DELAY-TOLERANT DISTRIBUTED BREGMAN PROXIMAL ALGORITHMS

S. CHRAIBI, F. IUTZELER, J. MALICK, AND A. ROGOZIN

Abstract. Many problems in machine learning write as the minimization of a sum of individual loss functions over the training examples. These functions are usually differentiable but, in some cases, their gradients are not Lipschitz continuous, which compromises the use of (proximal) gradient algorithms. Fortunately, changing the geometry and using Bregman divergences can alleviate this issue in several applications, such as for Poisson linear inverse problems. However, the Bregman operation makes the aggregation of several points and gradients more involved, hindering the distribution of computations for such problems. In this paper, we propose an asynchronous variant of the Bregman proximal-gradient method, able to adapt to any centralized computing system. In particular, we prove that the algorithm copes with arbitrarily long delays and we illustrate its behavior on distributed Poisson inverse problems.

Distributed optimization; Asynchronous methods; Proximal algorithms; Bregman divergence; Poisson inverse problems

1 Introduction

1.1 Context: distributed problems and lack of smoothness

Many problems in machine learning and signal processing involve the minimization of a sum of functions measuring the loss between the model and the data points. This sum form is highly practical when the data is distributed over several machines: the total loss is simply the sum of the losses over each machine's local data, and, similarly, the gradient of the total loss is the sum of the machines' local gradients. Thus, a central machine can minimize the total loss using a first-order method by simply querying the machines for their local gradients. Such a method is called *synchronous* since, at each iteration, the central machine waits for all the machines to respond before proceeding to a new query. Unfortunately, the time a machine takes to respond can vary a lot due to network/communication issues, uneven data, or heterogeneous computing power. This calls for *asynchronous* methods where the central machine updates the global model and queries a new gradient, as soon as a machine responds. Despite their more involved analysis, asynchronous optimization methods have become popular thanks to their practical efficiency. Indeed, having asynchronous exchanges is crucial in practice since then much more iterations can be performed in the same time interval compared to synchronous setups, see eg. the discussion in [15].

(S. Chraibi) Univ. Grenoble Alpes, LJK, 38000 Grenoble, France
(F. Iutzeler) Univ. Grenoble Alpes, CNRS, Grenoble INP, LJK, 38000 Grenoble, France
(J. Malick) Univ. Grenoble Alpes, CNRS, Grenoble INP, LJK, 38000 Grenoble, France
(A. Rogozin) Moscow Institute of Physics and Technology, Moscow, Russia
E-mail addresses: selim.chraibi@univ-grenoble-alpes.fr.

Most existing analyses of asynchronous first-order methods assume that the functions are *smooth* (ie. that their gradients are Lipschitz continuous). While such an assumption often holds, several objectives of interest do not satisfy it, even though they are differentiable; it is the case, for example, for recovery from quadratic measurements [11], and Poisson inverse problems [8]. This lack of smoothness breaks down the usual "descent lemmas" at the core of the analysis of first-order methods; see eg. [7, Chap. 10]. Fortunately, for several problems of interest, including the two mentioned above, changing the geometry can alleviate the issue of lack of smoothness. The idea is to use, instead of Euclidean geometry, the geometry induced by so-called *Bregman divergences* [12, 5]. Indeed, a descent/contraction lemma can be obtained for functions that are smooth with respect to a Bregman divergence [4], such as in the two examples above. This opens the way for the application and analysis of first-order methods, as we recall in Section 2.2. Our work in this paper can be included in this line of research: we develop and analyze a *distributed version* of the Bregman proximal method of [4].

1.2 Asynchronous Bregman proximal-gradient in centralized set-up

We consider a centralized setup where a central machine coordinates asynchronously M worker machines to solve an optimization problem of the form

$$\min_{x\in\mathcal{X}} \; F(x) := \frac{1}{M}\sum_{i=1}^{M} f_i(x) + g(x). \tag{$\mathcal{P}$}$$

We make no special assumption on the underlying system which can be completely heterogeneous. In the optimization problem, each function f_i is local to machine i, and the central machine shares the search space $\mathcal{X} \subset \mathbb{R}^n$ and a regularization function g. We focus on the case where g is convex and lower semi-continuous (*lsc*), and that $f_i : \mathcal{X} \to \mathbb{R}$ is convex and differentiable – but not necessarily smooth (ie. we do *not* assume that the gradients are Lipschitz continuous on $\mathcal{X}$).

