing the probabilistic model underlying the generation of the pairs index-delays.

Given Problem (P) and an initial point \mathbf{x}^0 , the pair (i^k, \mathbf{d}^k) in Step 1 of Algorithm 1, for each k , is a realization of a random vector $\underline{\omega}^k \triangleq (i^k, \mathbf{d}^k)$, taking values on $\mathcal{N} \times \mathcal{D}$, where \mathcal{D} is the set of all possible delay vectors. We anticipate that all the delays d_i^k are assumed to be bounded (Assumption C below), i.e., $d_i^k \leq \delta$, for all k and i . Hence, \mathcal{D} is contained in the set of all possible N -length vectors whose components are integers between 0 and δ . Let Ω be the sample space of all the sequences $\omega \triangleq \{(i^k, \mathbf{d}^k)\}_{k \in \mathbb{N}_+}$.¹ We will use the following shorthand notation: we set $\underline{\omega}^{0:k} \triangleq (\underline{\omega}^0, \underline{\omega}^1, \dots, \underline{\omega}^k)$ (the first $k+1$ random variables); $\omega^{0:k} \triangleq (\omega^0, \omega^1, \dots, \omega^k)$ ($k+1$ possible values for the

¹ With a slight abuse of notation, we denote by ω_k the k -th element of the sequence $\omega \in \Omega$, and by ω^k the value taken by the random variable $\underline{\omega}^k$ over ω , i.e. $\underline{\omega}^k(\omega) = \omega^k$.

random variables $\underline{\omega}^{0:k}$; and $\omega_{0:k} \triangleq (\omega_0, \omega_1, \dots, \omega_k)$ (the first $k+1$ elements of ω). We introduce next the probability space that will be used to build our probabilistic model.

The sample space is Ω . To define a σ -algebra on Ω , we consider, for $k \geq 0$ and every $\omega^{0:k} \in \mathcal{N} \times \mathcal{D}$, the cylinder

$$C^k(\omega^{0:k}) \triangleq \{\omega \in \Omega : \omega_{0:k} = \omega^{0:k}\},$$

i.e., $C^k(\omega^{0:k})$ is the subset of Ω of all sequences ω whose first k elements are $\omega^0, \dots, \omega^k$. Let us denote by \mathcal{C}^k the set of all possible $C^k(\omega^{0:k})$ when ω^t , $t = 0, \dots, k$, takes all possible values; note, for future reference, that \mathcal{C}^k is a partition of Ω . Denoting by $\sigma(\mathcal{C}^k)$ the σ -algebra generated by \mathcal{C}^k , define for all k ,

$$\mathcal{F}^k \triangleq \sigma(\mathcal{C}^k) \quad \text{and} \quad \mathcal{F} \triangleq \sigma\left(\bigcup_{t=0}^{\infty} \mathcal{C}^t\right). \quad (5)$$

We have $\mathcal{F}^k \subseteq \mathcal{F}^{k+1} \subseteq \mathcal{F}$ for all k . The latter inclusion is obvious, the former derives easily from the fact that any cylinder in \mathcal{C}^{k-1} can be obtained as a finite union of cylinders in \mathcal{C}^k .

The desired probability space is fully defined once $\mathbb{P}(C^k(\omega^{0:k}))$, the probabilities of all cylinders, are given. These probabilities should satisfy some very natural, minimal consistency properties, namely: (i) the probabilities of the union of a finite number of disjoint cylinders should be equal to the sum of the probabilities assigned to each cylinder; and (ii) suppose that a cylinder $C^k(\omega^{0:k})$ is contained in the union U of a countably infinite number of other cylinders, then $\mathbb{P}(C^k(\omega^{0:k})) \leq P(U)$. Suppose now that such a \mathbb{P} is given. Classical results (see, e.g., [18, Theorem 1.53]) ensure that one can extend these probabilities to a probability measure P over (Ω, \mathcal{F}) , thus defining our working probability space $A \triangleq (\Omega, \mathcal{F}, P)$. By appropriately choosing the probabilities of the cylinders, we can model in a unified way many cases of practical interest; several examples are given in Section 3.1.

Given A , we can finally define the discrete-time, discrete-value stochastic process $\underline{\omega}$, where $\{\underline{\omega}^k(\omega)\}_{k \in \mathbb{N}_+}$ is a sample path of the process. The k -th entry $\underline{\omega}^k(\omega)$ of $\underline{\omega}(\omega)$ —the k -th element of the sequence ω —is a realization of the random vector $\underline{\omega}^k = (i^k, \mathbf{d}^k) : \Omega \mapsto \mathcal{N} \times \mathcal{D}$. This process fully describes the evolution of Algorithm 1. Indeed, given an instance of Problem (P) and a starting point, the trajectories of the variables \mathbf{x}^k and $\mathbf{x}^{k-\mathbf{d}^k}$ are completely determined once a sample path $\{(i^k, \mathbf{d}^k)\}_{k \in \mathbb{N}_+}$ is drawn from $\underline{\omega}$.

Note that the joint probability

$$p_{\underline{\omega}^{0:k}}(\omega^{0:k}) \triangleq \mathbb{P}(\underline{\omega}^{0:k} = \omega^{0:k})$$

is simply the probability of the corresponding cylinder: $C^k(\omega^{0:k})$. We will often need to consider the conditional probabilities $p((i, \mathbf{d}) | \omega^{0:k}) \triangleq \mathbb{P}(\underline{\omega}^{k+1} = (i, \mathbf{d}) | \underline{\omega}^{0:k} = \omega^{0:k})$. Note that we have

$$p((i, \mathbf{d}) | \omega^{0:k}) = \frac{\mathbb{P}(C^{k+1}(\omega^{0:k+1}))}{\mathbb{P}(C^k(\omega^{0:k}))}, \quad (6)$$

where we tacitly assume $p((i, \mathbf{d}) | \boldsymbol{\omega}^{0:k}) = 0$, if $\mathbb{P}(C^k(\boldsymbol{\omega}^{0:k})) = 0$. We remark that these probabilities need not be known in practice to implement the algorithm. They are instead determined based on the particular system (hardware architecture, software implementation, asynchrony, etc.) one is interested to model. Here, we make only some minimal assumptions on these probabilities and stochastic model, as stated next.

Assumption C (On the probabilistic model). Given Algorithm 1 and the stochastic process $\underline{\omega}$, suppose that

(C1) There exists a $\delta \in \mathbb{N}_+$, such that $d_i^k \leq \delta$, for all i and k ;

(C2) For all $i \in \mathcal{N}$ and $\omega \in \Omega$, there exists at least one $t \in [0, \dots, T]$, with $T > 0$, such that

$$\sum_{\mathbf{d} \in \mathcal{D}} p((i, \mathbf{d}) | \boldsymbol{\omega}^{0:k+t-1}) \geq p_{\min}, \quad \text{if } p_{\underline{\omega}^{0:k+t-1}}(\boldsymbol{\omega}^{0:k+t-1}) > 0,$$

for some $p_{\min} > 0$;

(C3) $d_{i^k}^k = 0$, for any $k \geq 0$.

These are quite reasonable assumptions, with very intuitive interpretations. C1 just limits the age of the old information used in the updates. Condition C2 guarantees that every T iterations each block-index i has a non negligible positive probability to be updated. These are minimal requirements that are satisfied in practically all computational environments. The condition $d_{i^k}^k = 0$ means that when a worker updates the i^k -th block, it uses the most recent value of that block-variable. This assumption is automatically satisfied, e.g., in a message passing-based system or in a shared memory-based architecture if the variables are partitioned and assigned to different cores (see Example 6 in Section 3.1). If instead all the cores can update all variables, $d_{i^k}^k = 0$ can be simply enforced by a software lock on the i^k -th block of the shared memory: once a core c has read a block-variable \mathbf{x}_{i^k} , no other core can change it, until c has performed its update. Note that in practice it is very unlikely that this lock affects the performance of the algorithm, since usually the number of cores is much smaller than the number of block-variables. Actually, in some systems, this lock can bring in some benefits. For instance, consider two cores sharing all variables, with one core much faster than the other. A lock on $\mathbf{x}_{i^k}^k$ will prevent potentially much older information to overwrite most recent updates of the faster core. Note also that conditions similar to C3 are required by all block asynchronous methods in the literature but [27]: they take the form of locking the variable to update before performing a prox operation [21].

Remark 1 The knowledge of the probability space A is by no means required from the workers to perform the updates. One need not even specify explicitly the probability distribution; it is sufficient to show that a probability space A satisfying Assumption C exists for the specific system (e.g., computational architecture, asynchronous protocol, etc.) under consideration. We show next how to do so for several schemes of practical interest.

3.1 Examples and special cases

The proposed model encompasses a gamut of practical schemes, some of which are discussed next. It is of note that our framework allows us to analyze in a unified way not only randomized methods, but also deterministic algorithms.

1. Deterministic sequential cyclic BCD: In a deterministic, cyclic method there is only one core that cyclically updates all block-variables; for simplicity we assume the natural order, from 1 to N . Since there is only one core, the reading is always consistent and there are no delays: $\mathcal{D} = \{\mathbf{0}\}$. To represent the cyclic choice it is now enough to assign probability 1 too all cylinders of the type

$$C^k = \{\omega : \omega_0 = (1, \mathbf{0}), \omega_1 = (2, \mathbf{0}), \dots, \omega_k = ((k \bmod N) + 1, \mathbf{0})\}$$

and probability zero to all others. It is easy to see that Assumption C is satisfied. This can be seen as a probabilistic model of the deterministic algorithm in [32]. The consequence however is that, by Theorem 1, convergence can be claimed only in a probabilistic sense (a.s.). This is not surprising, as we are describing a deterministic algorithm as limiting case of a probabilistic model.

2. Randomized sequential BCD: Suppose now that there is only one core selecting at each iteration randomly an index i , with a positive probability. Therefore, at each iteration, $\mathbf{x}^{k-\mathbf{d}^k} = \mathbf{x}^k$ or, equivalently, $\mathcal{D} = \{\mathbf{0}\}$. This scheme can be described by a stochastic process, where the cylinders are assigned arbitrary probabilities but satisfying all the conditions given in previous subsection.

