HAL
archives-ouvertes.fr

# A Distributed Flexible Delay-tolerant Proximal Gradient Algorithm

Konstantin Mishchenko, Franck Iutzeler, Jérôme Malick

▶ **To cite this version:**

Konstantin Mishchenko, Franck Iutzeler, Jérôme Malick. A Distributed Flexible Delay-tolerant Proximal Gradient Algorithm. SIAM Journal on Optimization, Society for Industrial and Applied Mathematics, 2020, 30 (1), pp.933-959. 10.1137/18M1194699 . hal-01821683v2

**HAL Id: hal-01821683**
**https://hal.archives-ouvertes.fr/hal-01821683v2**

Submitted on 17 Nov 2020

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# A DISTRIBUTED FLEXIBLE DELAY-TOLERANT PROXIMAL GRADIENT ALGORITHM *

KONSTANTIN MISHCHENKO†, FRANCK IUTZELER‡, AND JÉRÔME MALICK§

**Abstract.** We develop and analyze an asynchronous algorithm for distributed convex optimization when the objective can be written as a sum of smooth functions, local to each worker, and a non-smooth function. Unlike many existing methods, our distributed algorithm is adjustable to various levels of communication cost, delays, machines' computational power, and functions' smoothness. A unique feature is that the stepsizes do not depend on communication delays nor number of machines, which is highly desirable for scalability. We prove that the algorithm converges linearly in the strongly convex case, and provide guarantees of convergence for the non-strongly convex case. The obtained rates are the same as the vanilla proximal gradient algorithm over some introduced epoch sequence that subsumes the delays of the system. We provide numerical results on large-scale machine learning problems to demonstrate the merits of the proposed method.

**1. Introduction.** A broad range of problems arising in machine learning and signal processing can be formulated as minimizing the sum of $M$ smooth functions $(f_i)$ and a non-smooth proximable function $g$

$$\min_{x\in\mathbb{R}^n} \quad \frac{1}{M}\sum_{i=1}^{M} f_i(x) \;+\; g(x). \tag{1}$$

For instance, $f_i$ may represent a local loss and $g$ a non-smooth regularizer that imposes some structure on optimal solutions. Typical examples include the $\ell_1$-regularized regression [27] in which $g$ is taken as the $\ell_1$-norm [2].

**1.1. Distributed setting.** In this paper[1], we consider the optimization problem (1) in a distributed setting with $M$ worker machines, where worker $i$ has private information on the smooth function $f_i$. More precisely, we assume that each worker $i$ can compute:

- the gradient of its local function $\nabla f_i$;
- the proximity operator of the common non-smooth function $\mathrm{prox}_g$.

We further consider a master slave framework where the workers exchange information with a master machine which has no global information about the problem but only coordinates the computation of agents in order to minimize (1). Having asynchronous exchanges between the workers and the master is of paramount importance for practical efficiency; indeed, asynchronous algorithms can perform more iterations per second for nearly the same improvement as their synchronous counterparts even with large delays (see e.g. the asynchronous parallel fixed point algorithm of [11]). In the considered setup, as soon as the master receives an update from a worker, it updates the master variable, and sends it to this worker which then carries on its computation.

This distributed setting covers a variety of scenarios when computation are scattered over distributed devices (computer clusters, mobiles), each having a local part of

*Submitted to the editors December 12, 2019.

†KAUST (konstantin.mishchenko@kaust.edu.sa, http://konstmish.github.io/).

‡Laboratoire Jean Kuntzmann, Univ. Grenoble Alpes (franck.iutzeler@univ-grenoble-alpes.fr).

§CNRS, Laboratoire Jean Kuntzmann (jerome.malick@univ-grenoble-alpes.fr).

[1]Our preliminary work in a machine learning context [18] presents briefly the asynchronous framework and a theoretical study in the strongly convex case. We extend this work on several aspects with in particular a deeper analysis of the asynchronous setting, the use of local stepsizes, and the study of the general convex case.

the data (the locality arising from the prohibitive size of the data, or its privacy [24]), as in federated learning [14]. In the large-scale machine learning applications for instance, data points can be split across the $M$ workers, so that each worker $i$ has a local loss function $f_i$ with properties which may be different due to data distribution unevenness.

We focus on the setup where (i) the workers' functions differ in their values and the computational complexity of their local oracles (e.g. due to non-i.i.d. unbalanced local datasets in a learning scenario); (ii) the communications between workers and the master are time-consuming (e.g. due to scarce availability or slow communications). This implies that we need to pay a special attention to the delays of workers' updates.

**1.2. Contributions and outline.** In this distributed setting, we provide an asynchronous algorithm and the associated analysis that adapts to local functions' parameters and can handle any kind of delays. The algorithm is based on totally asynchronous proximal gradient iterations with different stepsizes, which makes it adaptive to the individual functions' properties. In order to subsume delays, we develop a new epoch-based mathematical analysis, encompassing computation times and communication delays, to refocus the theory on algorithmics. We show convergence in the general convex case and linear convergence in the strongly convex case. More precisely, we show that the proposed method verifies a decrease depending only on the problem parameters (strong convexity and smoothness constants) over meta-iterations, called *epochs*, that subsume the delays between the different workers. This approach thus decouples the convergence between the problem parameters and the delays brought by asynchrony. The algorithm thus handles the diversity of the previously-discussed applications.

The paper is organized as follows. In Section 2, we give a description of the algorithm, split into the communication and the optimization scheme, as well as a comparison with the most related literature. In Section 3, we develop our epoch-based analysis of convergence, separating the general and the strongly convex case. In Section 4, we provide illustrative computational experiments on standard $\ell_1$-regularized problems showing the efficiency of the algorithm and its resilience to delays.

**1.3. Related work.** Most existing methods for solving problem (1) in the considered context are based either on parallel stochastic algorithms or on distributed extensions of standard algorithms.

Stochastic algorithms have received a lot of attention, regarding convergence rates, acceleration, parallelization, generalization to non-smooth or sparse gradient cases; see e.g. [23, 13, 8]. Parrallel versions of stochastic algorithms have also been proposed where subparts of the data are stored in different machines (Hogwild! [22], Distributed SDCA [26], Distributed SVRG [15], ProxASAGA [20]). Despite their theoretical properties and practical success in the context of multicore computers, these algorithms are not well-suited for our distributed setting where we focus not only on the number of data accesses, but also on the number of communication steps (see e.g. [16]). For example, ASAGA [20] makes computations in parallel but does not fit our framework, as it assumes uniform sampling with shared memory between computing parties. Thus, a naive extension of such parallel stochastic methods would be inefficient in practice due to large overheads in communications.

There also exists a rich literature on distributed optimization algorithms with no shared memory. We mention e.g. ARock [21], Asynchronous ADMM [30], COCOA [17], Delayed Proximal Gradient algorithms [1, 29], or dSAGA [6]. These methods often have restrictive assumptions about synchrony of communications, or boundedness

of the delays between fastest and slowest machines. For instance, the asynchronous distributed ADMM of [30] allows asynchronous updates only until a maximal delay, after which every worker has to wait for the slowest one.

Usually, the bounds on delays also impact the stepsizes in algorithms and the convergence rates, as in [21, 1]. The only other work establishing convergence with unbounded delays is [25, 12] for asynchronous coordinate descent methods (but with decreasing stepsizes). In contrast with all existing literature, we propose a totally asynchronous algorithm that does not require any knowledge about the delays: in practice, delays impact the observed convergence but i) the choice of the stepsize is delay-independent; and ii) our convergence analysis relies on the notion of *epochs* subsuming the delays produced by asynchrony (over these epochs the rate only depends on the problem parameters).

Let us finally point out the main improvements over the companion conference paper [18] which presents briefly the asynchronous framework and a theoretical study in the strongly convex case. In the present paper, we provide a pedagogical study of the mechanisms at play (in Section 2) and a refined analysis covering the non-strongly convex case (in Section 3 and the appendices). For a better understanding, we add several illustrative figures explaining the algorithms, the proof techniques, and toy examples. The experiments provided here are complementary to those presented in [18]; in particular, we illustrate the behavior of the algorithm for non-strongly convex objectives.

## 2. `DAve-RPG`: Distributed Averaging of Repeated Proximal Gradient.

In this section, we present the proposed `DAve-RPG` algorithm, where `DAve` stands for the global communication scheme based on distributed averaging of iterates, and `RPG` stands for the local optimization scheme, based on repeated proximal-gradient steps. We start by presenting the generic master slave setting and associated notations.

### 2.1. Asynchronous Master Slave Framework.

We consider the master slave model: in order to reach the global objective (1), the workers exchange information with a master machine. In view of practical efficiency (see e.g. the recent [11]), these exchanges are asynchronous: at each moment when the master receives an update from a worker, it revises its master variable and sends it back to the sender.

In compliance with this asynchronous setting, we call iteration/time $k$ (denoted by a superscript $k$), the moment of the $k$-th exchange between a worker and the master, or, equivalently, the $k$-th time the master has updated its master variable. For a worker $i$ and a time $k$, we denote by $d_i^k$ the delay for $i$ at time $k$, i.e. the number of master updates since worker $i$ conversed with the master. More precisely, at time $k$, the updating worker $i = i(k)$ suffers no delay (in terms of update in the master variable), i.e. $d_i^k = 0$, while the delays of the other workers are incremented ($d_j^k = d_j^{k-1} + 1$ for all $j \neq i(k)$). In addition, we denote by $D_i^k$ the relative delay from the penultimate update, mathematically defined as $D_j^k = d_j^k + d_j^{k-d_j^k-1} + 1$ for worker $j$ and time $k$. This asynchronous distributed setup and the corresponding definitions are illustrated in Fig. 1.

A key point in this work is that we do *not* assume that the delays are uniformly bounded. We prove instead the convergence of our algorithm and the associated rates using a companion sequence that subsumes the delays. This places this work in the *totally asynchronous* setting according to Bertsekas and Tsitsiklis' classification [4, Chap. 6.1]. Nevertheless, for clarity, we will also provide convergence rates in the partially asynchronous setting with uniform delay boundedness ($d_i^k \leq d$ for all $i, k$) or

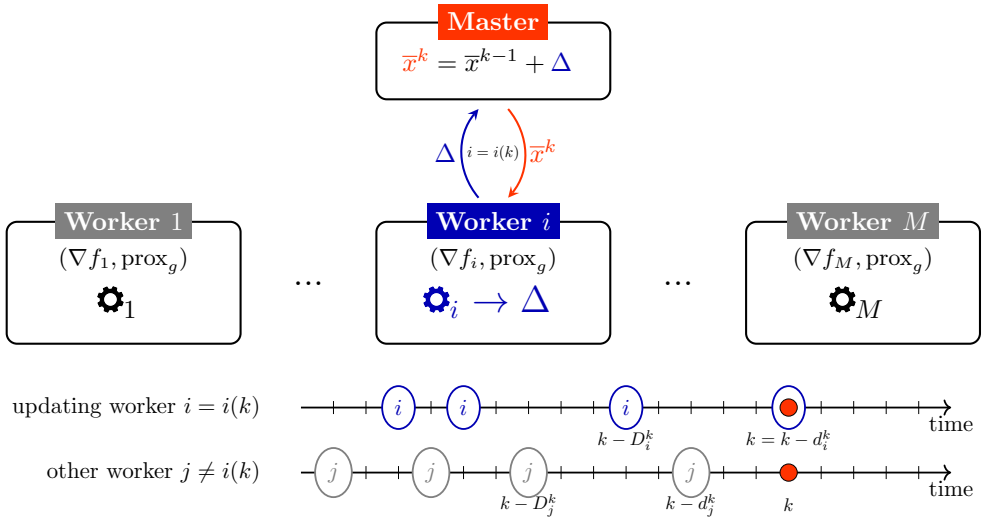



Fig. 1: Asynchronous distributed setting and delays notations at iteration $k$.

average delay boundedness ($1/M \sum_{i=1}^M d_i^k \leq (M-1)/2 + d$ for all $k$) in Section 3.4.

**2.2. `DAve` Communication scheme.** Our communication scheme is based on maintaining at the master the weighted average of the most updated parameters of the workers. At time $k$, worker $i = i(k)$ finishes the computation of a new *local parameter* $x_i^k$ and the corresponding *adjustment* $\Delta$ corresponding to the weighted difference between its new and former local parameter. As soon as the computation is finished, this adjustment is sent to the master node which, in turn, adds it to its *master parameter* $\overline{x}^k$. The master then immediately sends back this parameter to worker $i$, which can begin a new computation step. During the updates, the master variable is "locked" (see e.g. the description of [21]), guaranteeing read/write consistency.