The lack of smoothness ruins the theoretical properties of existing asynchronous distributed proximal-gradient algorithms (see e.g., [18, 23] and references therein) in the same way that it ruins the convergence of standard (non-distributed) proximal gradient methods (which has led to the works of [4] and [17]). In particular, [4] proposes a Bregman proximal-gradient method that fits our assumptions. However, an asynchronous distributed version of this method still has to be designed and analyzed, and this is what we propose in this paper.

More specifically, we use the tools developed in [4] to extend the asynchronous proximal-gradient algorithm of [18] when the local functions f_i are not smooth in the Euclidean geometry, but rather in an adapted Bregman geometry. The resulting Bregman proximal-gradient algorithm copes with flexible asynchronous communications, and we analyze its convergence under mild assumptions on both the functions and the computing setup. First, we rely on the same set of assumptions as in [4] on the functions which enable us to control the *Bregman divergence* between iterates and the solution. This change of metric (and in particular the loss of symmetry) introduces technical challenges compared to [18, 23] which rely on controlling the *Euclidean distance*. Second, we pay special attention to having realistic assumptions on communication delays to cover a diversity of the scenarios included in this framework (e.g. computing clusters, mobile devices): we do *not* assume that the delays between communications are bounded and we only assume that all workers eventually communicate with the central machine, as per the definition of *totally asynchronous* methods by Bertsekas and Tsitsiklis' classification [9, Chap. 6.1]. To the best of our knowledge, most papers on asynchronous distributed methods rely on a bounded delay assumption; exceptions include [22, 15, 16, 19, 18] but they rely on the smoothness of the objective that we do not have here.

The outline of the paper is as follows. In Section 2, we introduce the notation and recall the ideas of, both, Bregman smoothness [4] and standard asynchronous proximal-gradient algorithms [23, 18]. In Section 3, we present our asynchronous Bregman algorithm, by adapting the developments of the Euclidean setting. Then we analyze in Section 4 the well-posedness and the convergence of our algorithm. Notably, we prove convergence for a fixed stepsize, independent of the computing system (in particular independent of delays in the system). Finally, we provide, in Section 5, numerical illustrations of the behavior of the algorithm on distributed Poisson inverse problems.

2 Notation and recalls on proximal-gradient and Bregman geometry

This section introduces the notation used in this paper and recalls the main notions and ideas. Section 2.1 presents the natural splitting of the proximal-gradient method that serves as a basis for the developments of the next section. It also introduces important notation about delays that will be constantly used in the sequel. Section 2.2 presents how Bregman smoothness can replace the usual smoothness as a fundamental tool for convergence analysis.

2.1 From proximal-gradient to asynchronous optimization

A natural rationale to distribute algorithm is to first endow each worker with a copy x_i of the global variable and impose a consensus through the indicator function $\iota_C : \mathcal{X}^M \to \overline{\mathbb{R}}$ defined as

$$\iota_C(x_1, \dots, x_M) = 0 \text{ if } x_i = x_j \text{ for all } i, j \text{ and } +\infty \text{ elsewhere.} \tag{1}$$

Mathematically, we end up with the following problem equivalent [1] to (42)

$$\min_{(x_1,\dots,x_M)\in\mathcal{X}^M} \frac{1}{M}\sum_{i=1}^{M} f_i(x_i) + g(x_i) + \iota_C(x_1, \dots, x_M) \tag{2}$$

in which the first term is differentiable while the two others are convex and lower semi-continuous.