3. Randomized parallel BCD: Suppose that there are C cores and the block-variables are partitioned in C groups I_1, I_2, \dots, I_C ; each set I_c is assigned to one core only, say c . Hence, if core c performs the update at iteration k , all variables $i \in I_c$ satisfy $d_i^k = 0$. Denote by $\mathbf{0}(c), \mathbf{1}(c), \dots$, and $(\mathbf{C} - \mathbf{1})(c)$, $c = 1, \dots, C$, the N -length vectors whose components are zeros in the positions of the block-variables in the set I_c and all 0, 1, $\dots, C - 1$ in the other positions, respectively. Set $\mathcal{D} = \{\mathbf{0}(c), \mathbf{1}(c), \dots, (\mathbf{C} - \mathbf{1})(c), c = 1 \dots, C\}$, and denote by c^k the core performing the update at iteration k . Assign to the cylinders the following probabilities: $\forall i^0, i^1, \dots, i^{2C-1}, \dots \in \mathcal{N}$,

$$\mathbb{P}(C^0((i^0, \mathbf{0}(c^0)))) = 1/N,$$

$$\mathbb{P}(C^1((i^0, \mathbf{0}(c^0)), (i^1, \mathbf{1}(c^1)))) = 1/N^2,$$

...

$$\mathbb{P}(C^{C-1}((i^0, \mathbf{0}(c^0)), (i^1, \mathbf{1}(c^1)), \dots, (i^{C-1}, \mathbf{C} - \mathbf{1}(c^{C-1})))) = 1/N^C,$$

$$\mathbb{P}(C^C((i^0, \mathbf{0}(c^0)), (i^1, \mathbf{1}(c^1)), \dots, (i^{C-1}, \mathbf{C} - \mathbf{1}(c^{C-1})), (i^C, \mathbf{0}(c^C)))) = 1/N^{C+1},$$

$$\mathbb{P}(C^{C+1}((i^0, \mathbf{0}(c^0)), (i^1, \mathbf{1}(c^1)), \dots, (i^C, \mathbf{0}(c^C)), (i^{C+1}, \mathbf{1}(c^{C+1})))) = 1/N^{C+2},$$

...

$$\mathbb{P}(C^{2C-1}((i^0, \mathbf{0}(c^0)), (i^1, \mathbf{1}(c^1)), \dots, (i^{2C-1}, \mathbf{C} - \mathbf{1}(c^{2C-1})))) = 1/N^{2C},$$

...

In words, in the first C iterations (from $k = 0$ to $k = C - 1$), all updates are performed using the same vector $\mathbf{x}^{k-\mathbf{d}^k} = \mathbf{x}^0$; and at each iteration any

index has uniform probability to be selected. This situation is then repeated for the next C iterations, this time using $\mathbf{x}^{k-\mathbf{d}^k} = \mathbf{x}^C$, and so on. This model clearly corresponds to a randomized parallel block-coordinate descent method wherein C cores update C block-variables chosen uniformly at random. Note that Assumption C is trivially satisfied.

The example above clearly shows that defining probabilities by using the cylinders can be quite tedious even in simple cases. Using (6) we can equivalently define the probabilistic model by specifying the conditional probabilities $p((i, \mathbf{d}) | \boldsymbol{\omega}^{0:k})$, which is particularly convenient when at every iteration k the probability that $\boldsymbol{\omega}^k$ takes value (i, \mathbf{d}) is independent of $\boldsymbol{\omega}^{0:k-1}$. We exemplify this alternative approach in the following examples.

4. Asynchronous BCD in shared memory systems: Consider a generic shared memory system, under Assumption C3. Then, the set \mathcal{D} is given by all the N -length vectors whose components are non negative integers between 0 and δ . Suppose that, at every k , all cores select an index uniformly at random, but the probabilities associated with the delays can be different. Then, for every $k \geq 0$, given $\boldsymbol{\omega}^{0:k}$, and $i \in \mathcal{N}$, we have

$$\sum_{\mathbf{d} \in \mathcal{D}} p((i, \mathbf{d}) | \boldsymbol{\omega}^{0:k}) = \frac{1}{N}.$$

This setting is consistent with the one studied in [9, 10, 20, 21].

Our probabilistic model however is more general than that of [9, 10, 20, 21]. For instance, differently from [9, 10, 20, 21], we can easily model scenarios wherein $\sum_{\mathbf{d} \in \mathcal{D}} p((i, \mathbf{d}) | \boldsymbol{\omega}^{0:k})$ are not uniform and/or depend on the iteration and/or on the history of the algorithm. This possibility has important ramifications, since the assumption that the indices are selected uniformly at random is extremely strong and unrealistic. In fact, it is satisfied only if all cores have the same computational power and have access to all variables. This is not the case, in most of practical settings. For instance, consider a computational architecture composed of two CPUs sharing all the variables, with one CPU much faster than the other. If the recent history exhibits iterations with a small value of $\|\mathbf{d}^k\|_\infty$, then it is more likely that the slower core will perform the next update, and vice versa. Similar situations are expected also in other common settings, such as shared memory systems with variable partitioning (see Example 5 below) and message passing-based architectures. This clearly shows that our model captures realistic architectures more faithfully.

5. Asynchronous BCD in shared memory systems with variable partitioning: Consider the setting as in Example 4, but now partition the variables across cores, as described in Example 3. This is the configuration most often used in numerical experiments, since it has proven to be most effective in practice; it also models a message passing architecture. In order to satisfy C3, it is enough to set, for all $\boldsymbol{\omega}^{0:k}$ and $i \in I_c$,

$$p((i, \mathbf{d}) | \boldsymbol{\omega}^{0:k}) = 0, \text{ if some } \mathbf{d}_j \neq 0, j \in I_c.$$

A variant of this setting is the *without replacement* updating scheme considered in the numerical experiments of [21]: the block-variables are partitioned

among the cores and, at each “epoch”, variables in each partition are first randomly shuffled and then updated cyclically by the core. This choice of the updates was shown to be numerically very effective. While [21] cannot provide any theoretical analysis of such a scheme, we can easily cover this case by just merging this example with Example 2.

Other examples: Several other examples can be considered, which we omit because of space limitation. Here we only mention that it is quite straightforward to analyze by our model also “hybrid” systems, which combine somehow two or more examples described above. For instance, consider a cluster computer system wherein the optimization variables are partitioned across the machines; let I_m be the set of variables controlled by machine m and stored in its internal shared memory. The update of the variables in I_m is performed by the processors/cores of machine m according to some shared memory-based asynchronous scheme (e.g., subject to inconsistent read). The information on the variables not in I_m is instead updated through communication with the other processors (message passing)

4 AsyFLEXA: Convergence Results

We present now our main convergence theorem, under Assumptions A-C. The extension to the case of nonconvex constraints is addressed in Section 5.

We will use $\|M_F(\mathbf{x})\|_2$ as a measure of optimality, with

$$M_F(\mathbf{x}) \triangleq \mathbf{x} - \arg \min_{\mathbf{y} \in \mathcal{X}} \left\{ \nabla f(\mathbf{x})^T (\mathbf{y} - \mathbf{x}) + g(\mathbf{y}) + \frac{1}{2} \|\mathbf{y} - \mathbf{x}\|_2^2 \right\}. \quad (7)$$

This is a valid measure of stationarity because $M_F(\mathbf{x})$ is continuous and $\|M_F(\mathbf{x})\|_2 = 0$ if and only if \mathbf{x} is a stationary solution of Problem (P).

To state our major convergence result, we need to introduce first the following intermediate definitions. Recalling the definition of T as in Assumption C2, let $\underline{\mathcal{K}}_i^k$ be the (random) set of iterations between $k - \delta$ and $k + T - 1$ at which the block-variable i has been updated, $\mathcal{K}_i^k \triangleq \{t \in [k - \delta; k + T - 1] \mid \underline{i}^t = i\}$, while $\bar{\mathcal{K}}_i^k$ is the subset of $\underline{\mathcal{K}}_i^k$ containing only the elements of $\underline{\mathcal{K}}_i^k$ (iterations) between $k - \delta$ and $k - 1$. Our convergence results leverage a Lyapunov function \tilde{F} that suitably combines present and past iterates, and it is defined as

$$\tilde{F}(\mathbf{x}^k, \dots, \mathbf{x}^{k-\delta}) = F(\mathbf{x}^k) + \delta \frac{L_f}{2} \left(\sum_{l=k-\delta}^{k-1} (l - (k - 1) + \delta) \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^2 \right), \quad (8)$$

where it is understood that $\mathbf{x}^l = \mathbf{x}^0$, if $l < 0$; therefore, \tilde{F} is well defined for any $k \geq 0$. Note that, by this convention, $\tilde{F}(\mathbf{x}^0, \dots, \mathbf{x}^{0-\delta}) = F(\mathbf{x}^0)$. Furthermore, we also have $F^* \triangleq \min_{\mathbf{x} \in \mathcal{X}} F(\mathbf{x}) \leq \min_{\mathbf{x}^k, \dots, \mathbf{x}^{k-\delta} \in \mathcal{X}} \tilde{F}(\mathbf{x}^k, \dots, \mathbf{x}^{k-\delta})$. We are now ready to state our major convergence result.

Theorem 1 *Let Problem (P) be given, along with Algorithm 1 and the stochastic process $\underline{\omega}$. Let $\{\underline{\mathbf{x}}^k\}_{k \in \mathbb{N}_+}$ be the sequence generated by the algorithm, given*

$\mathbf{x}^0 \in \mathcal{X}$. Suppose that Assumptions A-C hold true and that

$$\gamma < \frac{c_{\bar{f}}}{L_f + \frac{\delta^2 L_f}{2}}. \quad (9)$$

Define K_ϵ to be the first iteration such that $\mathbb{E} (\|M_F(\mathbf{x}^k)\|_2^2) \leq \epsilon$. Then:

- (a) Every limit point of $\{\mathbf{x}^k\}_{k \in \mathbb{N}_+}$ is a stationary solution of (P) a.s.;
 (b) The sequence of objective function values $\{F(\mathbf{x}^k)\}_{k \in \mathbb{N}_+}$ converges a.s.;

$$(c) \quad K_\epsilon \leq \frac{C_1(\gamma, \delta)(T+1)(F(\mathbf{x}^0) - F^*)}{\epsilon} + \frac{C_2(\gamma, \delta)\gamma^2}{\epsilon} \underbrace{\sum_{k=0}^{K_\epsilon} \mathbb{E} \left(\sum_{i=1}^N M_i^k \sum_{t \in \underline{K}_i^k} \left(\tilde{F}(\mathbf{x}^t, \dots, \mathbf{x}^{t-\delta}) - \tilde{F}(\mathbf{x}^{t+1}, \dots, \mathbf{x}^{t+1-\delta}) \right) \right)}_B, \quad (10)$$

where:

$$C_1(\gamma, \delta) \triangleq \frac{2(1 + (1 + L_E)(1 + L_B + L_E) + \gamma^2 N p_{\min} \alpha^{-1} (1 + (L_f + 1)^2))}{\gamma \left(c_{\bar{f}} - \gamma \left(L_f + \frac{\delta^2 L_f}{2} \right) \right) (p_{\min} - p_{\min} \alpha)}, \quad (11)$$

$$C_2(\gamma, \delta) \triangleq \frac{2TL_B(1 + L_B + L_E)}{\gamma \left(c_{\bar{f}} - \gamma \left(L_f + \frac{\delta^2 L_f}{2} \right) \right) (p_{\min} - p_{\min} \alpha)}, \quad (12)$$

α is an arbitrary fixed value in $(0; 1)$, $\underline{M}_i^k \triangleq \max_{l=k, \dots, k+T} |\bar{K}_i^l|$.

Proof See Appendix.