Mathematically, at each time $k$, one has

$$\overline{x}^k = \overline{x}^{k-1} + \Delta \text{ with } \Delta = \pi_i (x_i^k - x_i^{k-D_i^k}) \text{ for } i = i(k) \tag{2}$$

$$\text{thus,} \quad \overline{x}^k = \sum_{i=1}^M \pi_i x_i^{k-d_i^k} = \sum_{i=1}^M \pi_i \blacksquare_i (\overline{x}^{k-D_i^k}) \tag{3}$$

where $\blacksquare_i$ represents the computation of worker $i$ (see Figure 1) and the $(\pi_i)_{i=1,..,M}$ are the weights of the workers contributions. These weights are positive real numbers such that $\sum_{i=1}^M \pi_i = 1$ and are kept fixed over time; their values are derived from the optimality conditions of (1) and the workers computation. In this paper, the agents will perform (proximal) gradient steps ($\blacksquare_i$ = proximal gradient step on $f_i + g$) which leads to the weights given by (5).

We see in Eq. (3) that $\overline{x}^k$ depends on local parameters $(x_i^{k-d_i^k})_i$, which themselves were computed using (once more delayed) global parameters $(\overline{x}^{k-D_i^k})_i$. A unique feature[2] of our distributed algorithm is that at each time, the master variable is as a

[2]We note that this idea of averaging iterates has also been used in the different context: for

weighted average of the agents' last contributions. This means that the contribution of each worker in the master variable stays fixed over time even if one worker updates much more frequently than the others. Though this simple idea might be counterproductive in other contexts, it allows here the algorithm to cope with heterogeneity in the computing system such as data distribution and agents delays. Roughly speaking, in standard approaches, if an agent has very outdated information, the output of its computation can lead to a counter-productive change, generating instability in the algorithm; keeping a fixed average of the contributions offers a counterbalance to such drastic updates. This phenomenon is illustrated in the case where the agents computations are gradients steps in Section 2.4, notably through Figures 2 and 3.

**2.3. Optimization scheme: Repeated Proximal Gradient `RPG`.** As the problem features a smooth and a non-smooth part, it is natural that the workers use proximal gradient steps. Furthermore, we allow the repetition of local proximal gradient steps before exchanging with the master, for higher flexibility in the computing time between two exchanges. We present our `RPG` scheme in 3 stages, explaining the three letters of the name. For more readability, we consider a generic worker $i$ and time $k$ when $i = i(k)$ is the exchanging worker (as represented in Figure 1).

♦ **G.** If $g \equiv 0$, then each worker may perform a simple gradient step on the last master parameter received $\overline{x}^{k-D_i^k}$:

$$x_i^k \leftarrow \overline{x}^{k-D_i^k} - \gamma_i \nabla f_i(\overline{x}^{k-D_i^k}), \qquad \Delta \leftarrow \pi_i \left( x_i^k - x_i^{k-D_i^k} \right) \tag{4}$$

where $\gamma_i$ is the *local stepsize* at worker $i$ (related only to function $f_i$) and

$$\pi_i := \frac{\frac{1}{\gamma_i}}{\sum_{j=1}^M \frac{1}{\gamma_j}} \tag{5}$$

is the *proportion* of worker $i$'s contribution, necessary to converge to the correct point.

♦ **PG.** For a general non-smooth convex function $g$, we consider the proximity operator, defined for any $\gamma > 0$ by

$$\mathrm{prox}_{\gamma g}(x) = \arg\min_z \left\{ g(z) + \frac{1}{2\gamma} \|z - x\|^2 \right\}.$$

One can extend (4) in the same way iteration $\mathrm{prox}_{\gamma g}(x - \gamma \nabla f(x))$ generalizes a gradient step. However, contrary to direct intuition, the proximity operator has to be computed first, leading to a temporary variable $z$, on which is taken the gradient step before exchanging:

$$z \leftarrow \mathrm{prox}_{\gamma g}(\overline{x}^{k-D_i^k}), \quad x_i^k \leftarrow z - \gamma_i \nabla f_i(z), \quad \Delta \leftarrow \pi_i \left( x_i^k - x_i^{k-D_i^k} \right) \tag{6}$$

with $\gamma$ being the *master stepsize* appearing in *all* proximity operators:

$$\gamma := \frac{M}{\sum_{i=1}^M \frac{1}{\gamma_i}} \tag{7}$$

equal to the harmonic average of the local stepsizes. Note that our algorithm allows for different local stepsizes, which simplifies parameters tuning as it can be done locally. Then, the proximity operators have to be taken with a separate master stepsize.

variance reduction in incremental methods [9, 19].

♦ **RPG.** Once all computations of iteration (6) are done, the slave could send the adjustment $\Delta$ to the master and get $\overline{x}^k$ in response. However, the difference between the latest $\overline{x}^k$ and $\overline{x}^{k-D_i^k}$ may be small, so the worker would only gain little information from a new exchange. Thus, instead of communicating right away, we suggest to perform additional proximal gradient updates by taking as the starting point $\overline{x}^{k-D_i^k} + \Delta$. The motivation behind this repetition is to lower the burden of communications and to focus on computing good updates. We will prove later that there is no restriction on the number of repetitions (called $p$ in the algorithm), as any value can be chosen and it can vary freely both across machines and over time.

DAve-RPG

**Master:**

Initialize $\overline{x} = \overline{x}^0$, $k = 0$
**while** *not converge* **do**
  **when** a worker finishes:
  Receive adjustment $\Delta$ from it
  $\overline{x} \leftarrow \overline{x} + \Delta$
  Send $\overline{x}$ to the agent in return
  $k \leftarrow k + 1$
**end**
Interrupt all slaves
**Output** $x = \mathrm{prox}_{\gamma g}(\overline{x})$

**Slave $i$:**

Initialize $x = x_i = \overline{x}$,
**while** *not interrupted by master* **do**
  Receive the most recent $\overline{x}$
  Select a number of repetitions $p$
  $\Delta \leftarrow 0$
  **for** $q = 1$ **to** $p$ **do**
    $z \leftarrow \mathrm{prox}_{\gamma g}(\overline{x} + \Delta)$
    $x^+ \leftarrow z - \gamma_i \nabla f_i(z)$
    $\Delta \leftarrow \Delta + \pi_i (x^+ - x)$
    $x \leftarrow x^+$
  **end**
  Send adjustment $\Delta$ to master
**end**

**2.4. Comparison between DAve-(R)PG and PIAG.** Our algorithm DAve-RPG performs a distributed minimization of the composite problem (1) by aggregating the agents contributions. It is closely related to the proximal incremental aggregated gradient (PIAG) method [1, 28]. We can compare the update of PIAG with the one of $x^k = \mathrm{prox}_{\gamma g}(\overline{x}^k)$ for DAve-PG[3] (with one repetition, $p = 1$).

| DAve-PG | PIAG |
|---|---|
| $x^k = \mathrm{prox}_{\gamma g}\left(\sum_{i=1}^M \pi_i x^{k-D_i^k} - \sum_{i=1}^M \pi_i \gamma_i \nabla f_i(x^{k-D_i^k})\right)$ $= \mathrm{prox}_{\gamma g}\left(\sum_{i=1}^M \pi_i x^{k-D_i^k} - \gamma \frac{1}{M} \sum_{i=1}^M \nabla f_i(x^{k-D_i^k})\right)$ | $x^k = \mathrm{prox}_{\gamma g}\left(x^{k-1} - \gamma \frac{1}{M} \sum_{i=1}^M \nabla f_i(x^{k-D_i^k})\right)$ |

These two algorithms are separated by a major difference: PIAG performs an aggregated delayed gradient descent from the most recent main variable $x^{k-1}$ and uses all gradients regardless of corresponding delays. Clearly, if one gradient has not

[3]For the master, the iteration $k$ reads $x^k = \mathrm{prox}_{\gamma g}(\overline{x}^k)$ where $\overline{x}^k$ is the average of the last update of each worker: $\overline{x}^k = \sum_{i=1}^M \pi_i x_i^k$ (see Eq. (3)). For each worker $i$, $x_i^k$ is the result of the last gradient step performed by this worker on its local function: $x_i^k = x^{k-D_i^k} - \gamma_i \nabla f_i(x^{k-D_i^k})$ (see Eq. (4)). Putting it all together, we get $x^k = \mathrm{prox}_{\gamma g}(\sum_{i=1}^M \pi_i x^{k-D_i^k} - \sum_{i=1}^M \pi_i \gamma_i \nabla f_i(x^{k-D_i^k}))$. Finally, this expression can be simplified by noticing that $\pi_i \gamma_i = \gamma / M$ (see Eqs. (5) and (7)).

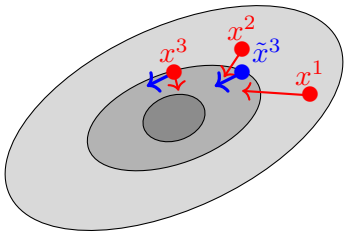



Fig. 2: Let the gray ellipses be the level-sets of a smooth function. In red are represented three iterates $(x^k)_{k=1,2,3}$ and their associated descent directions (taken as the opposite of the gradients computed at these points). The blue dots represent the averaged point $\tilde{x}^3 = (x^1 + x^2 + x^3)/3$, while the blue vectors both represent the average of the associated descent directions. We notice that in that situation, descending along the averaged gradient is much more interesting from the averaged point $\tilde{x}^3$ than from the last point $x^3$.

been updated for long time, this update rule may be harmful, as mentioned at the end of Section 2.2. On the other hand, DAve-(R)PG performs a similar aggregated delayed gradient descent (with more adaptive local stepsizes) but from the averaged main point $\sum_{i=1}^{M} \pi_i \overline{x}^{k-D_i^k}$. This more conservative update prevents instabilities in the case where some worker is silent for too long, and, thus, is more robust. See Figure 2 for a geometrical illustration.

In terms of theoretical results, this conservative approach allows us to get stronger convergence results and better rates as derived in the next section:

- the stepsize of PIAG, and, thus, its rate, depends heavily on the maximal delays whereas our stepsize does not depend on any form of delays;
- PIAG's stepsize is global and, thus, cannot adapt to each of the workers local functions, while we use locally adapted stepsizes;
- no version of PIAG exists with *multiple* proximal gradient steps before exchanging with the master.

In terms of performance, before more thorough comparisons, Fig. 3 gives an illustration of the benefits of the proposed approach compared to PIAG in terms of iterates behavior. In this plot, we consider two runs of DAve-RPG and PIAG applied to a two dimensional problem where one of the 5 functions/workers takes 10 times as much time to compute its update as the other workers and consequently produces more delayed updates. The objective used is a sum of 5 quadratics centered around different points and the initial point is (-20, -20) in all cases. Although the stepsize used for PIAG was 10 times smaller (due to its dependence to the delays), the iterates produced by PIAG show chaotic deviations from the optimal point while DAve-RPG steadily converges to the optimum.

## 3. Analysis.

**3.1. Revisiting the clock.** To the best of our knowledge, all papers on asynchronous distributed methods (except [25, 11, 12]) assume that delays are uniformly upper bounded by a constant. Moreover, the maximum stepsize is usually highly dependent on this upper bound. In the upcoming results, we show that our algorithm DAve-RPG converges without assuming bounded delays and with the stepsizes

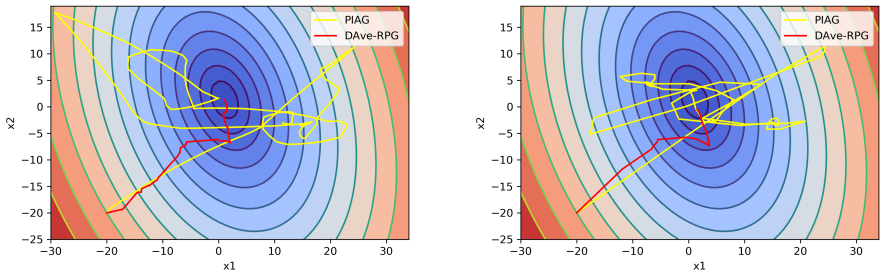



Fig. 3: Two runs of a two dimensional example with $n = 5$ and one worker suffering long delays.

depending only on local smoothness and convexity of the functions.

The forthcoming results are based on the careful definition of an *epoch sequence* along which we investigate the improvement of our algorithm (rather than looking at the improvement per iteration).

We define our *epochs sequence* $\{k_m\}_m$ by setting $k_0 = 0$ and the recursion:

$$\begin{aligned} k_{m+1} &= \min\{k : \text{ each machine made at least 2 updates on the interval } [k_m, k]\} \\ &= \min\{k : k - D_i^k \geq k_m \text{ for all } i = 1, .., M\} \end{aligned}$$

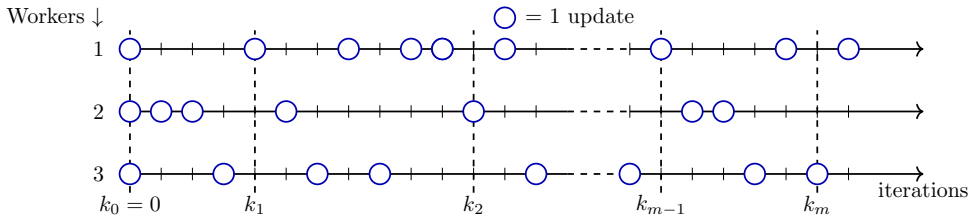



Fig. 4: Illustration of the epoch sequence for $M = 3$ workers. Each circle corresponds to one update, i.e, one iteration.