Such a problem naturally calls for proximal gradient methods (where the differentiable part of the objective is iteratively replaced by a quadratic model). Specifically, for a given step size γ

$$\begin{aligned}(x_1^k, \dots, x_M^k) = \operatorname*{argmin}_{(x_1,\dots,x_M)\in\mathcal{X}^M} \Bigg\{ \frac{1}{M}\sum_{i=1}^{M} \Big(f_i(x_i^{k-1}) + \langle x_i - x_i^{k-1}; \nabla f_i(x_i^{k-1})\rangle + \dots \\ \dots + \frac{1}{2\gamma}\left\|x_i - x_i^{k-1}\right\|^2 + g(x_i)\Big) + \iota_C(x_1, \dots, x_M) \Bigg\}\end{aligned} \tag{3}$$

[1] Equivalent here means that $x^\star$ is a solution of $(\mathcal{P})$ if and only if $(x^\star, \dots, x^\star)$ is a solution of (2).

where the particular form of ι_C immediately leads to $x_1^k = \dots = x_M^k$. Thus we have, for all $i \in 1, \dots, M$,

$$x_i^k = \operatorname*{argmin}_{x \in \mathcal{X}} \left\{ g(x) + \frac{1}{M} \sum_{i=1}^{M} \left(f_i(x_i^{k-1}) + \langle x - x_i^{k-1}; \nabla f_i(x_i^{k-1}) \rangle + \frac{1}{2\gamma} \left\| x - x_i^{k-1} \right\|^2 \right) \right\} \tag{4}$$

$$= \operatorname*{argmin}_{x \in \mathcal{X}} \left\{ g(x) + \left\| x - \frac{1}{M} \sum_{i=1}^{M} \left(x_i^{k-1} - \gamma \nabla f_i(x_i^{k-1}) \right) \right\|^2 \right\}. \tag{5}$$

In terms of distributed optimization, this means that the central machine has to gather the gradient steps of all the workers $(x_i^{k-1} - \gamma \nabla f_i(x_i^{k-1}))$, average them, and perform a proximal operation for g on $\mathcal{X}$ to get the next iterate, then send the result to all workers. This algorithm is by construction completely synchronous.

Let us now derive an asynchronous version of this algorithm. To do so, we need some notation, introduced below and illustrated in Figure 1. First, we call iteration (or time) k the moment of the k-th exchange between a worker and the central machine. Second, we denote by d_i^k the delay suffered by worker i at time k, defined by the number of iterations since worker i last exchanged with the central machine. We also define the *second-order* delay for worker i and time k by

$$D_i^k = d_i^k + d_i^{k - d_i^k - 1} + 1. \tag{6}$$

These two delays allow us to handle, at each time k, the worker i's gradient previously received by the central machine at the last exchange $k - d_i^k$, which was itself computed at a point of the *second* last exchange $x^{k-D_i^k}$.

To get an asynchronous variant of (5), the iteration has to be modified as follows. Regarding the use of gradient, an asynchronous version has to replace $\nabla f_i(x^{k-1})$ by $\nabla f_i(x^{k-D_i^k})$ in the iteration. Regarding the base point, there are two possibilities:

- either it is kept at the last iterate x^{k-1}, which leads to the update of the proximal incremental aggregated gradient (PIAG) method [2, 23]:

$$x^k = \operatorname*{argmin}_{x \in \mathcal{X}} \left\{ g(x) + \left\| x - \frac{1}{M} \sum_{i=1}^{M} \left(x^{k-1} - \gamma \nabla f_i(x^{k-D_i^k}) \right) \right\|^2 \right\}; \tag{7}$$

- or it is set to the point corresponding to the gradient computation $x^{k-D_i^k}$, which leads to the update of DAve-PG [19, 18]:

$$x^k = \operatorname*{argmin}_{x \in \mathcal{X}} \left\{ g(x) + \left\| x - \frac{1}{M} \sum_{i=1}^{M} \left(x^{k-D_i^k} - \gamma \nabla f_i(x^{k-D_i^k}) \right) \right\|^2 \right\}. \tag{8}$$

For our analysis, we focus here on the second option which is often faster in the Euclidean case, as shown in [18, Sections 2.4 and 3.4]). We will come back to this distinction in the numerical illustrations.