The theorem states that convergence to stationary points occurs a.s. (the objective function values converge too); it also gives an estimate of the number of iterations K_ϵ necessary to enforce $\mathbb{E} (\|M_F(\mathbf{x}^k)\|_2^2) \leq \epsilon$. Convergence is guaranteed if, in particular, the step-size is sufficiently small; the bound (9) makes this precise. Note that if the method is synchronous, $\delta = 0$, the bound in (9), going like the inverse of the Lipschitz constant, becomes the renowned conditions used in many synchronous (proximal-gradient-like) schemes. The term $\delta^2/2$ in the denominator of (9) should then be seen as the price to pay for asynchrony: the larger the possible delay δ , the smaller γ should be to tolerate such delays. Roughly speaking, this means that the more chaotic the computational environment, the more conservative the step should be, and consequently the smaller the steps of the algorithm are.

The interpretation of the bound (10) is not immediate, because of the presence of the term B ; we now elaborate on it. If there exists a (deterministic) bound C on \underline{M}_i^k , i.e., $\underline{M}_i^k \leq C$ for all k and i , then one can write

$$B \leq C \mathbb{E} \left(\sum_{k=0}^{K_\epsilon} \sum_{i=1}^N \sum_{t \in \underline{K}_i^k} \left(\tilde{F}(\mathbf{x}^t, \dots, \mathbf{x}^{t-\delta}) - \tilde{F}(\mathbf{x}^{t+1}, \dots, \mathbf{x}^{t+1-\delta}) \right) \right)$$

	Average Delay	δ	# of cores
Multi-core Machine: Balanced Workload	1.11	3	10
Multi-core Machine: Unbalanced Workload	2.58	28	10
Message Passing System: Balanced Workload	1.87	30	10 per node
Message Passing System: Unbalanced Workload	3.01	36	10 per node

Table 1: Average delay and maximum delay δ for AsyFLEXA, ran on a multi-core machine and on a message passing system.

$$\leq C(T + \delta)(F(\mathbf{x}^0) - F^*).$$

Therefore, (10) can be upper bounded as

$$K_\epsilon \leq \left[C_1(\gamma, \delta) \cdot (T + 1) + C_2(\gamma, \delta) \cdot \gamma^2 \cdot C \cdot (T + \delta) \right] \frac{F(\mathbf{x}^0) - F^*}{\epsilon}. \quad (13)$$

Recalling the definition of \underline{M}_i^k and that $|\bar{K}_i^k|$ is a random variable counting the number of times the index i has been updated in the iteration window $[k - \delta, k - 1]$, $\underline{M}_i^k \leq \delta$ always holds; therefore, one can always take $C = \delta$. Of course this is a very rough approximation: it is hard to expect that in a given time window always the same variable, \bar{i} , is updated and, even if this were the case, all other \underline{M}_i^k , $i \neq \bar{i}$, would be 0 and not δ . Consider for example the commonly analyzed “uniform case” where the processing of every block-variable requires the same time. In this case one can reasonably take $C = 1$ in (13) *independently of the number of workers*.

This intuition is corroborated by our experiments, which are summarized in Table 1. AsyFLEXA was ran on two different architectures, namely: a shared-memory system with 10 cores, and a message passing architecture composed of two nodes, with 10 cores each. Two LASSO problems with 10,000 variables each were considered, and the variables were equally partitioned across the workers. In the first LASSO instance, the Hessian matrix was a dense matrix, which models situations where the workload is equally distributed across the workers. In the second LASSO problem, the Hessian matrix had many sparse rows, to create some unbalancedness in the workers’ workload. Table 1 shows the empirical average delay (the average is taken over the components of the delay vector and time) and the maximum delay δ , estimated in 500 epochs (one epoch is triggered when all blocks have been updated once). As expected, δ is much larger than the experienced average delay, confirming that (13) with $C = \delta$ is a very conservative bound. While C can always be pessimistically upper bounded by δ , a tighter value can be found by tailoring the analysis to the specific problem and architecture under consideration.

Finally, we remark the importance of the use, in the complexity analysis, of the \underline{M}_i^k , counting the number of times the index i has been updated in a certain iteration window. The use of these variables seems to be a new feature of our analysis. While getting a sharp estimate for the upper bound C may be difficult in practice, the bound (10) gives a good insight into the elements that really influence the algorithm, showing that what really matters, in some expressions appearing in (10), is not δ , but the usually much smaller number of times the blocks are actually updated. The use of these variables allows us

to get a sharper bound with respect to the case in which one sets $C = \delta$. From this point of view, we believe that typical upper bounds, as those obtained in [9, 10, 21, 22, 27], where δ is the only considered “delay”, do not give an accurate description of the actual worst-case scenario.

Almost linear speedup: To study the speedup achievable by the proposed method, we make two simplifying assumptions, consistent with those made in the literature, namely: (a) δ is proportional to the number of workers, which is reasonable in “symmetric” situations; and (b) K_ϵ is a good proxy for the number of iterations performed by the algorithm to reach the desired accuracy. Choose the stepsize γ to be small enough so that (9) is always satisfied in the range of values of δ under consideration; then $C_1(\gamma, \delta)$ and $C_2(\gamma, \delta)$ can be taken to be constants. Consider now the two summands in square brackets in (13). Without the second term, one would have ideal linear speed-up. However, since one can expect the second term to be much smaller than the first (at least when δ is not large), an almost linear speed-up can be anticipated. In fact, by (11) and (12), the second term is smaller than the first one, if γ is sufficiently small. In practice, of course, the speed up and in particular the range of the number of workers for which linear speedup is expected, will be problem and architecture dependent.

5 Nonconvex Constraints

In this section, we remove the assumption that all constraints are convex, and study the following more general nonconvex constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & F(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^N g_i(\mathbf{x}_i) \\ & \left. \begin{aligned} \mathbf{x}_i \in \mathcal{X}_i, \quad & i = 1, \dots, N, \\ c_1(\mathbf{x}_1) \leq 0, \dots, c_N(\mathbf{x}_N) \leq 0, \end{aligned} \right\} \triangleq \mathcal{K} \end{aligned} \quad (\text{P}')$$

where $c_i(\mathbf{x}_i) \leq 0$ are nonconvex private constraints, with $c_i : \mathcal{O}_i \rightarrow \mathbb{R}^{m_i}$, and \mathcal{O}_i denoting an open set containing \mathcal{X}_i ; let also define $\mathcal{K}_i \triangleq \{\mathbf{x}_i \in \mathcal{X}_i : c_i(\mathbf{x}_i) \leq 0, \}$. Note that $c_i(\mathbf{x}_i)$ is a vector function, whose individual component is denoted by $c_{i,j}$, with $j = 1, \dots, m_i$. Problem (P') is motivated by several applications in signal processing, machine learning, and networking; see, [31] and references therein for some concrete examples.

To deal with nonconvex constraints, we need some *regularity* of the constraint functions. Anticipating that all c_i are assumed to be C^1 on \mathcal{X}_i , we will use the Mangasarian-Fromovitz Constraint Qualification (MFCQ).

Definition 1 A point $\bar{\mathbf{x}} \in \mathcal{K}$ satisfies the MFCQ if the following implication is satisfied:

$$\left. \begin{aligned} \mathbf{0} \in \sum_{i=1}^N \sum_{j \in \bar{J}_i} \mu_{i,j} \nabla_{\mathbf{x}} c_{i,j}(\bar{\mathbf{x}}_i) + N_{\mathcal{X}}(\bar{\mathbf{x}}) \\ \mu_{i,j} \geq 0, \forall j \in \bar{J}_i, \forall i \in \mathcal{N} \end{aligned} \right\} \Rightarrow \mu_{i,j} = 0, \forall j \in \bar{J}_i, \forall i \in \mathcal{N}, \quad (14)$$

where $N_{\mathcal{X}}(\bar{\mathbf{x}}) \triangleq \{\mathbf{z} \in \mathcal{X} : \mathbf{z}^\top(\mathbf{y} - \bar{\mathbf{x}}) \leq 0, \forall \mathbf{y} \in \mathcal{X}\}$ is the normal cone to \mathcal{X} at $\bar{\mathbf{x}}$, and $\bar{J}_i \triangleq \{j : c_{i,j}(\bar{\mathbf{x}}_i) = 0\}$ is the index set of nonconvex constraints that are active at $\bar{\mathbf{x}}_i$.

We study Problem (P') under the following assumptions.

Assumption A' (On the problem model). Suppose that

- (A1') Each set $\mathcal{X}_i \subseteq \mathbb{R}^{n_i}$ is nonempty, closed, and convex;
- (A2') $f : \mathcal{O} \rightarrow \mathbb{R}$ is C^1 , where \mathcal{O} is an open set containing \mathcal{K} ;
- (A3') $\nabla_{\mathbf{x}_i} f$ is L_f -Lipschitz continuous on \mathcal{K} ;
- (A4') Each $g_i : \mathcal{O}_i \rightarrow \mathbb{R}$ is convex, possibly nonsmooth, and L_g -Lipschitz continuous on \mathcal{X}_i , where \mathcal{O}_i is an open set containing \mathcal{X}_i ;
- (A5') \mathcal{K} is a compact set;
- (A6') Each $c_{i,j} : \mathcal{O}_i \rightarrow \mathbb{R}$ is C^1 ;
- (A7') All feasible points of problem (P') satisfy the MFCQ.

Assumptions A1'-A4' are a duplication of A1-A4, repeated here for ease of reference; A5' is stronger than A5, and made here for the sake of simplicity (one could relax it with A5); and A6' is a standard differentiability assumption on the non convex constraints $c_{i,j}$.

AsyFLEXA-NCC: We are now ready to introduce our asynchronous algorithmic framework for (P'), termed AsyFLEXA-NCC (where NCC stands for Non Convex Constraints). The method is still given by Algorithm 1, with the only difference that now also the nonconvex constraints are replaced by suitably chosen convex approximations; the probabilistic model concerning the choice of the the pair index-delays is the same as the one we used in the case of convex constraints, see Section 3. More specifically, AsyFLEXA-NCC is given by Algorithm 1 wherein the subproblem (3) in Step 2 is replaced by

$$\hat{\mathbf{x}}_{i^k}(\mathbf{x}^{k-d^k}) \triangleq \arg \min_{\mathbf{x}_{i^k} \in \mathcal{K}_{i^k}(\mathbf{x}_{i^k}^k)} \left\{ \tilde{F}_{i^k}(\mathbf{x}_i; \mathbf{x}^{k-d^k}) \triangleq \tilde{f}_{i^k}(\mathbf{x}_i; \mathbf{x}^{k-d^k}) + g_{i^k}(\mathbf{x}_i) \right\}, \quad (15)$$

where \tilde{f}_{i^k} is defined as in (3); $\mathcal{K}_{i^k}(\mathbf{x}_{i^k}^k)$ is a convex approximation of \mathcal{K}_{i^k} at \mathbf{x}^{k-d^k} , defined as

$$\mathcal{K}_{i^k}(\mathbf{x}_{i^k}^k) \triangleq \{\mathbf{x}_i \in \mathcal{X}_{i^k} : \tilde{c}_{i^k,j}(\mathbf{x}_i; \mathbf{x}_{i^k}^k) \leq 0, j = 1, \dots, m_{i^k}\};$$

and $\tilde{c}_{i^k,j} : \mathcal{X}_{i^k} \times \mathcal{K}_{i^k} \rightarrow \mathbb{R}$ is a suitably chosen surrogate of $c_{i^k,j}$. Note that \mathcal{K}_{i^k} depends on $\mathbf{x}_{i^k}^k$ and not on $\mathbf{x}_{i^k}^{k-d^k}$, because of Assumption C3 ($d_{i^k}^k = 0$).