In words, $k_m$ is the first moment when all workers have updated twice since $k_{m-1}$. This is illustrated by Figure 4. Thus, $k_m$ is the first moment when $\overline{x}^k$ no longer depends directly on information from moments before $k_{m-1}$. Indeed, we have $\overline{x}^k = \sum_i \pi_i x_i^{k-d_i^k}$ and $x_i^{k-d_i^k}$ was computed using $\overline{x}^{k-D_i^k}$.

Note that we always have $k_m \geq 2M - 1$. Furthermore, in the degenerate case when $M = 1$, the epoch sequence corresponds to the time sequence: we have $k_m = m$, because on the interval $[m, m + 1]$ there are exactly two updates of the only slave.

In addition, we will assume that the number of epochs goes to infinity, i.e. all workers eventually respond, in order to get convergence. This is in line with the literature on totally asynchronous algorithms (see Assumption 1.1 in Chap. 6 of [4]). Nevertheless, our results in the strongly convex case (Theorem 3.2 and its corollary) are still valid even when there is a finite number of epochs; in that case, they mean that the iterates will reach a ball around the solution of a radius controlled by the number of epochs performed.

**3.2. Preliminary: local iterations.** To understand why the algorithm converges as a whole, let us first take a close look at how one local iteration of RPG enables iterates to get closer to a local solution. Indeed, a special property of the algorithm is that local variables $(x_i^k)$ do not converge to the same value as the master variable $\overline{x}^k$. In contrast, they go to the *local shifted optimal point* $x_i^\star := x^\star - \gamma_i \nabla f_i(x^\star)$.

At worker $i$ and time $k$, $x_i^k = x_i^{k-d_i^k}$ was obtained by $p = p(i, k-d_i^k)$ repetitions of proximal gradient. Starting with the reception of $\overline{x}^{k-D_i^k}$ and initializing $\Delta^{(0)} = 0$, the $p$ local iterations (indexed by superscripts with parentheses) are obtained by the repetition of

$$\begin{aligned} z^{(q)} &= \mathrm{prox}_{\gamma g}(\overline{x}^{k-D_i^k} + \Delta^{(q-1)}), \\ x_i^{(q)} &= z^{(q)} - \gamma_i \nabla f_i(z^{(q)}) \\ \Delta^{(q)} &= \Delta^{(q-1)} + \pi_i \left( x_i^{(q)} - x_i^{(q-1)} \right) \end{aligned}$$

for $q = 1, .., p$. Then, $x_i^{k-d_i^k} = x_i^{(p)}$ and $\Delta^{k-d_i^k} = \Delta^{(p)}$.

The next lemma is fundamental to the analysis of our algorithm. It describes how the local computations go towards their own local shifted optimal point, compared to

$$\mathbf{a}^k := \max\left( \left\|\overline{x}^k - \overline{x}^\star\right\|^2, \left\|\overline{x}^k_{-i(k)} - \overline{x}^\star_{-i(k)}\right\|^2 \right), \tag{8}$$

where $i(k)$ is the updating agent at time $k$ and

$$\overline{x}^\star = \sum_{i=1}^M \pi_i x_i^\star, \qquad \overline{x}^k_{-i} = \frac{1}{1-\pi_i} \sum_{j \neq i} \pi_j x_j^k, \qquad \overline{x}^\star_{-i} = \frac{1}{1-\pi_i} \sum_{j \neq i} \pi_j x_j^\star.$$

In addition, we have that $x^\star = \mathrm{prox}_{\gamma g}(\overline{x}^\star)$ by first-order optimality conditions of Problem (1).

LEMMA 3.1. *Let $f_i$ be $\mu_i$-strongly convex ($\mu_i \geq 0$) and $L_i$-smooth, $g$ be convex lsc. Then, with $\gamma_i \in (0, 2/(L_i + \mu_i)]$, we have for any $k$ that after $p_i^k$ repetitions*

$$\left\|x_i^k - x_i^\star\right\|^2 \leq (1 - \gamma_i \mu_i)^2 \, r_i(p_i^k)^2 \; \mathbf{a}^{k-D_i^k}$$

*with $r_i(p) = 1 - \gamma_i \mu_i \sum_{q=1}^{p-1} (1 - \gamma_i \mu_i)^{q-1} \pi_i^q$.*

*Furthermore, if $\mu_i = 0$, with $\gamma_i \in (0, 2/L_i)$, we have for any $k$ and any number of repetitions*

$$\left\|x_i^k - x_i^\star\right\|^2 \leq \mathbf{a}^{k-D_i^k} - \gamma_i \left( \frac{2}{L_i} - \gamma_i \right) \left\| \nabla f_i(z^{(p)}) - \nabla f_i(x^\star) \right\|^2$$

*where $z^{(p)}$ is such that $x_i^k = z^{(p)} - \gamma_i \nabla f_i(z^{(p)})$.*

*Proof.* First, as $f_i$ is $\mu_i$-strongly convex and $L_i$ smooth, we have that for any

$q = 1, .., p$ (see for instance [5, Chap. 3.4.2]),

$$\begin{aligned}
&\left\|x^{(q)} - x_i^\star\right\|^2 = \left\|z^{(p)} - \gamma_i \nabla f_i(z^{(q)}) - (x^\star - \gamma_i \nabla f_i(x^\star))\right\|^2 \\
&\leq \left(1 - \frac{2\gamma_i \mu_i L_i}{\mu_i + L_i}\right) \left\|z^{(q)} - x^\star\right\|^2 - \gamma_i \left(\frac{2}{\mu_i + L_i} - \gamma_i\right) \left\|\nabla f_i(z^{(q)}) - \nabla f_i(x^\star)\right\|^2 \qquad (9) \\
&\leq \left[\left(1 - \frac{2\gamma_i \mu_i L_i}{\mu_i + L_i}\right) - \mu^2 \gamma_i \left(\frac{2}{\mu_i + L_i} - \gamma_i\right)\right] \left\|z^{(q)} - x^\star\right\|^2 \\
&= (1 - \gamma_i \mu_i)^2 \left\|z^{(q)} - x^\star\right\|^2 . \qquad (10)
\end{aligned}$$

Then, for $q = 1$, we have by non-expansivity of the proximity operator that

$$\left\|z^{(1)} - x^\star\right\|^2 \leq \left\|\overline{x}^{k-D_i^k} - \overline{x}^\star\right\|^2$$

which completes the proof for $p = 1$. Going further, for $q \geq 2$, non-expansivity and Jensen's inequality yield

$$\begin{aligned}
\left\|z^{(q)} - x^\star\right\|^2 &\leq \left\|\overline{x}^{k-D_i^k} + \Delta^{(q-1)} - \overline{x}^\star\right\|^2 \\
&= \left\|\pi_i \left(x^{(q-1)} - x_i^\star\right) + \sum_{j \neq i} \pi_j \left(x_j^{k-D_i^k} - x_j^\star\right)\right\|^2 \\
&= \left\|\pi_i \left(x^{(q-1)} - x_i^\star\right) + (1 - \pi_i) \left(\overline{x}_{-i}^{k-D_i^k} - \overline{x}_{-i}^\star\right)\right\|^2 \\
&\leq \pi_i \left\|x^{(q-1)} - x_i^\star\right\|^2 + (1 - \pi_i) \left\|\overline{x}_{-i}^{k-D_i^k} - \overline{x}_{-i}^\star\right\|^2 .
\end{aligned}$$

Then by induction, using the triangle inequality instead of convexity, one gets that for $p \geq 2$ (and using $\beta_i = (1 - \gamma_i \mu_i)\pi_i$)

$$\begin{aligned}
\left\|z^{(p)} - x^\star\right\| &\leq \pi_i \left\|x^{(p-1)} - x_i^\star\right\| + (1 - \pi_i) \left\|\overline{x}_{-i}^{k-D_i^k} - \overline{x}_{-i}^\star\right\| \\
&\leq \beta_i \left\|z^{(p-1)} - x^\star\right\| + (1 - \pi_i) \sqrt{\mathbf{a}^{k-D_i^k}} \\
&\leq \beta_i^{p-1} \left\|z^{(1)} - x^\star\right\| + \left[\sum_{q=1}^{p-1} \beta_i^{q-1} (1 - \pi_i)\right] \sqrt{\mathbf{a}^{k-D_i^k}} \\
&\leq \beta_i^{p-1} \left\|\overline{x}^{k-D_i^k} - x^\star\right\| + \left[\sum_{q=1}^{p-1} \beta_i^{q-1} (1 - \pi_i)\right] \sqrt{\mathbf{a}^{k-D_i^k}} \qquad (11) \\
&\leq \beta_i^{p-1} \sqrt{\mathbf{a}^{k-D_i^k}} + \left[\sum_{q=1}^{p-1} \beta_i^{q-1} (1 - \pi_i)\right] \sqrt{\mathbf{a}^{k-D_i^k}} \\
&= \left[\beta_i^{p-1} + \sum_{q=0}^{p-2} \beta_i^q - \frac{1}{1 - \gamma_i \mu_i} \sum_{q=1}^{p-1} \beta_i^q\right] \sqrt{\mathbf{a}^{k-D_i^k}} \\
&= \underbrace{\left[1 - \frac{\gamma_i \mu_i}{1 - \gamma_i \mu_i} \sum_{q=1}^{p-1} \beta_i^q\right]}_{= r_i(p)} \sqrt{\mathbf{a}^{k-D_i^k}}
\end{aligned}$$

noting that $i = i(k - D_i^k)$ was updating at time $k - D_i^k$ by definition. Using the last inequality on top of (9) or (10) leads to the claim, noting that $r_i(p) = 1$ for all $p$ when $\mu_i = 0$. ∎

**3.3. Convergence results.** In this section, we analyze the convergence of our algorithm, first in the strongly convex case, and second in the general case. In both cases, our results allow us to choose the same stepsize as for vanilla gradient descent (without any dependence on the delays). The derived rates involve the *number of epochs* rather than the number of iterations. In Section 3.4, we examine how these rates translate in terms of number of iteration under boundedness of the delays in order to compare with the literature.

**3.3.1. Linear convergence in the strongly convex case.** If all the local functions $(f_i)$ are strongly convex, the convergence of our algorithm is linear on the epoch sequence.

THEOREM 3.2 (Strongly convex case). *Let the functions $(f_i)$ be $\mu_i$-strongly convex ($\mu_i > 0$) and $L_i$-smooth. Let $g$ be convex lsc. Using $\gamma_i \in (0, \frac{2}{\mu_i + L_i}]$, DAve-RPG converges linearly on the epoch sequence $(k_m)$, with the rate $\rho := \min_i \gamma_i \mu_i$. More precisely, for all $k \in [k_m, k_{m+1})$*

$$\left\|x^k - x^\star\right\|^2 \leq (1-\rho)^{2m} \max_i \left\|x_i^0 - x_i^\star\right\|^2,$$

*with the shifted local solutions $x_i^\star = x^\star - \gamma_i \nabla f_i(x^\star)$.*

*Proof.* First, for any $i$ and any $k \in [k_m, k_{m+1})$, we have from Lemma 3.1

$$\left\|x_i^k - x_i^\star\right\|^2 = (1-\gamma_i\mu_i)^2 r_i(p_i^k)^2 \ \ \mathbf{a}_i^{k-D_i^k} \leq (1-\rho)^2 \ \ \mathbf{a}_i^{k-D_i^k}$$

Thus, for any $k \in [k_m; k_{m+1})$,

$$\begin{aligned}\left\|\overline{x}^k - \overline{x}^\star\right\|^2 &\leq \sum_{i=1}^M \pi_i \|x_i^k - x_i^\star\|^2 = \sum_{i=1}^M \pi_i \|x_i^{k-d_i^k} - x_i^\star\|^2 \\ &\leq (1-\rho)^2 \ \ \sum_{i=1}^M \pi_i \mathbf{a}^{k-D_i^k} \leq (1-\rho)^2 \ \ \max_i \mathbf{a}^{k-D_i^k} \end{aligned} \tag{12}$$

Similarly, for any $j$

$$\left\|\overline{x}_{-j}^k - \overline{x}_{-j}^\star\right\|^2 \leq (1-\pi_j)^{-1} \sum_{i \neq j} \pi_i \left\|x_i^{k-d_i^k} - x_i^\star\right\|^2 \leq (1-\rho)^2 \ \ \max_i \mathbf{a}^{k-D_i^k}. \tag{13}$$

Finally, we get

$$\mathbf{a}^k \leq (1-\rho)^2 \max_i \mathbf{a}^{k-D_i^k}$$

which is the workhorse for the rest of the proof.