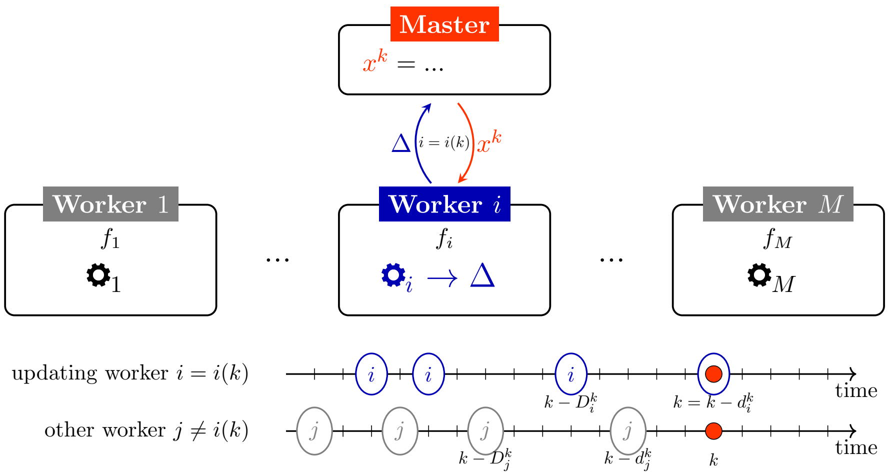

Figure 1. Asynchronous distributed setting and notation. Top: As soon as a worker finishes its computation, it sends its updates to the master, gets a new query point, and proceeds to the next computation. Bottom about delays notation: d_i^k is the delay suffered by worker i at time k defined as the number of iterations since worker i last exchanged with the master (the exchanging worker $i = i(k)$ has no delay); the second-order delay is D_i^k corresponds to the second-last exchange.

2.2 Descent with and without smoothness

Smoothness is a central property to obtain functional descent and contraction in first-order methods [20, 7]. Let a function $f : \mathbb{R}^n \to \mathbb{R}$ be convex and differentiable. If f is furthermore L-smooth, which means

$$\nabla f \text{ is } L\text{-Lipschitz continuous or, equivalently, } L\|\cdot\|_2^2/2 - f \text{ is convex,} \tag{9}$$

then we have that (see eg. [7, Lem. 5.])

$$f(y) \leq f(x) + \langle y - x; \nabla f(x)\rangle + \frac{L}{2}\|y - x\|_2^2 \quad \text{for all } x, y \in \mathbb{R}^n \tag{10}$$

Thus the gradient step, defined as

$$\begin{aligned} p_\gamma(x) &= \operatorname*{argmin}_y \left\{ f(x) + \langle y - x; \nabla f(x)\rangle + \frac{1}{2\gamma}\|y - x\|_2^2 \right\}, & (11)\\ &= x - \gamma \nabla f(x), & (12) \end{aligned}$$

grants both descent and contraction (see eg. [6, Th. 18.15]): for all $x \in \mathbb{R}^n$ and for a minimizer $x^\star$ of f, we have

$$f(p_\gamma(x)) \leq f(x) - \gamma\left(1 - \frac{L\gamma}{2}\right)\|\nabla f(x)\|_2^2, \quad \text{and} \tag{13}$$

$$\|p_\gamma(x) - x^\star\|_2^2 \leq \|x - x^\star\|_2^2 - \gamma\left(\frac{2}{L} - \gamma\right)\|\nabla f(x)\|_2^2. \tag{14}$$

When smoothness is not present, these inequalities do not hold anymore, which hinders the analysis of gradient methods. Fortunately, for a function $f\colon \mathcal{X} \to \mathbb{R}$, convex and differentiable on the interior of $\mathcal{X}$, some smoothness can be obtained by comparing it to a Bregman function; see eg. [21, Ch. 26].