The surrogate functions $\tilde{c}_{i^k,j}$ can be chosen according to the following assumptions ($\nabla \tilde{c}_{i^k,j}$ below denotes the partial gradient of $\tilde{c}_{i^k,j}$ with respect to the first argument).

Assumption D (On the surrogate functions $\tilde{c}_{i^k,j}$'s).

- (D1) Each $\tilde{c}_{i^k,j}(\bullet; \mathbf{y})$ C^1 on an open set containing \mathcal{X}_{i^k} , and convex on \mathcal{X}_{i^k} for all $\mathbf{y} \in \mathcal{K}_{i^k}$;
- (D2) $\tilde{c}_{i^k,j}(\mathbf{y}; \mathbf{y}) = c_{i^k,j}(\mathbf{y})$, for all $\mathbf{y} \in \mathcal{K}_{i^k}$;
- (D3) $c_{i^k,j}(\mathbf{z}) \leq \tilde{c}_{i^k,j}(\mathbf{z}; \mathbf{y})$ for all $\mathbf{z} \in \mathcal{X}_{i^k}$ and $\mathbf{y} \in \mathcal{K}_{i^k}$;

- (D4) $\tilde{c}_{i,j}(\bullet; \bullet)$ is continuous on $\mathcal{X}_i \times \mathcal{K}_i$;
- (D5) $\nabla_{\mathbf{y}_i} c_{i,j}(\mathbf{y}) = \nabla \tilde{c}_{i,j}(\mathbf{y}; \mathbf{y})$, for all $\mathbf{y} \in \mathcal{K}_i$;
- (D6) $\nabla \tilde{c}_{i,j}(\bullet; \bullet)$ is continuous on $\mathcal{X}_i \times \mathcal{K}_i$;
- (D7) Each $\tilde{c}_{i,j}(\bullet; \bullet)$ is Lipschitz continuous on $\mathcal{X}_i \times \mathcal{K}_i$.

Roughly speaking, Assumption D requires $\tilde{c}_{i,j}$ to be an upper convex approximation of $c_{i,j}$ having the same gradient of $c_{i,j}$ at the base point \mathbf{y} . Finding such approximations is less difficult than it might seem at a first sight. Two examples are given below, while we refer the reader to [12,31] for a richer list.

- Suppose $c_{i,j}$ has a $L_{\nabla c_{i,j}}$ -Lipschitz continuous gradient on the (compact) set \mathcal{K}_i . By the Descent Lemma [2, Proposition A32], the following convex approximation satisfies Assumption D:

$$\tilde{c}_{i,j}(\mathbf{x}; \mathbf{y}) \triangleq c_{i,j}(\mathbf{y}) + \nabla_{\mathbf{x}} c_{i,j}(\mathbf{y})^T (\mathbf{x} - \mathbf{y}) + \frac{L_{\nabla c_{i,j}}}{2} \|\mathbf{x} - \mathbf{y}\|_2^2 \geq c_{i,j}(\mathbf{x}).$$

- Suppose that $c_{i,j}$ has a DC structure, that is, $c_{i,j}(\mathbf{x}) = c_{i,j}^+(\mathbf{x}) - c_{i,j}^-(\mathbf{x})$, $c_{i,j}^+$ and $c_{i,j}^-$ are two convex and continuously differentiable functions. By linearizing the concave part $-c_{i,j}^-$ and keeping the convex part $c_{i,j}^+$ unchanged, we obtain the following convex upper approximation of $c_{i,j}$ that satisfies Assumption D:

$$\tilde{c}_{i,j}(\mathbf{x}; \mathbf{y}) \triangleq c_{i,j}^+(\mathbf{x}) - c_{i,j}^-(\mathbf{y}) - \nabla_{\mathbf{x}} c_{i,j}^-(\mathbf{y})^T (\mathbf{x} - \mathbf{y}) \geq c_{i,j}(\mathbf{x}).$$

Note that the former example is quite general and in principle can be applied to practically all constraints, even if it could be numerically undesirable if $L_{\nabla c_{i,j}}$ is too large; the latter example covers, in a possibly more suitable way, the case of concave constraints.

AsyFLEXA-NCC: Convergence. In order to gauge convergence, we redefine the stationarity measure M_F , to account for the presence of nonconvex constraints. We use $\|M_F^c(\mathbf{x})\|_2$, with

$$M_F^c(\mathbf{x}) = \mathbf{x} - \arg \min_{\mathbf{y} \in \mathcal{K}_1(\mathbf{x}_1) \times \dots \times \mathcal{K}_N(\mathbf{x}_N)} \{ \nabla f(\mathbf{x})^T (\mathbf{y} - \mathbf{x}) + g(\mathbf{y}) + \frac{1}{2} \|\mathbf{y} - \mathbf{x}\|_2^2 \}.$$

It is a valid merit function: $\|M_F^c(\mathbf{x})\|_2$ is continuous and is zero only at stationary solutions of (P') [30].

Theorem 2 *Let Problem (P') be given, along with AsyFLEXA-NCC and the stochastic process $\underline{\omega}$. Let $\{\underline{\mathbf{x}}^k\}_{k \in \mathbb{N}_+}$ be the sequence generated by the algorithm, given $\mathbf{x}^0 \in \mathcal{K}$. Suppose that Assumptions A', B-D hold and that γ is chosen as in (9). Define K_ϵ to be the first iteration such that $\mathbb{E}(\|M_F^c(\underline{\mathbf{x}}^k)\|_2^2) \leq \epsilon$. Then: i) $\underline{\mathbf{x}}^k \in \mathcal{K}_1(\underline{\mathbf{x}}_1^k) \times \dots \times \mathcal{K}_N(\underline{\mathbf{x}}_N^k) \subseteq \mathcal{K}$ for all $k \geq 0$ (iterate feasibility); and ii) all results in Theorem 1 hold with M_F replaced by M_F^c .*

Proof See Appendix 7.3.

We are aware of only one other BCD-asynchronous method [9,10] able to deal with nonconvex constraints. This method requires the ability to find global minima of *nonconvex* subproblems while our scheme does not suffer from this drawbacks, as it only calls for the solution of strongly convex subproblems. On

the other hand, it needs a feasible starting point and the ability to build approximations $\tilde{c}_{i,j}$ satisfying Assumption D. While our requirements are easier to be met in practice (and our analysis is based on a grounded probabilistic model), we think that the two approaches complement each other and may cover different applications.

6 Conclusions

We proposed a novel model for the parallel block-descent asynchronous minimization of the sum of a nonconvex smooth function and a convex nonsmooth one, subject to nonconvex constraints. Our model captures the essential features of modern multi-core architectures by providing a more realistic probabilistic description of asynchrony that that offered by the state of the art. Building on our new probabilistic model, we proved sublinear convergence rate of our algorithm and a near linear speedup when the number of workers is not too large. While we performed some simple numerical tests to validate some of our theoretical findings, extensive simulations are beyond the scope of this paper, and will be the subject of a subsequent work. Some preliminary numerical results can be found in [5].

References

1. G. M. Baudet. Asynchronous iterative methods for multiprocessors. *Journal of the ACM (JACM)*, 25(2):226–244, 1978.
2. D. P. Bertsekas and J. N. Tsitsiklis. Parallel and distributed computation: numerical methods. *Prentice-Hall Englewood Cliffs, NJ*, 23, 1989.
3. L. Cannelli, F. Facchinei, V. Kungurtsev, and G. Scutari. Asynchronous parallel algorithms for nonconvex big-data optimization - part i: Model and convergence. *arXiv preprint arXiv:1607.04818*, 2016.
4. L. Cannelli, F. Facchinei, V. Kungurtsev, and G. Scutari. Asynchronous parallel algorithms for nonconvex big-data optimization. part ii: Complexity and numerical results. *arXiv preprint arXiv:1701.04900*, 2017.
5. L. Cannelli, F. Facchinei, V. Kungurtsev, and G. Scutari. Asynchronous parallel nonconvex large-scale optimization. *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 4706–4710, 2017.
6. L. Cannelli, G. Scutari, F. Facchinei, and V. Kungurtsev. Parallel asynchronous lock-free algorithms for nonconvex big-data optimization. *Signals, Systems and Computers, 2016 50th Asilomar Conference on*, pages 1009–1013, 2016.
7. D. Chazan and W. Miranker. Chaotic relaxation. *Linear algebra and its applications*, 2(2):199–222, 1969.
8. A. Daneshmand, F. Facchinei, V. Kungurtsev, and G. Scutari. Hybrid random/deterministic parallel algorithms for convex and nonconvex big data optimization. *IEEE Transactions on Signal Processing*, 63(15):3914–3929, 2015.
9. D. Davis. The asynchronous palm algorithm for nonsmooth nonconvex problems. *arXiv preprint arXiv:1604.00526*, 2016.
10. D. Davis, B. Edmonds, and M Udell. The sound of apalm clapping: Faster nonsmooth nonconvex optimization with stochastic asynchronous palm. *Advances in Neural Information Processing Systems*, pages 226–234, 2016.
11. R. Durrett. Probability: theory and examples. *Cambridge university press*, 2010.
12. F. Facchinei, L. Lampariello, and G. Scutari. Feasible methods for nonconvex nonsmooth problems with applications in green communications. *Mathematical Programming*, 164(1):1–36, 2016.
13. F. Facchinei, G. Scutari, and S. Sagratella. Parallel selective algorithms for nonconvex big data optimization. *IEEE Transactions on Signal Processing*, 63(7):1874–1889, 2015.