Let $m > 0$ and $k \in [k_m, k_{m+1})$, then the definition of the epoch sequence $(k_m)$ gives $k - D_i^k \geq k_{m-1}$ and then

$$\mathbf{a}^k \leq (1-\rho)^2 \max_i \mathbf{a}^{k-D_i^k} \leq (1-\rho)^2 \max_{k' \in [k_{m-1}, k)} \mathbf{a}^{k'}$$

and applying this inequality sequentially to $k_m, k_m+1, \dots, k_{m+1}-1$, we get

$$
\begin{aligned}
\mathbf{a}^{k_m} &\leq (1-\rho)^2 \max_{k' \in [k_{m-1}, k_m)} \mathbf{a}^{k'}, \qquad (14)\\
\mathbf{a}^{k_m+1} &\leq (1-\rho)^2 \max\left( \max_{k' \in [k_{m-1}, k_m)} \mathbf{a}^{k'}, \mathbf{a}^{k_m} \right)\\
&\leq (1-\rho)^2 \max_{k' \in [k_{m-1}, k_m)} \mathbf{a}^{k'} \qquad \text{(using Eq. (14))}\\
&\dots\\
\max_{k \in [k_m, k_{m+1})} \mathbf{a}^{k} &\leq (1-\rho)^2 \max_{k' \in [k_{m-1}, k_m)} \mathbf{a}^{k'}\\
&\leq (1-\rho)^{2m} \max_{k' < k_0} \mathbf{a}^{k'} \leq (1-\rho)^{2m} \max_i \left\| x_i^0 - x_i^\star \right\|^2 .
\end{aligned}
$$

Finally, since the proximity operator of a convex function is non-expansive, we have for all $k \in [k_m; k_{m+1})$,

$$
\begin{aligned}
\|x^k - x^\star\|^2 = \|\operatorname{prox}_{\gamma g}(\overline{x}^k) - \operatorname{prox}_{\gamma g}(\overline{x}^\star)\|^2 &\leq \|\overline{x}^k - \overline{x}^\star\|^2\\
&\leq \max_{k \in [k_m, k_{m+1})} \mathbf{a}^k \leq (1-\rho)^{2m} \max_i \left\| x_i^0 - x_i^\star \right\|^2
\end{aligned}
$$

which concludes the proof. ∎

Notice that the rate provided by this theorem is valid for any choice of number of local iterations at any worker/time. The local contraction at agent $i$ can indeed be improved by doing $p$ local repetitions by a factor

$$
r_i(p) = 1 - \gamma_i \mu_i \sum_{q=1}^{p-1} (1-\gamma_i\mu_i)^{q-1} \pi_i^q = 1 - \gamma_i \mu_i \pi_i \frac{1-(1-\gamma_i\mu_i)^{p-1}\pi_i^{p-1}}{1-(1-\gamma_i\mu_i)\pi_i}
$$

where $r_i(1) = 1$ and $r_i$ is decreasing with $p$ and lower-bounded by

$$
r_i(\infty) = 1 - \frac{\gamma_i \mu_i \pi_i}{1-(1-\gamma_i\mu_i)\pi_i}.
$$

If all workers, or at least the ones with the slowest rates, perform several local iterations, the rate can thus be improved as stated by the following result. However, local iterations practically slow down the actual time between two epochs thus the number of local repetitions have to be carefully tuned in practice. The flexibility allowed by our algorithm enables a wide range of selection strategies such as online tuning, stopping the local iterations after some fixed time, etc.

Corollary 3.3 (Tighter rates for the strongly convex case). *Let the functions $(f_i)$ be $\mu_i$-strongly convex ($\mu_i > 0$) and $L_i$-smooth. Let $g$ be convex lsc. Using $\gamma_i \in (0, \frac{2}{\mu_i + L_i}]$, DAve-RPG converges linearly on the epoch sequence $(k_m)$, in the sense that for all $k \in [k_m, k_{m+1})$*

$$
\left\| x^k - x^\star \right\|^2 \leq (\Pi_{\ell=1}^m \alpha_\ell) \max_i \left\| x_i^0 - x_i^\star \right\|^2 ,
$$

*with $\alpha_\ell = \max_{i, k \in [k_\ell, k_{\ell+1})} (1-\gamma_i\mu_i)^2 r_i(p_i^k)^2$ and $x_i^\star = x^\star - \gamma_i \nabla f_i(x^\star)$.*

*In particular, the rate can be uniformly improved to $\alpha = \max_{i,k} (1-\gamma_i\mu_i)^2 r_i(p_i^k)^2$.*

**3.3.2. Convergence and sublinear rate in the general case.** When Problem (1) is not strongly convex, iterates convergence still holds with the fixed usual stepsizes at the expense of a sublinear rate.

THEOREM 3.4 (Convergence in the general case). *Let $(f_i)$ be convex $L_i$-smooth, $g$ be convex lsc, and $\gamma_i \in (0, 2/L_i)$. Then, if $x^\star$ is the unique minimizer of (1), the sequence $(x^k)$ converges to $x^\star$. Moreover, if Problem (1) has multiples minimizers, then $(x^k)$ still converges to a minimizer of (1), under two additional assumptions: (i) the difference between two consecutive epochs $k_m - k_{m-1}$ is uniformly bounded, (ii) the number of inner loops is uniformly bounded.*

From a mathematical point of view, this result and its proof are the main technical novelties of this paper. We put below the proof of the first part of the result: convergence under no additional assumptions when (1) has a unique minimizer. For readability, we postpone to Appendix A the proof the second part when (1) has multiple minimizers. Note that this second part requires an assumption on bounded delays (see more in the discussion of Section 3.4) but no knowledge about this bound (which does not appears in the stepsize range or in the proof).

*Proof.* For any $i$ and any $k \in [k_m; k_{m+1})$, we have from Lemma 3.1

$$\left\|x_i^k - x_i^\star\right\|^2 \leq \mathbf{a}^{k-D_i^k} - \gamma_i \left(\frac{2}{L_i} - \gamma_i\right) \left\|\nabla f_i(z^{(p)}) - \nabla f_i(x^\star)\right\|^2 \tag{15}$$

where $\mathbf{a}^{k-D_i^k}$ is the error at time $k - D_i^k$ (see (8)) and $z^{(p)}$ is such that $x_i^k = z^{(p)} - \gamma_i \nabla f_i(z^{(p)})$. Thus, as in Theorem 3.2, for any $k \in [k_m; k_{m+1})$, we have by dropping the last term

$$\left\|\overline{x}^k - \overline{x}^\star\right\|^2 \leq \sum_{i=1}^M \pi_i \|x_i^k - x_i^\star\|^2 = \sum_{i=1}^M \pi_i \|x_i^{k-d_i^k} - x_i^\star\|^2 \leq \sum_{i=1}^M \pi_i \mathbf{a}^{k-D_i^k} \leq \max_i \mathbf{a}^{k-D_i^k}.$$

Similarly, for any $j$

$$\left\|\overline{x}_{-j}^k - \overline{x}_{-j}^\star\right\|^2 \leq (1-\pi_j)^{-1} \sum_{i \neq j} \pi_i \left\|x_i^{k-d_i^k} - x_i^\star\right\|^2 \leq \max_i \mathbf{a}^{k-D_i^k}.$$

Finally, we get $\mathbf{a}^k \leq \max_i \mathbf{a}^{k-D_i^k}$ from which we can prove using the same arguments as in the proof of Theorem 3.2

$$\max_{k \in [k_m, k_{m+1})} \mathbf{a}^k \leq \max_{k' \in [k_{m-1}, k_m)} \mathbf{a}^{k'},$$

which means that

$$\mathbf{b}^m := \max_{k \in [k_m, k_{m+1})} \mathbf{a}^k \tag{16}$$

is non-increasing, so that it converges to a non-negative value $\mathbf{b}$. Getting back to (15), we get that for any $i$ and any $k \in [k_m; k_{m+1})$,

$$\left\|x_i^k - x_i^\star\right\|^2 \leq \mathbf{a}^{k-D_i^k} \leq \mathbf{b}^{m-1}$$

thus when $m \to \infty$, we get that

$$\limsup_k \left\|x_i^k - x_i^\star\right\|^2 \leq \mathbf{b}. \tag{17}$$

The remainder of the proof consists in proving that $\mathbf{b} = 0$.

Let $(l^m)$ be a time sequence realizing the max in (16), i.e.

$$l^m \in \operatorname*{argmax}_{k \in [k_m, k_{m+1})} \mathbf{a}^k \tag{18}$$

then, be get that $\mathbf{a}^{l^m} \to \mathbf{b}$ as $m \to \infty$. We have now two cases: 1) when $\mathbf{a}^{l^m} = \|\overline{x}^{l^m}_{-i(l^m)} - \overline{x}^\star_{-i(l^m)}\|^2$ infinitely often; and 2) when $\mathbf{a}^{l^m} = \|\overline{x}^{l^m} - \overline{x}^\star\|^2$ infinitely often.

We can show that the first case is impossible. In order to ease the reading, we report the proof of this statement at the end of the proof. So we consider now that the sequence

$$l_1^m \in \arg\max_{k \in [k_m, k_{m+1})} \mathbf{a}^k \text{ if } \mathbf{a}^{l^m} = \left\|\overline{x}^{l^m} - \overline{x}^\star\right\|^2 \quad \text{and} \quad l_1^m = l_1^{m-1} \text{ otherwise}$$

and we have that $l_1^m \to \infty$ when $m \to \infty$. We can extract a subsequence $(s^m)$ of $(l_1^m)$ such that $(\overline{x}^{s^m}, x_1^{s^m}, .., x_M^{s^m})$ converges to $(\overline{x}, x_1, .., x_M)$ with $\overline{x} = \sum_{i=1}^M \pi_i x_i$. We are going to show that these points are the limits of all the sequences. Later, the associated $x := \operatorname{prox}_{\gamma g}(\overline{x})$ will also come into play.

We first observe that

$$\mathbf{b} = \lim_{m\to\infty} \mathbf{a}^{l_1^m} = \lim_{m\to\infty} \left\|\overline{x}^{l_1^m} - \overline{x}^\star\right\|^2 \leq \sum_{i=1}^M \pi_i \limsup_{k\to\infty} \left\|x_i^k - x_i^\star\right\|^2 \leq \mathbf{b}.$$

This tells us that

$$\mathbf{b} = \|\overline{x} - \overline{x}^\star\|^2 = \left\|\sum_{i=1}^M \pi_i (x_i - x_i^\star)\right\|^2 \leq \sum_{i=1}^M \pi_i \|x_i - x_i^\star\|^2 \leq \mathbf{b}$$

and this inequality can only be satisfied if $x_i - x_i^\star = x_j - x_j^\star$ for any $i, j$ by direct computation (see e.g. [3, Lemma 2.13]). Thus,

$$x_i - x_i^\star = \sum_{j=1}^M \pi_j (x_j - x_j^\star) = \overline{x} - \overline{x}^\star \tag{19}$$

which leads to

$$\overline{x}^{s^m} - x_i^{s^m} \to \overline{x} - x_i = \overline{x}^\star - x_i^\star. \tag{20}$$

We turn now our attention to convergence of gradients at times $s^m$. Rearranging (15) and taking the limit, we get first

$$\limsup_m \left\|\nabla f_i(z_i^{s^m}) - \nabla f_i(x^\star)\right\|^2 \leq \frac{1}{\omega_i}\left(\lim_m \mathbf{b}^m - \lim_m \left\|x_i^{s^m} - x_i^\star\right\|^2\right) = 0$$

where $z_i^{s^m}$ is such that $x_i^{s^m} = z_i^{s^m} - \gamma_i \nabla f_i(z_i^{s^m})$ and $\omega_i = \gamma_i(2/L_i - \gamma_i)$. Thus,

$$\nabla f_i(z_i^{s^m}) \to \nabla f_i(x^\star) \tag{21}$$

and by definition we get

$$z_i^{s^m} = x_i^{s^m} + \gamma_i \nabla f_i(z_i^{s^m}) \to x_i + \gamma_i \nabla f_i(x^\star).$$