Assumption 1 (Bregman regularizer)**.** *The function $h\colon \mathbb{R}^n \to \mathbb{R}$ satisfies the following conditions: h is proper, lower semi-continuous, and convex;* $\operatorname{dom} h = \mathcal{X}$*; h is of Legendre type*[2]*.*

Under this assumption, we have that $\operatorname{dom} \partial h = \operatorname{int} \operatorname{dom} h$ and that for all $x \in \operatorname{int} \operatorname{dom} h$, $\partial h(x) = \{\nabla h(x)\}$. Building on this function h, we can define the Bregman distance [12]

$$D_h(x, y) = h(x) - h(y) - \langle \nabla h(y), x - y\rangle \tag{15}$$

for any $x \in \operatorname{dom} h$ and any $y \in \operatorname{int} \operatorname{dom} h$. We also define Bregman gradient steps [1] for $x \in \operatorname{int} \operatorname{dom} h$, similarly as (11), by

$$\begin{aligned} p_\gamma(x) &= \operatorname*{argmin}_{y \in \operatorname{dom} h} \left\{ f(x) + \langle y - x; \nabla f(x)\rangle + \frac{1}{\gamma} D_h(y, x) \right\} && (16)\\ &= \operatorname*{argmin}_{y \in \operatorname{dom} h} \left\{ h(y) + \langle y; \gamma \nabla f(x) - \nabla h(x)\rangle \right\}. && (17) \end{aligned}$$

The simple yet powerful idea of [4] is then to compare f to h in order to extend smoothness (9) beyond the Euclidean case: if there is an $L > 0$ such that $Lh - f$ is convex on $\operatorname{int} \operatorname{dom} h$, then [4, Lem. 1]

$$f(y) \le f(x) + \langle y - x; \nabla f(x)\rangle + L D_h(y, x) \quad \text{for all } x, y \in \operatorname{int} \operatorname{dom} h. \tag{18}$$

Therefore, counterparts of (14) hold

$$\begin{aligned} f(p_\gamma(x)) &\le f(x) - \frac{1}{\gamma} D_h(x, p_\gamma(x)) - (1 - \gamma L) D_h(p_\gamma(x), x), \text{ and} && (19)\\ D_h(x^\star, p_\gamma(x)) &\le D_h(x^\star, x) - (1 - \gamma L) D_h(p_\gamma(x), x) && (20) \end{aligned}$$

for all $x \in \operatorname{int} \operatorname{dom} h$ and $x^\star$ a solution of $\min_{\mathcal{X}} f$,[3] see eg. [4, Lem. 5].

Example 1 (Nonnegative linear regression)**.** *Let $A \in \mathbb{R}_+^{m \times n}$ be matrix with non-null rows $(a_i)_{i \in 1, \ldots, m}$, $b \in \mathbb{R}_{++}^m$ a positive output vector. Nonnegative linear regression minimizes the generalized Kullback-Leibler divergence between a linear model Ax, with $x \in \mathcal{X} = \mathbb{R}_+^n$, and the output b (see eg.* [4, Sec 5.3]*).*

$$\min_{x \in \mathbb{R}_+^n} f(x) := KL(Ax, b) \tag{21}$$

Where for $v, u \in \mathbb{R}_+^m$, $KL(v, u) = \sum_{i=1}^m v_i \log v_i / u_i - v_i + u_i$ and $KL(0, u) = \|u\|_1$, by continuous extension. We see that f is differentiable on $\operatorname{int} \mathcal{X}$*, but it is not smooth. It is however smooth relative to the Boltzmann-Shannon entropy ($h(x) = \sum_{j=1}^n x_j \log x_j$ with* $\operatorname{dom} h = \mathcal{X}$*), since by* [4, Lem. 8]*, $Lh - f$ is convex on* $\operatorname{int} \operatorname{dom} h$ *with*

$$L = \max_j \sum_{i=1}^m a_{ij} \ . \tag{22}$$

[2]We recall that a Legendre function is differentiable and strictly convex on $\operatorname{int} \operatorname{dom} h$ with $\|\nabla h(x^k)\| \to +\infty$ for (x^k) of $\operatorname{int} \operatorname{dom} h$ converging to a boundary point of $\operatorname{dom} h$.

[3]Here, we use explicitly that $\operatorname{dom} h = \mathcal{X}$.