14. A. Frommer and D. B. Szyld. On asynchronous iterations. *Journal of computational and applied mathematics*, 123(1):201–216, 2000.
15. M. Hong. A distributed, asynchronous and incremental algorithm for nonconvex optimization: An admm approach. *IEEE Transactions on Control of Network System*, PP(99), 2017.
16. Z. Huo and H. Huang. Asynchronous stochastic gradient descent with variance reduction for non-convex optimization. *arXiv preprint arXiv:1604.03584*, 2016.
17. F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem. Asynchronous distributed optimization using a randomized alternating direction method of multipliers. *52nd IEEE Conference on Decision and Control*, pages 3671–3676, 2013.
18. A. Klenke. Probability theory: a comprehensive course. *Springer Science & Business Media*, 2013.
19. R. Leblond, F. Pedregosa, and S. Lacoste-Julien. ASAGA: Asynchronous parallel SAGA. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 46–54, 2017.
20. X. Lian, Y. Huang, Y. Li, and J. Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 2719–2727, 2015.
21. J. Liu and S. J. Wright. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *SIAM Journal on Optimization*, 25(1):351–376, 2015.
22. J. Liu, S. J. Wright, C. Ré, V. Bittorf, and S. Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. *The Journal of Machine Learning Research*, 16(1):285–322, 2015.
23. H. Mania, X. Pan, D. Papailiopoulos, B. Recht, K. Ramchandran, and M. I. Jordan. Perturbed iterate analysis for asynchronous stochastic optimization. *arXiv:1507.06970*, 2016.
24. A. Nedić, D. P. Bertsekas, and V. S. Borkar. Distributed asynchronous incremental subgradient methods. *Studies in Computational Mathematics*, 8(C):381–407, 2001.
25. F. Niu, B. Recht, C. Re, and S. J. Wright. Hogwild: a lock-free approach to parallelizing stochastic gradient descent. *Advances in Neural Information Processing Systems*, pages 693–701, 2011.
26. F. Pedregosa, R. Leblond, and S. Lacoste-Julien. Breaking the nonsmooth barrier: A scalable parallel method for composite optimization. *arXiv preprint arXiv:1707.06468*, 2017.
27. Z. Peng, Y. Xu, M. Yan, and W. Yin. Arock: an algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing*, 38(5):A2851–A2879, 2016.
28. Z. Peng, Y. Xu, M. Yan, and W. Yin. On the convergence of asynchronous parallel iteration with arbitrary delays. *arXiv preprint arXiv:1612.04425*, 2016.
29. H. Robbins and D. Siegmund. A convergence theorem for non negative almost supermartingales and some applications. *Herbert Robbins Selected Papers*, pages 111–135, 1985.
30. G. Scutari, F. Facchinei, and L. Lampariello. Parallel and distributed methods for constrained nonconvex optimization—part i: Theory. *IEEE Transactions on Signal Processing*, 65(8):1929–1944, April 2017.
31. G. Scutari, F. Facchinei, L. Lampariello, S. Sardellitti, and P. Song. Parallel and distributed methods for constrained nonconvex optimization—part ii: Applications in communications and machine learning. *IEEE Transactions on Signal Processing*, 65(8):1945–1960, April 2017.
32. G. Scutari, F. Facchinei, P. Song, D. P. Palomar, and J.-S. Pang. Decomposition by partial linearization: Parallel optimization of multi-agent systems. *IEEE Transactions on Signal Processing*, 62(3):641–656, 2014.
33. R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.
34. P. Tseng. On the rate of convergence of a partially asynchronous gradient projection algorithm. *SIAM Journal on Optimization*, 1(4):603–619, 1991.
35. E. Wei and A. Ozdaglar. On the $\mathcal{O}(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers. *Global Conference on Signal and Information Processing (GlobalSIP)*, pages 551–554, 2013.

36. S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.
37. W. H. Young. On classes of summable functions and their fourier series. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 87(594):225–229, 1912.
38. H. Yun, H.-F. Yu, C.-J. Hsieh, SVN Vishwanathan, and I. Dhillon. Nomad: Non-locking, stochastic multi-machine algorithm for asynchronous and decentralized matrix completion. *Proceedings of the VLDB Endowment*, 7(11):975–986, 2014.

7 Appendix

7.1 Preliminaries

Hereafter, we simplify the notation using $\tilde{\mathbf{x}}^k \triangleq \mathbf{x}^{k-\mathbf{d}^k}$.

1. On conditional probabilities. In our developments, we will consider the conditional expectation of random variables \mathbf{Z} on Ω of the type $\mathbb{E}(\mathbf{Z}|\mathcal{F}^k)$. The following simple fact holds.

Proposition 1 *Let \mathbf{Z} be a random variable defined on Ω , and let \mathcal{F}^k be defined in (5). Then*

$$\mathbb{E}(\mathbf{Z}|\mathcal{F}^k) = \sum_{(i, \mathbf{d}) \in \mathcal{N} \times \mathcal{D}} p((i, \mathbf{d}) | \boldsymbol{\omega}^{0:k}) \mathbf{Z}((i, \mathbf{d}), \boldsymbol{\omega}^{0:k}). \quad (16)$$

Proof Recall that \mathcal{F}^k is the σ -algebra generated by \mathcal{C}^k , which is a finite partition of Ω . Therefore, one can write [11, Example 5.1.3]

$$\mathbb{E}(\mathbf{Z}|\mathcal{F}^k) = \frac{\mathbb{E}(\mathbf{Z}; \mathcal{C}^k(\boldsymbol{\omega}^{0:k}))}{\mathbb{P}(\mathcal{C}^k(\boldsymbol{\omega}^{0:k}))}.$$

The thesis follows readily from (6) and the fact that \mathbf{Z} depends only on $\boldsymbol{\omega}^{0:k+1}$ and takes a finite number of values.

2. Properties of the best response $\hat{\mathbf{x}}(\cdot)$. We introduce next some basic properties of the best-response maps defined in (3) and (15).

Proposition 2 ([13]) *Given the best-response map $\hat{\mathbf{x}}(\cdot) \triangleq (\hat{\mathbf{x}}_i(\cdot))_{i=1}^N$, with $\hat{\mathbf{x}}_i(\cdot)$ defined in (3). Under Assumptions A-B, the following hold.*

(a) [Optimality]: For any $i \in \mathcal{N}$ and $\mathbf{y} \in \mathcal{X}$,

$$(\hat{\mathbf{x}}_i(\mathbf{y}) - \mathbf{y}_i)^T \nabla_{\mathbf{y}_i} f(\mathbf{y}) + g_i(\hat{\mathbf{x}}_i(\mathbf{y})) - g_i(\mathbf{y}_i) \leq -c_{\bar{f}} \|\hat{\mathbf{x}}_i(\mathbf{y}) - \mathbf{y}_i\|_2^2; \quad (17)$$

(b) [Lipschitz continuity]: For any $i \in \mathcal{N}$ and $\mathbf{y}, \mathbf{z} \in \mathcal{X}$,

$$\|\hat{\mathbf{x}}_i(\mathbf{y}) - \hat{\mathbf{x}}_i(\mathbf{z})\|_2 \leq L_{\hat{\mathbf{x}}} \|\mathbf{y} - \mathbf{z}\|_2, \quad (18)$$

with $L_{\hat{\mathbf{x}}} = L_B/c_{\bar{f}}$;

(c) [Fixed-point characterization]: The set of fixed-points of $\hat{\mathbf{x}}(\cdot)$ coincides with the set of stationary solutions of Problem (P). Therefore $\hat{\mathbf{x}}(\cdot)$ has at least one fixed point.

Proposition 3 ([30]) *Given the best-response map $\hat{\mathbf{x}}(\cdot) \triangleq (\hat{\mathbf{x}}_i(\cdot))_{i=1}^N$, with $\hat{\mathbf{x}}_i(\cdot)$ defined in (15). Under Assumptions A'-B-D, the following hold.*

(a) [Optimality]: For any $i \in \mathcal{N}$ and $\mathbf{y} \in \mathcal{K}$,

$$(\hat{\mathbf{x}}_i(\mathbf{y}) - \mathbf{y}_i)^T \nabla_{\mathbf{y}_i} f(\mathbf{y}) + g_i(\hat{\mathbf{x}}_i(\mathbf{y})) - g_i(\mathbf{y}_i) \leq -c_{\bar{f}} \|\hat{\mathbf{x}}_i(\mathbf{y}) - \mathbf{y}_i\|_2^2; \quad (19)$$

(b) [Lipschitz continuity]: For any $i \in \mathcal{N}$ and $\mathbf{y}, \mathbf{z} \in \mathcal{K}$,

$$\|\hat{\mathbf{x}}_i(\mathbf{y}) - \hat{\mathbf{x}}_i(\mathbf{z})\|_2 \leq \tilde{L}_{\hat{\mathbf{x}}} \|\mathbf{y} - \mathbf{z}\|_2^{1/2}, \quad (20)$$

with $\tilde{L}_{\hat{\mathbf{x}}} > 0$.

3. Young's Inequality [37]. For any $\alpha, \mu_1, \mu_2 > 0$, there holds

$$\mu_1 \mu_2 \leq \frac{1}{2} (\alpha \mu_1^2 + \alpha^{-1} \mu_2^2). \quad (21)$$

4. Representation of $\tilde{\mathbf{x}}^k$. Since at each iteration only one block of variables is updated, $\tilde{\mathbf{x}}^k$ can be written as

$$\tilde{\mathbf{x}}_i^k = \mathbf{x}_i^k + \sum_{l \in \mathcal{K}_i^k} (\mathbf{x}_i^l - \mathbf{x}_i^{l+1}), \quad (22)$$

where \mathcal{K}_i^k is defined in Section 4.

7.2 Proof of Theorem 1

In this section, the best-response map $\hat{\mathbf{x}}(\cdot)$ is the one defined in (3).

For any given realization $\omega \in \Omega$ and $k \geq 0$, the following holds:

$$\begin{aligned} F(\mathbf{x}^{k+1}) &= f(\mathbf{x}^{k+1}) + g(\mathbf{x}^{k+1}) \\ &\stackrel{(a)}{=} f(\mathbf{x}^{k+1}) + \sum_{i \neq i^k} g_i(\mathbf{x}_i^k) + g_{i^k}(\mathbf{x}_{i^k}^{k+1}) \\ &\stackrel{(b)}{\leq} f(\mathbf{x}^k) + \gamma \nabla_{\mathbf{x}_{i^k}} f(\tilde{\mathbf{x}}^k)^\top (\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k) + \sum_{i \neq i^k} g_i(\mathbf{x}_i^k) + g_{i^k}(\mathbf{x}_{i^k}^{k+1}) \\ &\quad + (\nabla_{\mathbf{x}_{i^k}} f(\mathbf{x}^k) - \nabla_{\mathbf{x}_{i^k}} f(\tilde{\mathbf{x}}^k))^\top (\gamma (\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k)) + \frac{\gamma^2 L_f}{2} \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2 \\ &\stackrel{(c)}{\leq} f(\mathbf{x}^k) + \gamma \nabla_{\mathbf{x}_{i^k}} f(\tilde{\mathbf{x}}^k)^\top (\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k) \\ &\quad + (\nabla_{\mathbf{x}_{i^k}} f(\mathbf{x}^k) - \nabla_{\mathbf{x}_{i^k}} f(\tilde{\mathbf{x}}^k))^\top (\gamma (\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k)) + \frac{\gamma^2 L_f}{2} \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k\|_2^2 \\ &\quad + \sum_{i \neq i^k} g_i(\mathbf{x}_i^k) + \gamma g_{i^k}(\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k)) + g_{i^k}(\mathbf{x}_{i^k}^k) - \gamma g_{i^k}(\tilde{\mathbf{x}}_{i^k}^k) \\ &\stackrel{(d)}{\leq} F(\mathbf{x}^k) - \gamma \left(c_{\bar{f}} - \frac{\gamma L_f}{2} \right) \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k\|_2^2 \\ &\quad + L_f \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2 \|\gamma (\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k)\|_2 \\ &\stackrel{(e)}{\leq} F(\mathbf{x}^k) - \gamma \left(c_{\bar{f}} - \gamma L_f \right) \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \tilde{\mathbf{x}}_{i^k}^k\|_2^2 + \frac{L_f}{2} \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2 \end{aligned}$$

$$\stackrel{(f)}{=} F(\mathbf{x}^k) - \gamma \left(c_{\bar{f}} - \gamma L_f \right) \|\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{ik}^k\|_2^2 + \frac{L_f}{2} \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2, \quad (23)$$

where (a) follows from the updating rule of the algorithm; in (b) we used the Descent Lemma on f ; (c) comes from the convexity of g_i and C3; in (d) we used Proposition 2 and A3; (e) is due to the Young's inequality; and (f) is due to C3.