Define for each $i$ and $k$ vector $w_i^k$ as the one used to get $z_i^k$, i.e. $z_i^k = \mathrm{prox}_{\gamma g}\left(w_i^k\right)$. Using the firm non-expansiveness of the proximal operator $g$ (see [3, Lemma 12.27]), we obtain

$$\begin{aligned}\left\|x_i^k - x_i^\star\right\|^2 \leq \left\|z_i^k - x^\star\right\|^2 &= \left\|\mathrm{prox}_{\gamma g}(w_i^k) - \mathrm{prox}_{\gamma g}(\overline{x}^\star)\right\|^2 \\ &\leq \left\|w_i^k - \overline{x}^\star\right\|^2 - \left\|z_i^k - w_i^k - (x^\star - \overline{x}^\star)\right\|^2 \\ &\leq \mathbf{a}^{k-D_i^k} - \left\|z_i^k - w_i^k - (x^\star - \overline{x}^\star)\right\|^2,\end{aligned}$$

which yields

$$\limsup_m \left\|z_i^{s^m} - w_i^{s^m} - (x^\star - \overline{x}^\star)\right\|^2 \leq \lim_m \mathbf{b}^m - \lim_m \left\|x_i^{s^m} - x_i^\star\right\|^2 = 0. \tag{22}$$

This yields in turn, by (19), as $w_i^{s^m} - z_i^{s^m} + (x^\star - \overline{x}^\star) \to 0$, that

$$w_i^{s^m} \to x_i + \gamma_i \nabla f_i(x^\star) - (x^\star - \overline{x}^\star) = (x_i^\star + \overline{x} - \overline{x}^\star) + \gamma_i \nabla f_i(x^\star) - (x^\star - \overline{x}^\star) = \overline{x}.$$

To finish the proof, we consider the point $x = \mathrm{prox}_{\gamma g}(\overline{x})$, and we observe that the non-expansiveness of $\mathrm{prox}_{\gamma g}$ gives

$$\begin{aligned}&\left\|x^{s^m} - x\right\| = \left\|\mathrm{prox}_{\gamma g}(\overline{x}^{s^m}) - \mathrm{prox}_{\gamma g}(\overline{x})\right\| \leq \left\|\overline{x}^{s^m} - \overline{x}\right\| \to 0 \\ \text{and} \quad &\left\|z_i^{s^m} - x\right\| = \left\|\mathrm{prox}_{\gamma g}(\overline{w}_i^{s^m}) - \mathrm{prox}_{\gamma g}(\overline{x})\right\| \leq \left\|\overline{w}_i^{s^m} - \overline{x}\right\| \to 0 \text{ for any } i.\end{aligned}$$

Therefore, the $L_i$-Lipschitz continuity of $\nabla f_i$ gives for any $i$,

$$\left\|\nabla f_i(z_i^{s^m}) - \nabla f_i(x)\right\| \leq L_i \left\|z_i^{s^m} - x\right\| \to 0$$

so $\nabla f_i(x) = \nabla f_i(x^\star)$ using (21). Finally, (22) also gives us that $\lim_m \|z_i^{s^m} - w_i^{s^m} - (x^\star - \overline{x}^\star)\|^2 = \|x - \overline{x} - (x^\star - \overline{x}^\star)\|^2 = 0$ so $x - \overline{x} = x^\star - \overline{x}^\star$.

Thus, for any $i$, we get from the definitions of $x^\star, \overline{x}^\star, x_i^\star = x^\star - \gamma_i \nabla f_i(x^\star)$ and the characterization $x = \mathrm{prox}_{\gamma g}(\overline{x}) \Leftrightarrow x + \gamma \partial g(x) \ni \overline{x}$ that

$$\begin{aligned}\gamma \nabla f(x) = \sum_{i=1}^M \pi_i \gamma_i \nabla f_i(x) &= \sum_{i=1}^M \pi_i \gamma_i \nabla f_i(x^\star) = \sum_{i=1}^M \pi_i (x^\star - x_i^\star) \\ &= x^\star - \overline{x}^\star = x - \overline{x} \in \gamma \partial g(x)\end{aligned}$$

thus $0 \in \partial(f+g)(x)$. We can conclude by using the unique minimizer assumption on $f+g$: we get indeed that $x = x^\star$, so $\overline{x} = \overline{x}^\star$. This leads to

$$\mathbf{b} = \lim_m \left\|\overline{x}^{s^m} - \overline{x}^\star\right\|^2 = 0$$

which directly implies that $x^k \to x^\star$, and ends the proof. Note that we use the fact that we are in the case of unique minimizer only here for the final conclusion.

*Proof of the statement that $l_1^m \not\to \infty$ when $m \to \infty$ is impossible.*
In this case, we have $\limsup_{m\to\infty} \|\overline{x}^{l^m} - \overline{x}^\star\|^2 < \mathbf{b}$. Introducing

$$l_2^m \in \arg\max_{k \in [k_m, k_{m+1})} \mathbf{a}^k \text{ if } \mathbf{a}^{l^m} = \left\|\overline{x}_{-i(l^m)}^{l^m} - \overline{x}_{-i(l^m)}^\star\right\|^2 \quad \text{and} \quad l_2^m = l_2^{m-1} \text{ otherwise}$$

we have that $l_2^m \to \infty$. We also have

$$\mathbf{b} = \lim_{m\to\infty} \mathbf{a}^{l_2^m} = \lim_{m\to\infty} \left\| \overline{x}^{l_2^m}_{-i(l_2^m)} - \overline{x}^\star_{-i(l_2^m)} \right\|^2 \leq \mathbf{b};$$

and we are going to show that it leads to a contradiction.

We extract a subsequence $(s_2^m)$ from $(l_2^m)$ such that $i(s_2^m) = i$ is fixed and and $(\overline{x}^{s_2^m}, x_1^{s_2^m}, .., x_M^{s_2^m})$ converge to $(\overline{x}, x_1, .., x_M)$. Using $\|\overline{x}^{s_2^m}_{-i} - \overline{x}^\star_{-i}\|^2 \to \mathbf{b}$, one can repeat the arguments of the other case to prove that for any $j, \ell \neq i$

$$x_j - x_j^\star = x_\ell - x_\ell^\star. \tag{23}$$

We would like to have this property for another $i' \neq i$, so that we would have equality of all the $x_j - x_j^\star$ which would yield

$$\|\overline{x} - \overline{x}^\star\|^2 = \left\| \sum_{j=1}^M \pi_j \left(x_j - x_j^\star\right) \right\|^2 = \sum_{j=1}^M \pi_j \left\|x_j - x_j^\star\right\|^2 = \mathbf{b}$$

and then contradicts $\limsup_{m\to\infty} \|\overline{x}^{l^m} - \overline{x}^\star\|^2 < \mathbf{b}$.

We have left to prove the existence of this second machine $i'$ with the same property (23). If the machine $i$ is the only machine that is making updates infinitely many times on times $s_2^m$, we have that for any $j \neq i$, $\|x_j^{s_2^m} - x_j^\star\|^2 \to \mathbf{b}$. From Lemma 3.1, it follows that $\|\overline{x}^{s_2^m - D_j^{s_2^m}}_{-j} - \overline{x}^\star_{-j}\|^2 \to \mathbf{b}$ so we can unite the two sequences $(s_2^m)$ and $(s_2^m - D_j^{s_2^m})$ to get a new sequence with the same properties but two slaves making updates infinitely many times. Without loss of generality, we then have that at least workers $i$ and $i'$ and then we get (3), and the contradiction follows. ∎

Besides convergence, we can also establish the rate of our algorithm in the general case, showing that it matches the one of vanilla gradient descent along the epoch sequence. The proof of this result is reported in Appendix B.

THEOREM 3.5 (Rate of convergence). *Let the functions $(f_i)$ be convex $L_i$-smooth and $g$ be convex lsc. Then, for $\gamma_i \in (0, 2/L_i)$ and any $k \in [k_m, k_{m+1})$*

$$\min_{k'\leq k} \|\partial F(x^{k'})\| \leq \frac{2\sqrt{2}}{\sqrt{m}} \frac{\max_i \left\|x_i^0 - x_i^\star\right\|}{\min_j \left(\gamma_j \sqrt{2 - \gamma_j L_j}\right)},$$

*where $\|\partial F(x^{k'})\| \coloneqq \min_{h\in\partial F(x^{k'})} \|h\|$.*

**3.4. Comparison of the results with the literature.** The main feature of the epoch sequence introduced in Section 3.1 is that it automatically adapts to variations of behaviors of machines across time (such as one worker being slow at first that gets faster with time). The sequence then allows for a intrinsic convergence analysis without any knowledge of the delays, as shown in the previous sections. This simple but powerful remark is one of the main technical contributions of this paper. For comparisons with the literature, the following result provides explicit connections between number of iterations and number of epochs with two standard bounds on delays uniformly in time[4].

[4]A notable exception allowing for potentially unbounded delays is the preprint [10] (more precisely Assumption A). However, in that paper the delays are seen as random variables and bounded in $L^p$ and thus differ from the deterministic treatment we propose.

PROPOSITION 3.6 (epoch scaling with delays). *For $M > 1$ machines[5], uniformly over time:*

- *if the delays are uniformly bounded by $d$ over the workers, i.e. $d_i^k \leq d$ for all $i$, then $d \geq M$ and the epoch sequence has complexity $k_m = \mathcal{O}(mM)$;*
- *if the average delay is bounded by $\overline{d}$, i.e. $1/M \sum_{i=1}^{M} d_i^k \leq \overline{d}$, then $\overline{d} \geq (M-1)/2$ and the epoch sequence has complexity $k_m = \mathcal{O}(mM)$.*

The proof of this proposition is basic and reported in Appendix C. The detailed results are summarized in the following table.

| | uniform bound | average bound |
|---|---|---|
| Condition | $d_i^k \leq d$ for all $i$ | $\frac{1}{M}\sum_{i=1}^{M} d_i^k \leq \overline{d}$ |
| Unimprov. bound | $d = M + \tau; \quad \tau \geq 0$ | $\overline{d} = \frac{M-1}{2} + \tau; \quad \tau \geq 0$ |
| 1 Epoch | $k_{m+1} - k_m \leq 2d + 1$ | $k_{m+1} - k_m \leq 2M(2\overline{d} - M + 3) - 3$ |
| Epoch sequence | $k_m \leq (2M + 2\tau + 1)m$ | $k_m \leq 4M(\tau + 1)m$ |

Bounding the average delay among the workers is an attractive assumption which is however much less common in the literature. The defined epoch sequence and associated analysis subsumes this kind of assumption.

In the case of uniformly bounded delays, the derived link between epoch and time sequence enables us to compare our rates in the strongly convex case (Theorem 3.2) with the ones obtained for PIAG [1, 28, 29]. To simply the comparison, let us consider the case where all the workers share the same strong convexity and smoothness constants $\mu$ and $L$. The first thing to notice is that the admissible stepsize for PIAG depends on the delays' uniform upper bound $d$ which is practically concerning, while the usual proximal gradient stepsizes are used for the proposed DAve-RPG. Using the optimal stepsizes in each case, the convergence rates in terms of time $k$ are:

| | DAve-RPG | PIAG |
|---|---|---|
| Reference | Th. 3.2 | Th. 3.4 of [29] |
| Stepsize | $\gamma = \frac{2}{\mu + L}$ | $\gamma = \frac{16}{\mu}\left[(1 + \frac{\mu}{48L})^{\frac{1}{d+1}} - 1\right]$ |
| Rate | $\left(1 - \frac{2}{1+\frac{L}{\mu}}\right)^{\frac{k}{d+0.5}}$ | $\left(1 - \frac{1}{49\frac{L}{\mu}}\right)^{\frac{k}{d+1}}$ |

We notice in both cases the exponent inversely proportional to the maximal delay $d$ but the term inside the parenthesis is a hundred times smaller for PIAG. Even if our algorithm is made for handling the flexible delays, this comparison illustrates the interest of our approach over PIAG for distributed asynchronous optimization in the case of bounded delays.

**4. Numerical Illustrations.** In this section, we run some numerical experiments to illustrate the behavior of our algorithm in the general convex case: we compare with the synchronous version and state-of-the-art method PIAG; we also point out the effect of repeated local iterations. These experiments complement the ones of the companion short paper [18] which presents results for strongly convex function, different worker loads, and increasing number of machines.

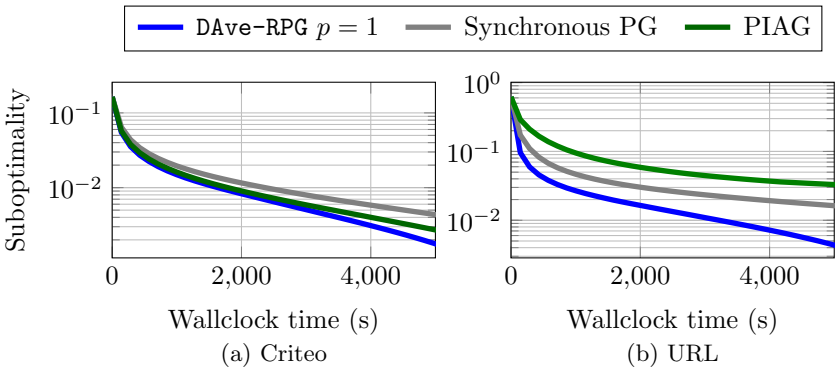
[5]For $M = 1$ machine, we have $k_m = m$ as mentioned in Section 3.1 and we recover exactly the convergence rates of the vanilla proximal gradient.