Since $\nabla f(x) = \sum_{i=1}^m a_i \log(\langle a_i, x\rangle / b_i)$ *and* $\nabla h(x) = 1 + \log(x)$*, the Bregman gradient operator* $p_\gamma(x) = \operatorname{argmin}_{y\in\operatorname{dom} h}\{h(y) + \langle y; \gamma\nabla f(x) - \nabla h(x)\rangle\}$ *can be computed coordinate-wise and we obtain that for any* $x \in \operatorname{int}\operatorname{dom} h$*,*

$$[p_\gamma(x)]_j = \operatorname*{argmin}_{y\geq 0}\left\{ y\log y + y\left(\gamma\sum_{i=1}^m a_{ij}\log\left(\frac{\langle a_i, x\rangle}{b_i}\right) - 1 - \log(x_j)\right)\right\} \tag{23}$$

$$= x_j \exp\left(-\gamma\sum_{i=1}^m a_{ij}\log\left(\frac{\langle a_i, x\rangle}{b_i}\right)\right) = \frac{x_j}{\prod_{i=1}^m\left(\frac{\langle a_i, x\rangle}{b_i}\right)^{\gamma\, a_{ij}}}. \tag{24}$$

Hence, $p_\gamma(x) \in \operatorname{int}\operatorname{dom} h$ *and brings functional descent and contraction from* (20) *with* $\gamma \leq 1/L$ *given by* (22)*. Note that the objective admits a unique minimizer on* $\mathcal{X}$ *when* A *is full-rank. In this case indeed, observe first that* f *is coercive which proves the existence of a minimizer on* $\mathcal{X}$*. Second, considering two minimizers* x_1, x_2 *of* f *on* $\mathbb{R}^n_+$*, the strict convexity of* $v \mapsto KL(v, u)$ *implies that* $x_1 - x_2 \in \ker A = \{0\}$*, which shows the uniqueness.* ◀

3 Asynchronous Bregman proximal gradient

3.1 Assumptions

In the framework introduced in the previous section, we present our algorithm for solving (42), under the following assumptions. First, we need to assume that the problem has minimizers, which can be done by constraining $\mathcal{X}$ to be closed and convex and g to be lower semi-continuous, and proper.

Assumption 2 (On the problem)**.** *The following conditions hold:*

i. *the constraint set* $\mathcal{X} \subseteq \mathbb{R}^n$ *is closed*[4] *and convex with a non-empty interior;*
ii. *the function* g *is proper, lsc, convex, with* $\operatorname{dom} g \cap \operatorname{int}\mathcal{X} \neq \emptyset$ *;*
iii. *the minimizers of* ($\mathcal{P}$) *form a non-empty compact set in* $\mathcal{X}$*.*

We also formalize our assumption that the local functions f_i of the problem are smooth with respect to the Bregman divergence generated by h in the sense of [4].

Assumption 3 (On the functions (f_i))**.** *For every* $i = 1, \ldots, m$*, we assume that the* $\mathcal{X} \to \mathbb{R}$ *function* f_i *verifies:*

i. f_i *is proper, lower semi-continuous, and convex;*
ii. f_i *is differentiable on* $\operatorname{int}\operatorname{dom} h$*;*
iii. f_i *is* L*-smooth with respect to* h*, ie.* $Lh - f_i$ *is convex on* $\operatorname{int}\operatorname{dom} h$*.*

As we will see in Section 4.1, this set of assumptions ensures that the Bregman steps at the core of our development are well-defined.

[4]The closedness, combined with Assumption 1, implies that $\operatorname{dom} h$ is closed, which is not always assumed in the case of optimization with Bregman divergences. In our case, it will be helpful to characterize pointwise convergence (as in eg. [4, Th. 2 ii]) since functional decrease is out of reach in our case (see [18]).