We bound $\|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2$ as follows:

$$\begin{aligned} \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2 &\stackrel{(a)}{\leq} \left(\sum_{l=k-\delta}^{k-1} \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2 \right)^2 \stackrel{(b)}{\leq} \delta \sum_{l=k-\delta}^{k-1} \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^2 \\ &= \delta \left(\sum_{l=k-\delta}^{k-1} (l - (k-1) + \delta) \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^2 - \sum_{l=k+1-\delta}^k (l - k + \delta) \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^2 \right) \\ &\quad + \delta^2 \gamma^2 \|\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{ik}^k\|_2^2, \end{aligned} \quad (24)$$

where (a) comes from (22); and (b) is due to the Jensen's inequality.

Using (24) in (23), the Lyapunov function (8), and rearranging the terms, the following holds: for all $k \geq 0$,

$$\begin{aligned} &\tilde{F}(\mathbf{x}^{k+1}, \dots, \mathbf{x}^{k+1-\delta}) \\ &\leq \tilde{F}(\mathbf{x}^k, \dots, \mathbf{x}^{k-\delta}) - \gamma \left(c_{\bar{f}} - \gamma \left(L_f + \frac{\delta^2 L_f}{2} \right) \right) \|\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{ik}^k\|_2^2; \end{aligned} \quad (25)$$

and

$$\begin{aligned} \tilde{F}(\mathbf{x}^{k+T}, \dots, \mathbf{x}^{k+T-\delta}) &\leq \tilde{F}(\mathbf{x}^{k+T-1}, \dots, \mathbf{x}^{k+T-1-\delta}) \\ &\quad - \gamma \left(c_{\bar{f}} - \gamma \left(L_f + \frac{\delta^2 L_f}{2} \right) \right) \|\hat{\mathbf{x}}_{ik+T-1}(\tilde{\mathbf{x}}^{k+T-1}) - \mathbf{x}_{ik+T-1}^{k+T-1}\|_2^2 \\ &\leq \tilde{F}(\mathbf{x}^k, \dots, \mathbf{x}^{k-\delta}) - \gamma \left(c_{\bar{f}} - \gamma \left(L_f + \frac{\delta^2 L_f}{2} \right) \right) \sum_{t=k}^{k+T-1} \|\hat{\mathbf{x}}_{it}(\tilde{\mathbf{x}}^t) - \mathbf{x}_{it}^t\|_2^2. \end{aligned} \quad (26)$$

Taking conditional expectation both sides we have that the following holds a.s.:

$$\begin{aligned} &\mathbb{E} \left(\tilde{F}(\underline{\mathbf{x}}^{k+T}, \dots, \underline{\mathbf{x}}^{k+T-\delta}) | \mathcal{F}^{k-1} \right) \leq \tilde{F}(\underline{\mathbf{x}}^k, \dots, \underline{\mathbf{x}}^{k-\delta}) \\ &\quad - \gamma \left(c_{\bar{f}} - \gamma \left(L_f + \frac{\delta^2 L_f}{2} \right) \right) \sum_{t=k}^{k+T-1} \mathbb{E} \left(\|\hat{\mathbf{x}}_{it}(\underline{\mathbf{x}}^t) - \mathbf{x}_{it}^t\|_2^2 | \mathcal{F}^{t-1} \right). \end{aligned} \quad (27)$$

Using (9), (27), A5, and the Martingale's theorem [29], we deduce that i) $\{\tilde{F}(\underline{\mathbf{x}}^k, \dots, \underline{\mathbf{x}}^{k-\delta})\}_{k \in \mathbb{N}_+}$, and thus $\{F(\underline{\mathbf{x}}^k, \dots, \underline{\mathbf{x}}^{k-\delta})\}_{k \in \mathbb{N}_+}$ converge a.s., ii) $\{\underline{\mathbf{x}}^k\}_{k \in \mathbb{N}_+}$ is bounded on \mathcal{X} a.s., and iii)

$$\lim_{k \rightarrow +\infty} \sum_{t=k}^{k+T-1} \mathbb{E} \left(\|\hat{\mathbf{x}}_{it}(\underline{\mathbf{x}}^t) - \mathbf{x}_{it}^t\|_2^2 | \mathcal{F}^{t-1} \right) = 0, \quad \text{a.s.} \quad (28)$$

From (28), it follows that there exists a set $\bar{\Omega} \subseteq \Omega$, with $\mathbb{P}(\bar{\Omega}) = 1$, such that for any $\omega \in \bar{\Omega}$,

$$\begin{aligned} & \sum_{t=k}^{k+T-1} \mathbb{E} \left(\|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t) - \mathbf{x}_{i^t}^t\|_2^2 | \mathcal{F}^{t-1} \right) \\ & \stackrel{(a)}{=} \sum_{t=k}^{k+T-1} \sum_{(i, \mathbf{d}) \in \mathcal{N} \times \mathcal{D}} p((i, \mathbf{d}) | \omega^{0:t-1}) \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^t) - \mathbf{x}_i^t\|_2 \\ & \stackrel{(b)}{\geq} p_{\min} \sum_{i=1}^N \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^{k+t_k(i)}) - \mathbf{x}_i^{k+t_k(i)}\|_2, \end{aligned} \quad (29)$$

where in (a) we used (16); and in (b) we used C2 and defined $t_k(i) \triangleq \min\{t \in [0; T] | p(i | \omega^{0:t+k-1}) \geq p_{\min}\}$. We also have:

$$\begin{aligned} \|\hat{\mathbf{x}}(\mathbf{x}^k) - \mathbf{x}^k\|_2 & \leq \sum_{i=1}^N \|\hat{\mathbf{x}}_i(\mathbf{x}^k) - \mathbf{x}_i^k\|_2 \\ & \stackrel{(a)}{\leq} \sum_{i=1}^N \left(\|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^{k+t_k(i)}) - \mathbf{x}_i^{k+t_k(i)}\|_2 + (1 + L_{\hat{\mathbf{x}}}) \|\tilde{\mathbf{x}}^{k+t_k(i)} - \mathbf{x}^k\|_2 \right) \\ & \stackrel{(b)}{\leq} \sum_{i=1}^N \left(\|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^{k+t_k(i)}) - \mathbf{x}_i^{k+t_k(i)}\|_2 + (1 + L_{\hat{\mathbf{x}}}) \left(\|\mathbf{x}^{k+t_k(i)} - \mathbf{x}^k\|_2 \right. \right. \\ & \quad \left. \left. + \sum_{l=k+t_k(i)-\delta}^{k+t_k(i)-1} \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2 \right) \right) \\ & \stackrel{(c)}{\leq} \sum_{i=1}^N \left(\|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^{k+t_k(i)}) - \mathbf{x}_i^{k+t_k(i)}\|_2 + 2\gamma(1 + L_{\hat{\mathbf{x}}}) \sum_{l=k-\delta}^{k+T-1} \|\hat{\mathbf{x}}_{i^l}(\tilde{\mathbf{x}}^l) - \mathbf{x}_{i^l}^l\|_2 \right), \end{aligned} \quad (30)$$

where in (a) we used Proposition 2; (b) comes from (22); and (c) from the updating rule of the algorithm. We deduce from (28), (29), and (30), that

$$\lim_{k \rightarrow +\infty} \|\hat{\mathbf{x}}(\mathbf{x}^k) - \mathbf{x}^k\|_2 = 0. \quad (31)$$

Since the sequence $\{\mathbf{x}^k\}_{k \in \mathbb{N}_+}$ is bounded, it has at least one limit point $\bar{\mathbf{x}}$ that belongs to \mathcal{X} . By the continuity of $\hat{\mathbf{x}}(\cdot)$ (see Proposition 2) and (31), it must be $\hat{\mathbf{x}}(\bar{\mathbf{x}}) = \bar{\mathbf{x}}$, and thus by Proposition 2 $\bar{\mathbf{x}}$ is a stationary solution of Problem (P). Since (31) holds for any $\omega \in \bar{\Omega}$, the previous results hold a.s..

Let us now define:

$$\hat{\mathbf{y}}_i(\mathbf{x}^k) = \operatorname{argmin}_{\mathbf{y}_i \in \mathcal{X}_i} \left\{ \nabla_{\mathbf{x}_i} f(\mathbf{x}^k)^\top (\mathbf{y}_i - \mathbf{x}_i^k) + g_i(\mathbf{y}_i) + \frac{1}{2} \|\mathbf{y}_i - \mathbf{x}_i^k\|_2^2 \right\}, \quad (32)$$

and note that $M_F(\mathbf{x}) = [\mathbf{x}_1^k - \hat{\mathbf{y}}_1(\mathbf{x}^k), \dots, \mathbf{x}_N^k - \hat{\mathbf{y}}_N(\mathbf{x}^k)]^\top$. It is easy to check that $\hat{\mathbf{y}}(\cdot)$ is $L_{\hat{\mathbf{y}}}$ -Lipschitz continuous on \mathcal{X} , with $L_{\hat{\mathbf{y}}} \triangleq L_f + 1$. Fix a realization $\omega \in \bar{\Omega}$. The optimality of $\hat{\mathbf{y}}_{i^k}(\mathbf{x}^k)$ along with the convexity of g_{i^k} , leads

$$\left(\nabla_{\mathbf{x}_{i^k}} f(\mathbf{x}^k) + \hat{\mathbf{y}}_{i^k}(\mathbf{x}^k) - \mathbf{x}_{i^k}^k \right)^\top \left(\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \hat{\mathbf{y}}_{i^k}(\mathbf{x}^k) \right) \quad (33)$$

$$+ g_{ik}(\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k)) - g_{ik}(\hat{\mathbf{y}}_{ik}(\mathbf{x}^k)) \geq 0. \quad (34)$$

Similarly, one can write for $\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k)$:

$$\nabla \tilde{f}_{ik}(\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k); \tilde{\mathbf{x}}^k)^\top (\hat{\mathbf{y}}_{ik}(\mathbf{x}^k) - \hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k)) + g_{ik}(\hat{\mathbf{y}}_{ik}(\mathbf{x}^k)) - g_{ik}(\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k)) \geq 0. \quad (35)$$

Summing (34) and (35), adding and subtracting $\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k)$, and using the gradient consistency B2, yield

$$\begin{aligned} & \left(\nabla \tilde{f}_{ik}(\mathbf{x}_{ik}^k; \mathbf{x}^k) - \nabla \tilde{f}_{ik}(\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k); \tilde{\mathbf{x}}^k) + \hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{ik}^k \right)^\top \\ & (\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \hat{\mathbf{y}}_{ik}(\mathbf{x}^k)) \geq \|\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \hat{\mathbf{y}}_{ik}(\mathbf{x}^k)\|_2^2. \end{aligned} \quad (36)$$