We consider the problem of minimizing the logistic loss with the $\ell_1$ and $\ell_2$-regularization on a dataset split among the workers. The problem reads

$$\min_{x\in\mathbb{R}^n} \frac{1}{M}\sum_{i=1}^{M} \underbrace{\sum_{j\in\mathcal{S}_i} \log\left(1+\exp(-b_j a_j^\top x)\right) + \frac{\lambda_2}{2}\|x\|_2^2}_{f_i(x)} + \lambda_1\|x\|_1,$$

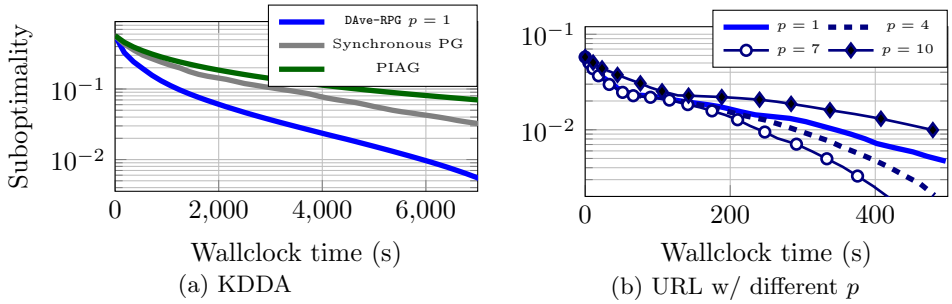
where for each example $j$, the pair $(a_j, b_j)$ represents the features $a_j \in \mathbb{R}^n$ together with the corresponding label $b_j \in \{-1, 1\}$; and $\mathcal{S}_i$ represents the examples stored locally at machine $i$; the total number of examples is denoted by $m$.

The experiments were run on a CPU cluster, one core corresponding to one worker. Each core had 4 GB of memory and used one thread to produce updates. The code was written in Python using standard libraries only. The datasets used for the experiments are Criteo ($n = 1,000,000$, $m = 45,840,617$), URL ($n = 3,231,961$, $m = 2,396,130$), and KDDA ($n = 20,216,830$, $m = 8,407,752$) from the LIBSVM datasets library [7].




(a) Criteo (b) URL

Fig. 5: Performance on general convex functions ($\lambda_2 = 0$).




(a) KDDA (b) URL w/ different $p$

Fig. 6: Performance on strongly convex functions.

In Fig. 5, we plot the suboptimality versus wallclock time for the proposed DAve-RPG with $p = 1$, the usual synchronous proximal gradient, and PIAG [1]. For each of the datasets, we use the first 100,000 features, and split evenly the examples over 50 workers. We take $\lambda_1 = 10^{-11}$ and $10^{-7}$ respectively and $\lambda_2 = 0$ for both. As

we do not use $\ell_2$-regularization, the problem is not strongly convex and the rate is not linear. However, it is clear that, just as the synchronous proximal gradient descent, DAve-RPG appears to converge with rate $O(\frac{1}{k})$, in line with Theorem 3.5. For all algorithms, we used the maximal stepsize (for PIAG, we took the limit $\mu \to 0$ in [1]). Even in this case where the workers have similar computational loads, the performance of DAve-RPG is clearly better than that of the synchronous gradient descent. DAve-RPG also outperforms PIAG, notably thanks to its robustness (as expected from Fig. 3).

In Fig. 6a, we use a non-zero $\ell_2$-regularization, leading to a strongly convex problem: we plot the suboptimality versus wallclock time for the proposed DAve-RPG with $p = 1$, the usual synchronous proximal gradient, and PIAG for the KDDA dataset. We use the first 200,000 features, and split evenly the examples over 60 workers. In this, the performance gain brought by DAve-RPG is even more significant. Finally, in Fig. 6b, we illustrate the repetition of local iterations: we plot the suboptimality versus wallclock time for the proposed DAve-RPG with $p = 1, 4, 7, 10$ on the full URL dataset with $\lambda_1 = 10^{-6}$ and $\lambda_2 = 1/m$ split evenly over 100 workers. We see that a tradeoff appears between computation and communications/updates; in this particular case, the performance improves up to $p = 7$ and then degrades afterwards.

**5. Conclusions.** This paper describes a novel algorithm for asynchronous distributed optimization. A key property of this algorithm is that it does not require unrealistic assumptions on machine delays. It is based on two original algorithmic features. First, the master machine keeps a combination of the output of all the workers last repeated proximal gradient steps, whereas for most algorithms in the literature, the master performs a step using the last gradients computed by the workers. Second, the workers can freely choose how many proximal gradient repetitions they make, leading to scarcer exchanges and more flexible communications.

These special features lead us to two key theoretical findings: i) an epoch-based analysis adapted to any kind of delays; and ii) the use of the same stepsizes as in the classical proximal gradient algorithm. We proved the convergence of the algorithm in the general case and with a linear rate in the strongly convex case. Although long delays may slow down the algorithm, it still converges both in theory and in experiments without being biased by more frequently updating workers.

The analysis suggests that some of the provided ideas may be used if updates are performed differently. Just in the way the vanilla proximal-gradient algorithm and its analysis form a base for studying advanced methods, we believe that the proposed algorithm and its original analysis may serve for future works in distributed optimization.

**Acknowledgments.** We thank Robert Gower for valuable comments on the first versions of this paper.

REFERENCES

[1] A. Aytekin, H. R. Feyzmahdavian, and M. Johansson, *Analysis and implementation of an asynchronous optimization algorithm for the parameter server*, arXiv:1610.05507, (2016).

[2] F. Bach, R. Jenatton, J. Mairal, G. Obozinski, et al., *Optimization with sparsity-inducing penalties*, Foundations and Trends® in Machine Learning, 4 (2012), pp. 1–106.

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

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

[5] S. Bubeck et al., *Convex optimization: Algorithms and complexity*, Foundations and Trends® in Machine Learning, 8 (2015), pp. 231–357.
[6] C. Calauzènes and N. L. Roux, *Distributed saga: Maintaining linear convergence rate with limited communication*, arXiv:1705.10405, (2017).
[7] C.-C. Chang and C.-J. Lin, *Libsvm: a library for support vector machines*, ACM transactions on intelligent systems and technology (TIST), 2 (2011), p. 27.
[8] A. Defazio, F. Bach, and S. Lacoste-Julien, *Saga: A fast incremental gradient method with support for non-strongly convex composite objectives*, in Advances in Neural Information Processing Systems, 2014, pp. 1646–1654.
[9] A. Defazio, J. Domke, and T. Caetano, *Finito: A faster, permutable incremental gradient method for big data problems*, in Proceedings of the 31st international conference on machine learning (ICML-14), 2014, pp. 1125–1133.
[10] J. C. Duchi, S. Chaturapruek, and C. Ré, *Asynchronous stochastic convex optimization*, arXiv preprint arXiv:1508.00882, (2015).
[11] R. Hannah and W. Yin, *More iterations per second, same quality–why asynchronous algorithms may drastically outperform traditional ones*, arXiv:1708.05136, (2017).
[12] R. Hannah and W. Yin, *On unbounded delays in asynchronous parallel fixed-point algorithms*, Journal of Scientific Computing, 76 (2018), pp. 299–326.
[13] R. Johnson and T. Zhang, *Accelerating stochastic gradient descent using predictive variance reduction*, in Advances in neural information processing systems, 2013, pp. 315–323.
[14] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, *Federated optimization: distributed machine learning for on-device intelligence*, arXiv:1610.02527, (2016).
[15] J. D. Lee, Q. Lin, T. Ma, and T. Yang, *Distributed stochastic variance reduced gradient methods and a lower bound for communication complexity*, arXiv:1507.07595, (2015).
[16] C. Ma, J. Konecny, M. Jaggi, V. Smith, M. I. Jordan, P. Richtarik, and M. Takac;, *Distributed optimization with arbitrary local solvers*, Optimization Methods Software, 32 (2017), pp. 813–848.
[17] C. Ma, V. Smith, M. Jaggi, M. Jordan, P. Richtarik, and M. Takac, *Adding vs. averaging in distributed primal-dual optimization*, in International Conference on Machine Learning, 2015, pp. 1973–1982.
[18] K. Mishchenko, F. Iutzeler, J. Malick, and M.-R. Amini, *A delay-tolerant proximal-gradient algorithm for distributed learning*, in Proceedings of the 35th international conference on machine learning (ICML), 2018.
[19] A. Mokhtari, M. Gürbüzbalaban, and A. Ribeiro, *Surpassing gradient descent provably: A cyclic incremental method with linear convergence rate*, arXiv:1611.00347, (2016).
[20] F. Pedregosa, R. Leblond, and S. Lacoste-Julien, *Breaking the nonsmooth barrier: A scalable parallel method for composite optimization*, Advances in Neural Information Processing System 30 (NIPS), (2017).
[21] Z. Peng, Y. Xu, M. Yan, and W. Yin, *Arock: an algorithmic framework for asynchronous parallel coordinate updates*, SIAM Journal on Scientific Computing, 38 (2016), pp. A2851–A2879.
[22] B. Recht, C. Re, S. Wright, and F. Niu, *Hogwild: A lock-free approach to parallelizing stochastic gradient descent*, in Advances in neural information processing systems, 2011, pp. 693–701.
[23] S. Shalev-Shwartz and T. Zhang, *Accelerated mini-batch stochastic dual coordinate ascent*, in Advances in Neural Information Processing Systems, 2013, pp. 378–385.
[24] R. Shokri and V. Shmatikov, *Privacy-preserving deep learning*, in Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, ACM, 2015, pp. 1310–1321.
[25] T. Sun, R. Hannah, and W. Yin, *Asynchronous coordinate descent under more realistic assumptions*, in Advances in Neural Information Processing Systems, 2017, pp. 6182–6190.
[26] M. Takáč, P. Richtárik, and N. Srebro, *Distributed mini-batch sdca*, arXiv:1507.08322, (2015).
[27] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society. Series B (Methodological), (1996), pp. 267–288.
[28] N. D. Vanli, M. Gurbuzbalaban, and A. Ozdaglar, *Global convergence rate of proximal incremental aggregated gradient methods*, arXiv:1608.01713, (2016).
[29] N. D. Vanli, M. Gurbuzbalaban, and A. Ozdaglar, *A stronger convergence result on the proximal incremental aggregated gradient method*, arXiv:1611.08022, (2016).
[30] R. Zhang and J. Kwok, *Asynchronous distributed admm for consensus optimization*, in International Conference on Machine Learning, 2014, pp. 1701–1709.

**Appendix A. Proof of convergence in the general case.** This appendix

completes the proof of Theorem 3.4 given in the main text to lift the unique minimizer assumption using an additional boundedness assumption on delays and inner loops.

Let $X^\star$ be the set of minimizers of (1), and fix $x^\star \in X^\star$. We are going to show the existence of another minimizer $x \in X$ having properties controlled with the two additional assumptions.

We use first the additional assumption that the number of inner loops is uniformly bounded by $p < \infty$. We define sequence $\mathbf{a'}^k$ by

$$\mathbf{a'}^k := \beta^{p-1} \left\|\overline{x}^k - \overline{x}^\star\right\|^2 + \left(1 - \beta^{p-1}\right) \max\left(\left\|\overline{x}^k - \overline{x}^\star\right\|^2, \left\|\overline{x}^k_{-i(k)} - \overline{x}^\star_{-i(k)}\right\|^2\right)$$

with $\beta := \min_i \pi_i$. Following (11) in the proof of Lemma 3.1, we still have the bound

$$\left\|x_i^k - x_i^\star\right\|^2 \leq \mathbf{a'}^{k-D_i^k}. \tag{24}$$

Furthermore, (12) and (13) imply that $\mathbf{a'}^k \leq \max_i \mathbf{a'}^{k-D_i^k}$.

We use now the additional assumption that $(D_i^k)$ are bounded by $D$. We introduce

$$\mathbf{e}^k := \max_{0 \leq d < D} \mathbf{a'}^{k+d}.$$

for a fixed $k > 0$, we write $k = mD + r$ with $m = \lfloor k/D \rfloor$ and $r = k - mD$. We can prove by induction (as in the proof of Theorem 3.2), that for any $r \in [0, D]$ that

$$\mathbf{e}_r^m := \max_{0 \leq d < D} \mathbf{a'}^{mD+r+d} \leq \mathbf{e}_r^{m-1}.$$

thus, have $\mathbf{e}_r^m \to \mathbf{b}_r$ for some $\mathbf{b}_r$. In addition, we have for any $r$ and $r' > r$ that $\mathbf{e}_{r'}^m \leq \max(\mathbf{e}_r^m, \mathbf{e}_r^{m+1})$ as the latter maximum covers the interval of the former (see Fig. 7) thus $\mathbf{b}_{r'} \leq \mathbf{b}_r$. Similarly, we have $\max(\mathbf{e}_{r'}^{m-1}, \mathbf{e}_{r'}^m) \geq \mathbf{e}_r^m$ which gives the reverse inequality; thus $\mathbf{b}_{r'} = \mathbf{b}_r = \mathbf{b}$.