3.2 From Bregman proximal gradient to asynchronous optimization

We now take another look at problem (2) by considering a Bregman proximal gradient method. We can perform steps of the form

$$(x_1^k, \dots, x_M^k) = \operatorname*{argmin}_{(x_1,\dots,x_M)\in\mathcal{X}^M} \left\{ \frac{1}{M}\sum_{i=1}^M \Big(f_i(x_i^{k-1}) + \langle x_i - x_i^{k-1}; \nabla f_i(x_i^{k-1})\rangle + \dots \right.$$
$$\left. \dots + \frac{1}{\gamma} D_h(x_i, x_i^{k-1}) + g(x_i) \Big) + \iota_C(x_1,\dots,x_M) \right\} \quad (25)$$

where the particular form of ι_C immediately leads to $x_1^k = \dots = x_M^k$, and therefore, for all $i \in 1,\dots,M$,

$$x_i^k = \operatorname*{argmin}_{x\in\mathcal{X}} \left\{ g(x) + \frac{1}{M}\sum_{i=1}^M \Big(f_i(x_i^{k-1}) + \langle x - x_i^{k-1}; \nabla f_i(x_i^{k-1})\rangle + \frac{1}{\gamma} D_h(x, x_i^{k-1}) \Big) \right\} \quad (26)$$

$$= \operatorname*{argmin}_{x\in\mathcal{X}} \left\{ \gamma g(x) + h(x) + \left\langle x; \frac{1}{M}\sum_{i=1}^M \big(\gamma \nabla f_i(x_i^{k-1}) - \nabla h(x^{k-1})\big) \right\rangle \right\}. \quad (27)$$

We obtain an asynchronous variant of this iteration by following the same steps as in section 2.2. First we replace $\nabla f_i(x^{k-1})$ by $\nabla f_i(x^{k-D_i^k})$ for each worker i. Second, we have two choices for replacing $\nabla h(x^{k-1})$:

- either it is kept at the last iterate x^{k-1}, which leads to the Bregman version of the PIAG method, which appears in [2, Sec. V]:

$$x^k = \operatorname*{argmin}_{x\in\mathcal{X}} \left\{ \gamma g(x) + h(x) + \left\langle x; \frac{1}{M}\sum_{i=1}^M \Big(\gamma \nabla f_i(x_i^{k-D_i^k}) - \nabla h(x^{k-1})\Big) \right\rangle \right\} \quad (28)$$

- or, as prescribed here, we set it to the point corresponding to the gradient computation $x^{k-D_i^k}$, which leads to:

$$x^k = \operatorname*{argmin}_{x\in\mathcal{X}} \left\{ \gamma g(x) + h(x) + \left\langle x; \frac{1}{M}\sum_{i=1}^M \Big(\gamma \nabla f_i(x_i^{k-D_i^k}) - \nabla h(x^{k-D_i^k})\Big) \right\rangle \right\}. \quad (29)$$

We consider here the second option, inspired by the better results of [18] for the Euclidean case. Note also that the theory developed in [2] for the first option is based on the strong assumption that the divergence D_h should be lower and upper bounded by the squared Euclidean distance, which is not the case in Example 1 for example.

3.3 Algorithm & Practical implementation

We are now ready to state our asynchronous Bregman proximal-gradient algorithm. First, the central machine initializes $x^0 \in \operatorname{int}\operatorname{dom} h$ and sends it to all workers. Then, the central machine keeps track of the quantity

$$\overline{u}^k := \frac{1}{M}\sum_{i=1}^M \underbrace{\gamma \nabla f_i(x^{k-D_i^k}) - \nabla h(x^{k-D_i^k})}_{:=u_i^k} \quad (30)$$

where u_i^k corresponds to the last contribution of agent i, received at time $k - d_i^k$ (hence $u_i^k = u_i^{k-1} = \cdots = u_i^{k-d_i^k}$), which results from its local computation from point $x^{k-D_i^k}$.

A key feature of our algorithm emerges here: regardless of each worker's response delay D_i^k, their contribution to the central node's aggregate $\overline{u}^k$ remains constant. As we will illustrate in our subsequent analysis, this allows us to select a constant step-size γ that is independent of any hypothetical bound on these delays.

Along the algorithm, an iteration is triggered as soon as the central machine receives a communication. At the k-th iteration, with an incoming call from worker i, the central machine:

- receives a contribution u_i from agent i,
- updates $\overline{u}^k$ by setting $u_i^k = u_i$ and $u_{i'}^k = u_{i'}^{k-1}$ for all $i' \neq i$,
- computes the new point
$$x^k = \operatorname*{argmin}_{x \in \mathcal{X}} \left\{ h(x) + \gamma g(x) + \langle \overline{u}^k, x \rangle \right\} \tag{31}$$
- sends x^k to agent i.