Summing and subtracting $\nabla \tilde{f}_{ik}(\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k); \mathbf{x}^k)$ and using the triangular inequality, the LHS of (36) can be upper bounded as

$$\begin{aligned} & \|\nabla \tilde{f}_{ik}(\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k); \mathbf{x}^k) - \nabla \tilde{f}_{ik}(\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k); \tilde{\mathbf{x}}^k)\|_2 \\ & + \|\nabla \tilde{f}_{ik}(\mathbf{x}_{ik}^k; \mathbf{x}^k) - \nabla \tilde{f}_{ik}(\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k); \mathbf{x}^k)\|_2 + \|\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{ik}^k\|_2 \\ & \geq \|\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \hat{\mathbf{y}}_{ik}(\mathbf{x}^k)\|_2. \end{aligned} \quad (37)$$

We can further upper-bound the left hand side invoking B3 and B4, and write:

$$\|\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \hat{\mathbf{y}}_{ik}(\mathbf{x}^k)\|_2 \leq (1 + L_E) \|\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{ik}^k\|_2 + L_B \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2. \quad (38)$$

Finally, squaring both sides, we get

$$\begin{aligned} \|\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \hat{\mathbf{y}}_{ik}(\mathbf{x}^k)\|_2^2 & \leq (1 + L_E)^2 \|\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{ik}^k\|_2^2 + L_B^2 \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2 \\ & + 2L_B(1 + L_E) \|\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{ik}^k\|_2 \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2. \end{aligned} \quad (39)$$

We bound next the term $\|\mathbf{x}_{ik}^k - \hat{\mathbf{y}}_{ik}(\mathbf{x}^k)\|_2^2$. We write

$$\begin{aligned} \|\mathbf{x}_{ik}^k - \hat{\mathbf{y}}_{ik}(\mathbf{x}^k)\|_2^2 & = \|\mathbf{x}_{ik}^k - \hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) + \hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \hat{\mathbf{y}}_{ik}(\mathbf{x}^k)\|_2^2 \\ & \leq 2 \left(\|\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{ik}^k\|_2^2 + \|\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \hat{\mathbf{y}}_{ik}(\mathbf{x}^k)\|_2^2 \right) \\ & \stackrel{(a)}{\leq} (2 + 2(1 + L_E)^2) \|\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{ik}^k\|_2^2 + 2L_B^2 \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2 \\ & \quad + 4L_B(1 + L_E) \|\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{ik}^k\|_2 \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2 \\ & \stackrel{(b)}{\leq} 2(1 + (1 + L_E)(1 + L_B + L_E)) \|\hat{\mathbf{x}}_{ik}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{ik}^k\|_2^2 \\ & \quad + 2L_B(1 + L_B + L_E) \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2, \end{aligned} \quad (40)$$

where (a) comes from (39); and (b) follows from the Young's inequality. Note that

$$\begin{aligned} \|\mathbf{x}^k - \tilde{\mathbf{x}}^k\|_2^2 & = \sum_{i=1}^N \|\mathbf{x}_i^k - \tilde{\mathbf{x}}_i^k\|_2^2 \stackrel{(a)}{\leq} \sum_{i=1}^N \left(\sum_{l \in \bar{\mathcal{K}}_i^k} \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2 \right)^2 \\ & \stackrel{(b)}{\leq} \sum_{i=1}^N \bar{M}_i^k \sum_{l \in \bar{\mathcal{K}}_i^k} \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^2 = \gamma^2 \sum_{i=1}^N \bar{M}_i^k \sum_{l \in \bar{\mathcal{K}}_i^k} \|\hat{\mathbf{x}}_{i^l}(\tilde{\mathbf{x}}^l) - \mathbf{x}_{i^l}^l\|_2^2, \end{aligned} \quad (41)$$

where (a) comes from (22); and in (b) we used the Jensen's inequality and defined $\bar{M}_i^k \triangleq |\bar{\mathcal{K}}_i^k|$. Combining (40) and (41), we get:

$$\begin{aligned} \|\mathbf{x}_{i^k}^k - \hat{\mathbf{y}}_{i^k}(\mathbf{x}^k)\|_2^2 &\leq 2(1 + (1 + L_E)(1 + L_B + L_E)) \|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2 \\ &+ 2\gamma^2 L_B(1 + L_B + L_E) \sum_{i=1}^N \bar{M}_i^k \sum_{l \in \bar{\mathcal{K}}_i^k} \|\hat{\mathbf{x}}_{i^l}(\tilde{\mathbf{x}}^l) - \mathbf{x}_{i^l}^l\|_2^2. \end{aligned} \quad (42)$$

We take now the conditional expectation of the term on the LHS of (42), and obtain

$$\begin{aligned} &\sum_{t=k}^{k+T} \mathbb{E} \left(\|\mathbf{x}_{i^t}^t - \hat{\mathbf{y}}_{i^t}(\mathbf{x}^t)\|_2^2 | \mathcal{F}^{t-1} \right) (\omega) \stackrel{(a)}{=} \sum_{t=k}^{k+T} \sum_{i=1}^N p(i | \omega^{0:t-1}) \|\mathbf{x}_i^t - \hat{\mathbf{y}}_i(\mathbf{x}^t)\|_2^2 \\ &\stackrel{(b)}{\geq} \sum_{i=1}^N p_{\min} \|\mathbf{x}_i^{k+t_k(i)} - \hat{\mathbf{y}}_i(\mathbf{x}^{k+t_k(i)})\|_2^2 \quad (43) \\ &\stackrel{(c)}{\geq} p_{\min} \sum_{i=1}^N \left(\|\mathbf{x}_i^k - \hat{\mathbf{y}}_i(\mathbf{x}^k)\|_2 - \|\mathbf{x}_i^{k+t_k(i)} - \hat{\mathbf{y}}_i(\mathbf{x}^{k+t_k(i)}) - \mathbf{x}_i^k + \hat{\mathbf{y}}_i(\mathbf{x}^k)\|_2 \right)^2 \\ &\geq p_{\min} \sum_{i=1}^N \left(\|\mathbf{x}_i^k - \hat{\mathbf{y}}_i(\mathbf{x}^k)\|_2^2 \right. \\ &\quad \left. - 2\|\mathbf{x}_i^k - \hat{\mathbf{y}}_i(\mathbf{x}^k)\|_2 \|\mathbf{x}_i^{k+t_k(i)} - \hat{\mathbf{y}}_i(\mathbf{x}^{k+t_k(i)}) - \mathbf{x}_i^k + \hat{\mathbf{y}}_i(\mathbf{x}^k)\|_2 \right), \end{aligned}$$

where in (a) we used (16); (b) follows from C2; and in (c) we used the reverse triangle inequality. By (43) and (42), we obtain:

$$\begin{aligned} &p_{\min} \sum_{i=1}^N \|\mathbf{x}_i^k - \hat{\mathbf{y}}_i(\mathbf{x}^k)\|_2^2 = p_{\min} \|M_F(\mathbf{x}^k)\|_2^2 \\ &\leq \sum_{t=k}^{k+T} \left(2(1 + (1 + L_E)(1 + L_B + L_E)) \mathbb{E} \left(\|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t) - \mathbf{x}_{i^t}^t\|_2^2 | \mathcal{F}^{t-1} \right) (\omega) \right. \\ &\quad \left. + 2\gamma^2 L_B(1 + L_B + L_E) \sum_{i=1}^N \bar{M}_i^t \sum_{l \in \bar{\mathcal{K}}_i^t} \|\hat{\mathbf{x}}_{i^l}(\tilde{\mathbf{x}}^l) - \mathbf{x}_{i^l}^l\|_2^2 \right) \\ &\quad + 2p_{\min} \sum_{i=1}^N \|\mathbf{x}_i^k - \hat{\mathbf{y}}_i(\mathbf{x}^k)\|_2 \|\mathbf{x}_i^{k+t_k(i)} - \hat{\mathbf{y}}_i(\mathbf{x}^{k+t_k(i)}) - \mathbf{x}_i^k + \hat{\mathbf{y}}_i(\mathbf{x}^k)\|_2 \\ &\stackrel{(a)}{\leq} 2(1 + (1 + L_E)(1 + L_B + L_E)) \sum_{t=k}^{k+T} \mathbb{E} \left(\|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t) - \mathbf{x}_{i^t}^t\|_2^2 | \mathcal{F}^{t-1} \right) (\omega) \\ &\quad + 2T\gamma^2 L_B(1 + L_B + L_E) \sum_{i=1}^N \bar{M}_i^k \sum_{l \in \bar{\mathcal{K}}_i^k} \|\hat{\mathbf{x}}_{i^l}(\tilde{\mathbf{x}}^l) - \mathbf{x}_{i^l}^l\|_2^2 + p_{\min} \alpha \|M_F(\mathbf{x}^k)\|_2^2 \\ &\quad + p_{\min} \alpha^{-1} \sum_{i=1}^N \|\mathbf{x}_i^{k+t_k(i)} - \hat{\mathbf{y}}_i(\mathbf{x}^{k+t_k(i)}) - \mathbf{x}_i^k + \hat{\mathbf{y}}_i(\mathbf{x}^k)\|_2^2 \end{aligned}$$

$$\begin{aligned}
&\stackrel{(b)}{\leq} 2(1 + (1 + L_E)(1 + L_B + L_E)) \sum_{t=k}^{k+T} \mathbb{E} \left(\|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t) - \mathbf{x}_{i^t}^t\|_2^2 | \mathcal{F}^{t-1} \right) (\omega) \\
&\quad + 2T\gamma^2 L_B(1 + L_B + L_E) \sum_{i=1}^N M_i^k \sum_{l \in \mathcal{K}_i^k} \|\hat{\mathbf{x}}_{i^l}(\tilde{\mathbf{x}}^l) - \mathbf{x}_{i^l}^l\|_2^2 + p_{\min} \alpha \|M_F(\mathbf{x}^k)\|_2^2 \\
&\quad + 2p_{\min} \alpha^{-1} \sum_{i=1}^N \left(\|\mathbf{x}_i^{k+t_k(i)} - \mathbf{x}_i^k\|_2^2 + \|\hat{\mathbf{y}}_i(\mathbf{x}^{k+t_k(i)}) - \hat{\mathbf{y}}_i(\mathbf{x}^k)\|_2^2 \right) \\
&\stackrel{(c)}{\leq} 2(1 + (1 + L_E)(1 + L_B + L_E)) \sum_{t=k}^{k+T} \mathbb{E} \left(\|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t) - \mathbf{x}_{i^t}^t\|_2^2 | \mathcal{F}^{t-1} \right) (\omega) \\
&\quad + 2T\gamma^2 L_B(1 + L_B + L_E) \sum_{i=1}^N M_i^k \sum_{l \in \mathcal{K}_i^k} \|\hat{\mathbf{x}}_{i^l}(\tilde{\mathbf{x}}^l) - \mathbf{x}_{i^l}^l\|_2^2 + p_{\min} \alpha \|M_F(\mathbf{x}^k)\|_2^2 \\
&\quad + 2\gamma^2 p_{\min} \alpha^{-1} (1 + L_{\hat{\mathbf{y}}}^2) \sum_{i=1}^N \sum_{l=k}^{k+t_k(i)-1} \|\hat{\mathbf{x}}_{i^l}(\tilde{\mathbf{x}}^l) - \mathbf{x}_{i^l}^l\|_2^2 \\
&\leq 2(1 + (1 + L_E)(1 + L_B + L_E)) \sum_{t=k}^{k+T} \mathbb{E} \left(\|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t) - \mathbf{x}_{i^t}^t\|_2^2 | \mathcal{F}^{t-1} \right) (\omega) \\
&\quad + 2T\gamma^2 L_B(1 + L_B + L_E) \sum_{i=1}^N M_i^k \sum_{l \in \mathcal{K}_i^k} \|\hat{\mathbf{x}}_{i^l}(\tilde{\mathbf{x}}^l) - \mathbf{x}_{i^l}^l\|_2^2 + p_{\min} \alpha \|M_F(\mathbf{x}^k)\|_2^2 \\
&\quad + 2\gamma^2 p_{\min} \alpha^{-1} (1 + L_{\hat{\mathbf{y}}}^2) N \sum_{l=k}^{k+T-1} \|\hat{\mathbf{x}}_{i^l}(\tilde{\mathbf{x}}^l) - \mathbf{x}_{i^l}^l\|_2^2, \tag{44}
\end{aligned}$$