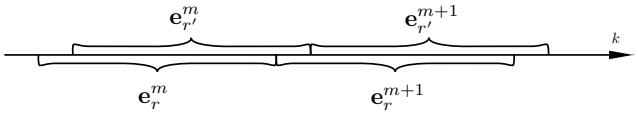



Fig. 7: Maxima over covering intervals of times

Thus we have that the sequence $(\mathbf{e}^k)$ is the union of $D$ sequences converging to $\mathbf{b}$ and thus converges itself to $b$. Moreover, using (24), we get that for any $i = 1, .., M$

$$\limsup_{k\to\infty} \left\|x_i^k - x_i^\star\right\|^2 \leq \mathbf{b},$$
$$\limsup_{k\to\infty} \left\|\overline{x}^k - \overline{x}^\star\right\|^2 \leq \mathbf{b}, \quad \text{and} \quad \limsup_{k\to\infty} \left\|\overline{x}^k_{-i(k)} - \overline{x}^\star_{-i(k)}\right\|^2 \leq \mathbf{b}.$$

This implies that the lim sup of the second term in $\mathbf{a'}^k$ is upper bounded by $\mathbf{b}$ and so is the maximum over $D$ consecutive times. Thus we have that for any $\varepsilon > 0$, there is

a $K$ such that for all $k > K$,

$$\begin{aligned} & & \mathbf{b} - \varepsilon &\leq (1-\beta^p) \max_{0\leq d<D} \left\|\overline{x}^{k+d} - \overline{x}^\star\right\|^2 + \beta^p(\mathbf{b}-\varepsilon) \\ & \text{thus} & \mathbf{b} - \frac{1+\beta^p}{1-\beta^p}\varepsilon &\leq \max_{0\leq d<D} \left\|\overline{x}^{k+d} - \overline{x}^\star\right\|^2 \leq \mathbf{b} + \varepsilon \\ (25) \qquad & \text{so} & \max_{0\leq d<D} \left\|\overline{x}^{k+d} - \overline{x}^\star\right\|^2 &\to \mathbf{b}. \end{aligned}$$

This convergence yields in turn that $\|\overline{x}^k - \overline{x}^\star\|^2 \to \mathbf{b}$; for better readability, we postpone the proof of this fact at the end of this section.

We have now all the ingredients to establish convergence of $(x^k)$ in the case of multiple minimizers. In the proof of Th. 3.4 for a unique minimizer (in Sec. 3.3.2 of the main text), the uniqueness of the minimizer is used only that the last steps. All the previous arguments could be repeated here to establish the existence of a subsequence of $(\overline{x}^k)$ converging to $\overline{x}$ with $x = \mathrm{prox}_{\gamma g}(\overline{x})$ being an optimal point. So let us pick this special optimal point, as $x^\star$ used in the above analysis. Since $\left\|\overline{x}^k - \overline{x}^\star\right\|^2 \to \mathbf{b}$, this limit can be only equal to 0, which directly implies that $x^k \to x^\star$, and ends the proof.

**Proof of the statement that $\|\overline{x}^k - \overline{x}^\star\|^2 \to \mathbf{b}$.**
We will establish the convergence by contradiction. Let $(n_m)$ be a diverging sequence such that $\|\overline{x}^{n_m} - x^\star\|^2 \leq \mathbf{b} - \varepsilon$ for some $\varepsilon > 0$. From (25) we have that there also exists a sequence $(l_m)$ such that $\left\|\overline{x}^{l_m} - x^\star\right\|^2 \to \mathbf{b}$ and $l_{m+1} - l_m \leq D' < \infty$. Thus, for any $\delta > 0$, there is $K < \infty$ such that for any $k > K, m > K$,

$$\left\|x_i^k - x_i^\star\right\|^2 \leq \mathbf{b} + \delta, \quad \left\|\overline{x}^{l_m} - x^\star\right\|^2 \geq \mathbf{b} - \delta, \quad \text{and} \quad \|\overline{x}^{n_m} - x^\star\|^2 \leq \mathbf{b} - \varepsilon.$$

For any moment $n = n_m$ and $l = l_m$ fulfilling $l_{m-1} < n_m \leq l_m$ and $m > K$, denote by $i$ the agent updating at time $n$. Let $u+1$ be the number of updates of $i$ between $n$ and $l$, and let $n = s_0 < s_1 < \cdots < s_u \leq l$ be the moments of these updates, we get for any $q = 1, .., u$ that

$$\begin{aligned} \|\overline{x}^{s_q} - \overline{x}^\star\|^2 &\leq \sum_{j=1}^M \pi_j \left\|x_j^{s_q} - x_j^\star\right\|^2 \leq \sum_{j\neq i}^M \pi_j \left\|x_j^{s_q} - x_j^\star\right\|^2 + \pi_i \mathbf{a}'^{s_{q-1}} \\ &\leq (1-\pi_i)(\mathbf{b}+\delta) + \pi_i\beta^{p-1}\|\overline{x}^{s_{q-1}} - \overline{x}^\star\|^2 + \pi_i\left(1-\beta^{p-1}\right)(\mathbf{b}+\delta) \\ &\leq (1-\phi)(\mathbf{b}+\delta) + \phi\|\overline{x}^{s_{q-1}} - \overline{x}^\star\|^2 \end{aligned}$$

with $\phi := \pi_i\beta^{p-1}$. Thus, by induction for $q = 1, .., u$,

$$\begin{aligned} \|\overline{x}^{s_u} - \overline{x}^\star\|^2 &\leq (1-\phi^u)(\mathbf{b}+\delta) + \phi^u\|\overline{x}^n - \overline{x}^\star\|^2 \\ &\leq (1-\phi^u)(\mathbf{b}+\delta) + \phi^u(\mathbf{b}-\varepsilon) = \mathbf{b} + \delta - \varepsilon\phi^u \end{aligned}$$

As $u < D'$, we obtain

$$\mathbf{b} - \delta \leq \left\|\overline{x}^l - x^\star\right\|^2 \leq \sum_{j\neq i}^M \pi_j \left\|x_j^l - x_j^\star\right\|^2 + \pi_i \mathbf{a}'^{s_u} \leq \mathbf{b} + \delta - \varepsilon\beta^{p(u+1)} \leq \mathbf{b} + \delta - \varepsilon\pi_i\phi^{D'}.$$

This yields $\delta \geq \varepsilon\pi_i\phi^{d'}/2 > 0$ which contradicts the arbitrariness of $\delta$, and then proves that $\left\|\overline{x}^k - \overline{x}^\star\right\|^2 \to \mathbf{b}$.

**Appendix B. Proof of the rate of convergence.** This appendix presents the proof of Theorem 3.5. We first introduce some notation and establish a key lemma.

Pick any $x^\star$ in the set of minimizers of $F$. We are going to bound the maximal sum of three terms $g^k$, $o^k$ and $r^k$ defined as follows as means of quantities over all the machines. For technical reasons, we also need to define $g^k_{-i}$, $o^k_{-i}$ and $r^k_{-i}$ for all $i$, as means of the same quantities without $i$-th summand. Specifically,

$$r^k := \sum_{i=1}^M \pi_i \left\|x_i^k - x_i^\star - (\overline{x}^k - \overline{x}^\star)\right\|^2, \qquad r^k_{-i} := \sum_{j\neq i} \pi_j \left\|x_j^k - x_j^\star - (\overline{x}^k - \overline{x}^\star)\right\|^2,$$

$$g^k := \sum_{i=1}^M \omega_i\pi_i \left\|\nabla f_i(z_i^k) - \nabla f_i(x^\star)\right\|^2, \quad g^k_{-i} := \sum_{j\neq i} \omega_j\pi_j \left\|\nabla f_j(z_j^k) - \nabla f_j(x^\star)\right\|^2,$$

$$o^k := \sum_{i=1}^M \pi_i \left\|z_i^k - w_i^k - (x^\star - \overline{x}^\star)\right\|^2, \qquad o^k_{-i} := \sum_{j\neq i} \pi_j \left\|z_j^k - w_j^k - (x^\star - \overline{x}^\star)\right\|^2,$$

$$s^k := \min_{k'\leq k} \left(r^{k'} + g^{k'} + o^{k'}\right), \qquad s^k_{-i} := \min_{k'<k} \left(r^{k'}_{-i} + g^{k'}_{-i} + o^{k'}_{-i}\right),$$

where $z_i^k$ and $w_i^k$ satisfy $x_i^k = z_i^k - \gamma_i\nabla f_i(z_i^k)$ and $z_i^k = \operatorname{prox}_{\gamma g}(w_i^k)$; $\omega_i = \gamma_i(2/L_i - \gamma_i)$. The quantity $s^k$ controls the decrease of the error in the algorithm, as formalized in Lemma B.1. The others quantities are involved in the following three useful inequalities. Using variance decomposition, we get that

$$\left\|\overline{x}^k - \overline{x}^\star\right\|^2 = \sum_{i=1}^M \pi_i \left\|x_i^k - x_i^\star\right\|^2 - r^k, \tag{26}$$
$$\left\|\overline{x}^k_{-i} - \overline{x}^\star_{-i}\right\|^2 = (1-\pi_i)^{-1}\sum_{j\neq i} \pi_j \left\|x_j^k - x_j^\star\right\|^2 - r^k_{-i}.$$

From the smoothness of the $(f_i)$, we have

$$\sum_{i=1}^M \pi_i \left\|x_i^k - x_i^\star\right\|^2 \leq \sum_{i=1}^M \pi_i \left\|z_i^k - x^\star\right\|^2 - g^k, \tag{27}$$
$$(1-\pi_i)^{-1}\sum_{j\neq i} \pi_j \left\|x_j^k - x_j^\star\right\|^2 \leq (1-\pi_i)^{-1}\sum_{j\neq i} \pi_j \left\|z_j^k - x^\star\right\|^2 - g^k_{-i}.$$

Finally, by (3), we also have

$$\sum_{i=1}^M \pi_i \left\|z_i^k - x^\star\right\|^2 \leq \sum_{i=1}^M \pi_i \mathbf{a}^{k-D_i^K} - o^k, \tag{28}$$
$$(1-\pi_i)^{-1}\sum_{j\neq i} \pi_j \left\|z_j^k - x^\star\right\|^2 \leq (1-\pi_i)^{-1}\sum_{j\neq i} \pi_j \mathbf{a}^{k-D_j^k} - o^k_{-i}.$$

LEMMA B.1. *For any $k \in [k_m, k_{m+1})$, we have*

$$\mathbf{b}^m \leq \mathbf{b}^{m-2} - s^k.$$

*with $\mathbf{a}^k$ defined by (8) and $\mathbf{b}^m = \max_{k\in[k_m,k_{m+1})} \mathbf{a}^k$ as in the proof of Theorem 3.4.*

*Proof.* Combining Eqs. (26), (27), and (28), we get for any $k \in [k_m, k_{m+1})$

$$\begin{aligned}\left\|\overline{x}^k - \overline{x}^\star\right\|^2 &= \sum_{i=1}^M \pi_i \left\|x_i^k - x_i^\star\right\|^2 - r^k \leq \sum_{i=1}^M \pi_i \left\|z_i^k - x^\star\right\|^2 - r^k - g^k \\ &\leq \sum_{i=1}^M \pi_i \mathbf{a}^{k-D_i^K} - r^k - g^k - o^k \leq \max_{k' \in [k_{m-1}, k)} \mathbf{a}^{k'} - s^k \leq \mathbf{b}^{m-1} - s^k \end{aligned} \tag{29}$$

where the last inequality comes from two facts: (i) for any $k' \in [k_m, k_{m+1})$, $\mathbf{a}^{k'} \leq \mathbf{b}^m$ by definition and (ii) $\mathbf{b}^m \leq \mathbf{b}^{m-1}$ (as shown in the proof of Theorem 3.4).

Similarly, if at moment $k$ the update is done by slave $i$, we have

$$\begin{aligned}\left\|\overline{x}_{-i}^k - \overline{x}_{-i}^\star\right\|^2 &\leq (1-\pi_i)^{-1} \sum_{j \neq i} \pi_j \mathbf{a}^{k-D_j^k} - g_{-i}^k - o_{-i}^k - r_{-i}^k \\ &\leq \max_{k' \in [k_{m-1}, k_m):\ d_i^{k'} \neq 0} \mathbf{a}^{k'} - s_{-i}^k, \end{aligned} \tag{30}$$

where $d_i^{k'} \neq 0$ comes from the fact that $k' = k - D_j^k$ was an update from a worker $j \neq i$ thus $d_i^k \neq 0$ (recall Fig. 1).