where in (a) we used the Young's inequality and the definition of M_i^k (cf. Section 4); in (b) we used the triangle and Jensen's inequalities; and (c) comes from the updating rule of the algorithm. Rearranging the terms and taking expectation of both sides, we get:

$$\begin{aligned}
&\mathbb{E} \left(\|M_F(\mathbf{x}^k)\|_2^2 \right) \\
&\leq \frac{2 \left(1 + (1 + L_E)(1 + L_B + L_E) + \gamma^2 N p_{\min} \alpha^{-1} (1 + L_{\hat{\mathbf{y}}}^2) \right)}{p_{\min} - \alpha p_{\min}} \\
&\quad \sum_{t=k}^{k+T} \mathbb{E} \left(\|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t) - \mathbf{x}_{i^t}^t\|_2^2 \right) \\
&\quad + \frac{2T\gamma^2 L_B(1 + L_B + L_E)}{p_{\min} - \alpha p_{\min}} \mathbb{E} \left(\sum_{i=1}^N M_i^k \sum_{t \in \mathcal{K}_i^k} \|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t) - \mathbf{x}_{i^t}^t\|_2^2 \right). \tag{45}
\end{aligned}$$

Invoking (9) and (25), we can write

$$\|\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k\|_2^2$$

$$\leq \frac{1}{\gamma \left(c_{\tilde{f}} - \gamma \left(L_f + \frac{\delta^2 L_f}{2} \right) \right)} \left(\tilde{F}(\mathbf{x}^k, \dots, \mathbf{x}^{k-\delta}) - \tilde{F}(\mathbf{x}^{k+1}, \dots, \mathbf{x}^{k+1-\delta}) \right). \quad (46)$$

Using this bound in (45), we get

$$\begin{aligned} \mathbb{E} (\|M_F(\mathbf{x}^k)\|_2^2) &\leq C_1 \sum_{t=k}^{k+T} \mathbb{E} \left(\tilde{F}(\mathbf{x}^t, \dots, \mathbf{x}^{t-\delta}) - \tilde{F}(\mathbf{x}^{t+1}, \dots, \mathbf{x}^{t+1-\delta}) \right) \\ &+ \gamma^2 C_2 \mathbb{E} \left(\sum_{i=1}^N M_i^k \sum_{t \in \mathcal{K}_i^k} \left(\tilde{F}(\mathbf{x}^t, \dots, \mathbf{x}^{t-\delta}) - \tilde{F}(\mathbf{x}^{t+1}, \dots, \mathbf{x}^{t+1-\delta}) \right) \right). \end{aligned} \quad (47)$$

Finally,

$$\begin{aligned} K_\epsilon \epsilon &\leq \sum_{k=0}^{K_\epsilon} \mathbb{E} (\|M_F(\mathbf{x}^k)\|_2^2) \\ &\leq C_1 \sum_{k=0}^{K_\epsilon} \mathbb{E} \left(\tilde{F}(\mathbf{x}^k, \dots, \mathbf{x}^{k-\delta}) - \tilde{F}(\mathbf{x}^{k+T+1}, \dots, \mathbf{x}^{k+T+1-\delta}) \right) \\ &+ \gamma^2 C_2 \mathbb{E} \left(\sum_{i=1}^N M_i^k \sum_{t \in \mathcal{K}_i^k} \left(\tilde{F}(\mathbf{x}^t, \dots, \mathbf{x}^{t-\delta}) - \tilde{F}(\mathbf{x}^{t+1}, \dots, \mathbf{x}^{t+1-\delta}) \right) \right) \\ &\leq C_1 (T+1) (F(\mathbf{x}^0) - F^*) \\ &+ C_2 \gamma^2 \sum_{k=0}^{K_\epsilon} \mathbb{E} \left(\sum_{i=1}^N M_i^k \sum_{t \in \mathcal{K}_i^k} \left(\tilde{F}(\mathbf{x}^t, \dots, \mathbf{x}^{t-\delta}) - \tilde{F}(\mathbf{x}^{t+1}, \dots, \mathbf{x}^{t+1-\delta}) \right) \right). \end{aligned} \quad (48)$$

This completes the proof.

7.3 Proof of Theorem 2

In this section, the best-response map $\hat{\mathbf{x}}(\cdot)$ is the one defined in (15).

Statement (ii) of the theorem follow readily from the feasibility of $\mathbf{x}^0 \in \mathcal{K}$ and the fact that $\mathbf{x}_{i^k}^{k+1} = \mathbf{x}_{i^k}^k + \gamma(\hat{\mathbf{x}}_{i^k}(\tilde{\mathbf{x}}^k) - \mathbf{x}_{i^k}^k)$ is a convex combinations of points in $\mathcal{K}_{i^k}(\mathbf{x}_{i^k}^k)$.

To prove statement (ii), let us fix a realization $\omega \in \Omega$. Following the steps from (23) to (27), one can prove that the following holds a.s.:

$$\begin{aligned} \mathbb{E} \left(\tilde{F}(\mathbf{x}^{k+T}, \dots, \mathbf{x}^{k+T-\delta}) | \mathcal{F}^{k-1} \right) &\leq \tilde{F}(\mathbf{x}^k, \dots, \mathbf{x}^{k-\delta}) \\ &- \gamma \left(c_{\tilde{f}} - \gamma \left(L_f + \frac{\delta^2 L_f}{2} \right) \right) \sum_{t=k}^{k+T-1} \mathbb{E} \left(\|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t) - \mathbf{x}_{i^t}^t\|_2^2 | \mathcal{F}^{t-1} \right). \end{aligned} \quad (49)$$

Using (9), (49) and A5', we deduce that i) $\{F(\underline{\mathbf{x}}^k, \dots, \underline{\mathbf{x}}^{k-\delta})\}_{k \in \mathbb{N}_+}$ converges a.s., and ii)

$$\lim_{k \rightarrow +\infty} \sum_{t=k}^{k+T-1} \mathbb{E} \left(\|\hat{\mathbf{x}}_{i^t}(\tilde{\mathbf{x}}^t) - \mathbf{x}_{i^t}^t\|_2 | \mathcal{F}^{t-1} \right) = 0 \quad \text{a.s.} \quad (50)$$

It follows from (50) and C2 that

$$\lim_{k \rightarrow +\infty} \sum_{i=1}^N \|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^{k+t_k(i)}) - \mathbf{x}_i^{k+t_k(i)}\|_2 = 0 \quad \text{a.s.} \quad (51)$$

Therefore, there exists a set $\bar{\Omega} \subseteq \Omega$, with $\mathbb{P}(\bar{\Omega}) = 1$, such that, for any $\omega \in \bar{\Omega}$,

$$\begin{aligned} \|\hat{\mathbf{x}}(\mathbf{x}^k) - \mathbf{x}^k\|_2 &\leq \sum_{i=1}^N \|\hat{\mathbf{x}}_i(\mathbf{x}^k) - \mathbf{x}_i^k\|_2 \stackrel{(a)}{\leq} \sum_{i=1}^N \left(\|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^{k+t_k(i)}) - \mathbf{x}_i^{k+t_k(i)}\|_2 \right. \\ &\quad \left. + \|\tilde{\mathbf{x}}^{k+t_k(i)} - \mathbf{x}^k\|_2^{1/2} \left(\|\tilde{\mathbf{x}}^{k+t_k(i)} - \mathbf{x}^k\|_2^{1/2} + \tilde{L}_{\hat{\mathbf{x}}} \right) \right) \\ &\stackrel{(b)}{\leq} \sum_{i=1}^N \left(\|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^{k+t_k(i)}) - \mathbf{x}_i^{k+t_k(i)}\|_2 + \left(\|\mathbf{x}^{k+t_k(i)} - \mathbf{x}^k\|_2^{1/2} \right. \right. \\ &\quad \left. \left. + \sum_{l=k+t_k(i)-\delta}^{k+t_k(i)-1} \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^{1/2} \right) \right. \\ &\quad \left. \left(\|\mathbf{x}^{k+t_k(i)} - \mathbf{x}^k\|_2^{1/2} + \sum_{l=k+t_k(i)-\delta}^{k+t_k(i)-1} \|\mathbf{x}^{l+1} - \mathbf{x}^l\|_2^{1/2} + \tilde{L}_{\hat{\mathbf{x}}} \right) \right) \\ &\stackrel{(c)}{\leq} \sum_{i=1}^N \left(\|\hat{\mathbf{x}}_i(\tilde{\mathbf{x}}^{k+t_k(i)}) - \mathbf{x}_i^{k+t_k(i)}\|_2 \right. \\ &\quad \left. + 2\sqrt{\gamma} \sum_{l=k-\delta}^{k+T-1} \|\hat{\mathbf{x}}_{i^l}(\tilde{\mathbf{x}}^l) - \mathbf{x}_{i^l}^l\|_2^{1/2} \left(2\sqrt{\gamma} \sum_{l=k-\delta}^{k+T-1} \|\hat{\mathbf{x}}_{i^l}(\tilde{\mathbf{x}}^l) - \mathbf{x}_{i^l}^l\|_2^{1/2} + \tilde{L}_{\hat{\mathbf{x}}} \right) \right), \end{aligned} \quad (52)$$

where in (a) we used Proposition 3; (b) comes from (22); and in (c) we used the updating rule of the algorithm. Using (50), (51) and (52), we conclude that

$$\lim_{k \rightarrow +\infty} \|\hat{\mathbf{x}}(\mathbf{x}^k) - \mathbf{x}^k\|_2 = 0. \quad (53)$$

A straightforward generalization of [30, Theorem 11] together with (53) proves that every limit point of $\{\mathbf{x}^k\}_{k \in \mathbb{N}_+}$ is a stationary solution of Problem (P'). Since (53) holds for any given realization $\omega \in \bar{\Omega}$, the above results hold a.s..

Iteration complexity can be proved following the steps (32)-(48) and using the convexification of the nonconvex constraint sets where needed; details are omitted.