This can be wrapped up as

$$\mathbf{a}^k \leq \max\left(\mathbf{b}^{m-1} - s^k, \max_{k' \in [k_{m-1}, k_m):\ d_i^{k'} \neq 0} \mathbf{a}^{k'} - s_{-i}^k\right). \tag{31}$$

Denote by $j(k')$ the agent who is responsible for the update at moment $k'$. Then, plugging (31) into (30) yields

$$\begin{aligned}\left\|\overline{x}_{-i}^k - \overline{x}_{-i}^\star\right\|^2 \leq \max_{k' \in [k_{m-1}, k_m):\ d_i^{k'} \neq 0} \max\Big(&\mathbf{b}^{m-2} - s^{k'} - s_{-i}^k, \\ &\max_{k'' \in [k_{m-2}, k_{m-1}):\ d_{j(k')}^{k''} \neq 0} \left(\mathbf{a}^{k''} - s_{-j(k')}^{k'} - s_{-i}^k\right)\Big). \end{aligned} \tag{32}$$

By definition, $(s^k)$ and $(s_{-i}^k)$ are non-negative, non-increasing sequences; furthermore, for any $i$ and $j$ such that $i \neq j$ it holds that $s^k \leq \max\left(s_{-i}^k, s_{-j}^k\right)$. Thus, (32) can be recast as

$$\left\|\overline{x}_{-i}^k - \overline{x}_{-i}^\star\right\|^2 \leq \max\left(\mathbf{b}^{m-2} - s^k, \max_{k'' \in [k_{m-2}, k_{m-1}):\ d_{j(k')}^{k''} \neq 0} \mathbf{a}^{k''} - s^k\right) \leq \mathbf{b}^{m-2} - s^k$$

and finally, since $\mathbf{b}^{m-1} \leq \mathbf{b}^{m-2}$, we obtain

$$\mathbf{b}^m = \max_{k \in [k_m, k_{m+1})} \mathbf{a}^k \leq \max(\mathbf{b}^{m-1} - s^k, \mathbf{b}^{m-2} - s^k) = \mathbf{b}_{m-2} - s^k. \qquad \square$$

We are now in position to give the proof of Theorem 3.5, establishing the rate of convergence of our algorithm.

*Proof. (of Theorem 3.5)* Applying $m/2$ times Lemma B.1 and using that $(s^k)$ is non-increasing, we get

$$\mathbf{b}^m \leq \mathbf{b}^0 - \frac{m}{2} s^k,$$

We deduce

$$s^k \leq \frac{2(\mathbf{b}^0 - \mathbf{b}^m)}{m} \leq 2\frac{\max_{k' \in [k_0, k_1)} \mathbf{a}^{k'}}{m}.$$

Using that $\|\overline{x}^k - \overline{x}^\star\|^2 \leq \max_i \left\|x_i^k - x_i^\star\right\|^2$ and $\|\overline{x}_{-i}^k - \overline{x}_{-i}^\star\|^2 \leq \max_i \left\|x_i^k - x_i^\star\right\|^2$, we deduce from Lemma 3.1 that

$$\mathbf{a}^k \leq \max_i \left\|x_i^k - x_i^\star\right\|^2 \leq \max_{k' \leq k-1} \mathbf{a}^{k'} \leq \cdots \leq \mathbf{a}^0 \leq \max_i \left\|x_i^0 - x_i^\star\right\|^2$$

$$\text{and } s^k \leq \frac{2\max_i \left\|x_i^0 - x_i^\star\right\|^2}{m}.$$

On the other hand, we have that $x^k = \operatorname{prox}_{\gamma g}(\overline{x}^k)$ satisfies $\overline{x}^k - x^k \in \gamma \partial g(x^k)$ (see e.g. [3, Prop. 16.34]) We then introduce

$$h^k \coloneqq (\overline{x}^k - x^k)/\gamma + \nabla f(x^k) \in \partial F(x^k).$$

Writing $\overline{x}^k$ as $\sum_i \pi_i(z_i^k - \gamma_i \nabla f_i(z_i^k))$ and using each $f_i$'s smoothness, we have

$$\begin{aligned}
\|h^k\|^2 &= \left\|\gamma^{-1}\sum_{i=1}^M \pi_i\left(z_i^k - x^k\right) - \sum_{i=1}^M \pi_i\left(\nabla f_i(z_i^k) - \nabla f_i(x^k)\right)\right\|^2 \\
&\leq \frac{1}{\gamma^2}\sum_{i=1}^M \pi_i \left\|\left(z_i^k - x^k\right) - \gamma_i\left(\nabla f_i(z_i^k) - \nabla f_i(x^k)\right)\right\|^2 \\
&\leq \frac{1}{\gamma^2}\sum_{i=1}^M \pi_i \left(\left\|z_i^k - x^k\right\|^2 - \omega_i \left\|\nabla f_i(z_i^k) - \nabla f_i(x^k)\right\|^2\right) \\
&\leq \frac{1}{\gamma^2 \sum_i \gamma_i^{-1}}\sum_{i=1}^M \gamma_i^{-1}\left\|z_i^k - x^k\right\|^2. \qquad (33)
\end{aligned}$$

Then, as $\|a+b+c\|^2 \leq \left(1+\frac{\delta}{2}\right)\|a+b\|^2 + \left(1+\left(\frac{\delta}{2}\right)^{-1}\right)\|c\|^2 \leq (2+\delta)\left(\|a\|^2 + \|b\|^2\right)$ $+ \left(1+2\delta^{-1}\right)\|c\|^2$ for any $\delta > 0$, we can bound each summand with individual $\delta_i$:

$$\begin{aligned}
&\|x^k - z_i^k\|^2 = \left\|\operatorname{prox}_{\gamma g}(\overline{x}^k) - \operatorname{prox}_{\gamma g}(w_i^k)\right\|^2 \leq \|\overline{x}^k - w_i^k\|^2 \\
&= \left\|\left(\overline{x}^k - \overline{x}^\star + x_i^\star - x_i^k\right) - \left(w_i^k - \overline{x}^\star + x_i^\star - x_i^k\right)\right\|^2 \\
&= \left\|-\left(x_i^k - x_i^\star - (\overline{x}^k - \overline{x}^\star)\right) + \left(z_i^k - w_i^k - (x^\star - \overline{x}^\star)\right) - \gamma_i\left(\nabla f_i(z_i^k) - \nabla f_i(x^\star)\right)\right\|^2 \\
&\leq (2+\delta_i)\left(\left\|x_i^k - x_i^\star - (\overline{x}^k - \overline{x}^\star)\right\|^2 + \left\|z_i^k - w_i^k - (x^\star - \overline{x}^\star)\right\|^2\right) \\
&\quad + \left(1+2\delta_i^{-1}\right)\gamma_i^2\left\|\nabla f_i(z_i^k) - \nabla f_i(x^\star)\right\|^2.
\end{aligned}$$

$$\text{Hence,} \quad \sum_{i=1}^M \pi_i \|x^k - z_i^k\|^2 \leq \max_i\left[(2+\delta_i)\left(r^k + o^k\right) + \left(1+2\delta_i^{-1}\right)\frac{\gamma_i^2}{\omega_i} g^k\right].$$

Thus (33) gives

$$\begin{aligned}
\min_{k' \leq k} \|h^{k'}\|^2 &\leq \gamma^{-2} \min_{k' \leq k} \max_i \left((2+\delta_i)\left(r^{k'} + o^{k'}\right) + \left(1+2\delta_i^{-1}\right)\frac{\gamma^2}{\omega} g^{k'}\right) \\
&\leq \gamma^{-2} \max_i \max\left(2+\delta_i, \gamma_i^2 \omega_i^{-1}\left(1+2\delta_i^{-1}\right)\right) s^k.
\end{aligned}$$

Taking $\delta_i = \frac{\gamma_i}{2/L_i - \gamma_i}$ yields

$$\min_{k' \leq k} \|h^{k'}\|^2 \leq \gamma^{-2} \max_i (2+\delta_i)\, s^k \leq \gamma^{-2} \max_i \left(2 + \frac{\gamma_i}{2/L_i - \gamma_i}\right) \frac{2 \max_j \left\|x_i^0 - x_i^\star\right\|^2}{m}$$
$$\leq \frac{8 \max_j \left\|x_j^0 - x_j^\star\right\|^2}{m\gamma^2 \min_i (2 - \gamma_i L_i)}.$$

where at the last step we used our assumption $\gamma_i \in (0, 2/L_i)$. ∎

**Appendix C. Proof of epoch scaling with delays.** This appendix gives the proof of the results of Proposition 3.6 and the following table.

*Case of delays uniformly bounded by d.* By definition of time, we have $d \geq M$, and then $d = M + \tau$ with $\tau \geq 0$. It is easy to see on the definition of the epoch sequence of Section 3.1 that $k_{m+1} - k_m \leq 2d+1$ as $d_i^{k_{m+1}} \leq d$ for all $i$. Then there was a least one update of each machine in $[k_{m+1} - d; k_{k+1}]$. Repeating this reasoning at $k_{m+1} - d - 1$, one gets that two update occured in $[k_{m+1} - 2d - 1, k_{m+1}]$ hence the result.

*Case of average delay bounded by $\overline{d}$.* To prove that $\overline{d} = \frac{M-1}{2} + \tau$ with $\tau \geq 0$, one can notice that at any time $k$ there can be only one worker with a zero delay (the updating one), only one with a delay equal to 1, and so on. Consequently, the sum of the delays is at least $M(M-1)/2$ thus the average is at least $(M-1)/2$.

We now look carefully at the epoch sequence. To simplify notation, we introduce $N := k_{m+1} - k_m$ and $i := i(k_{m+1})$ the machine updating at moment $k_{m+1}$. We will consider two subcases depending on which worker performed the update at $k_m$:

- *When $i(k_m) = i$.* In this case, there cannot be any other update of $i$ between $k_m$ and $k_{m+1}$. Indeed, by definition of $k_{m+1}$ it is the first moment when every machine has been updated at least twice since moment $k_m$, so for $i$ it has to be the second time (including $k_m$). Therefore,

$$\sum_{k=k_m}^{k_{m+1}-1} d_i^k = 0 + 1 + \cdots + (N-1) = \frac{N(N-1)}{2}.$$

- *When $i(k_m) \neq i$.* In this case, there is a moment $\widetilde{k} \in (k_m, k_{m+1})$ such that $i(\widetilde{k}) = i$. Since $d_i^{k_m} \geq 1$ and for any two numbers $a, b$ we have $a^2 + b^2 \geq \frac{(a+b)^2}{2}$,

$$\begin{aligned}
\sum_{k=k_m}^{k_{m+1}-1} d_i^k &= 1 + \cdots + (\widetilde{k} - k_m) + 0 + 1 + \cdots + (k_{m+1} - \widetilde{k} - 1) \\
&= \frac{(\widetilde{k} - k_m)(\widetilde{k} - k_m + 1)}{2} + \frac{(k_{m+1} - \widetilde{k} - 1)(k_{m+1} - \widetilde{k})}{2} \\
&= \frac{1}{2}\left(\left(\widetilde{k} - k_m + \frac{1}{2}\right)^2 + \left(k_{m+1} - \widetilde{k} - \frac{1}{2}\right)^2 - \frac{1}{2}\right) \\
&\geq \frac{(k_{m+1} - k_m)^2 - 1}{4} = \frac{N^2 - 1}{4} \geq \frac{N(N-1)}{4}.
\end{aligned}$$

In both cases, we have

$$\sum_{k=k_m}^{k_{m+1}-1} d_i^k \geq \frac{N(N-1)}{4}. \tag{34}$$

In addition, for any moment $k_m + l$ ($l \geq 0$); among workers $j \neq i$, at least $M-2$ have a delay greater than 0, at least $M-3$ have a delay greater than 1, etc.

$$\sum_{j \neq i} d_j^{k_m+l} \geq \sum_{q=0}^{M-2} q = \frac{(M-1)(M-2)}{2}. \tag{35}$$

Summing (34) and (35) over $l = 0, \ldots, N-1$ we obtain

$$\sum_{k=k_m}^{k_{m+1}-1} \sum_{j=1}^{M} d_j^k \geq N \frac{(M-1)(M-2)}{2} + \frac{N(N-1)}{4}.$$

Combining this with the fact that $\sum_{k=k_m}^{k_{m+1}-1} \sum_{j=1}^{M} d_j^k \leq MN\bar{d}$ for the average bound leads to the result.