This algorithm is described in Algorithm 1, where the workers communicate only the adjustments between two iterates (to avoid storing all u_i at the central machine).

Algorithm 1 `Asynchronous Bregman proximal-gradient algorithm`

Central machine:

```
Initialize $\overline{u}$, $k = 0$
Send $\overline{u}$ to all workers
while not stopped do
    when a worker finishes:
    Receive adjustment $\Delta$ from it
    $\overline{u} \leftarrow \overline{u} + \frac{\Delta}{M}$
    $x \leftarrow \operatorname*{argmin}_{x \in \mathcal{X}} h(x) + \gamma g(x) + \langle \overline{u}, x \rangle$
    Send $x$ to the agent in return
    $k \leftarrow k + 1$
end
Interrupt all slaves
Output $x$
```

Worker i:

```
Initialize $u = \overline{u}$
while not interrupted by central machine do
    Receive the most recent $x$
    $u^+ \leftarrow \gamma \nabla f_i(x) - \nabla h(x)$
    $\Delta \leftarrow u^+ - u$
    $u \leftarrow u^+$
    Send adjustment $\Delta$ to central machine
end
```

4 Analysis of the algorithm

The analysis of the algorithm described in the last section consists of two parts, to which the next two subsections are devoted: we establish first that the iterates are well-defined, and second that they converge to the minimum of (42), under no specific assumption on the computing system.

4.1 The algorithm is well-defined

To show that the iterates produced by Algorithm 1 are valid, we first show a generic result on the Bregman proximal gradient operator, which is at the core of our developments. The proof of this result simply consists in applying, in the product space $\bigtimes_{i=1}^M \mathcal{X}$, the general well-posedness result of Bregman proximal gradient operator of [4, Lemma 2].

Lemma 1 (Well-posedness)**.** *Let Assumptions 1 to 3 hold. For any stepsize $\gamma > 0$ and any vector $y = (y_1, \ldots, y_M) \in \bigtimes_{i=1}^M \operatorname{int}\operatorname{dom} h$, the operator*

$$T_\gamma(y) := \operatorname*{argmin}_{x\in\mathcal{X}} \left\{ \gamma g(x) + h(x) + \left\langle x; \frac{1}{M}\sum_{i=1}^M (\gamma\nabla f_i(y_i) - \nabla h(y_i)) \right\rangle \right\} \tag{32}$$

is non-empty and single-valued in $\operatorname{int}\operatorname{dom} h$*.*

Proof. Let $\gamma > 0$ and $y = (y_1, \ldots, y_M) \in \bigtimes_{i=1}^M \operatorname{int}\operatorname{dom} h$, let us define the operator

$$\begin{aligned}\mathbb{T}_\gamma(y) := \operatorname*{argmin}_{x\in\bigtimes_{i=1}^M \mathcal{X}} \Bigg\{ \frac{1}{M}\sum_{i=1}^M g(x_i) + \frac{1}{M}\sum_{i=1}^M [f_i(y_i) + \langle \nabla f_i(y_i), x_i - y_i\rangle] + \ldots \\ \cdots + \frac{1}{\gamma M}\sum_{i=1}^M D_h(x_i, y_i) + \iota_C(x) \Bigg\} \end{aligned} \tag{33}$$

with ι_C the consensus indicator defined by (1). By applying the same reasoning as in section 3.2, it is easy to see that $\mathbb{T}_\gamma(y) = (T_\gamma(y), \ldots, T_\gamma(y))$.

Under Assumptions 1 to 3, we have that i) $x \mapsto \iota_C(x) + \frac{1}{M}\sum_{i=1}^M g(x_i)$ is proper, lsc., and convex on $\bigtimes_{i=1}^M \mathcal{X}$; ii) $x \mapsto \frac{1}{M}\sum_{i=1}^M h(x_i)$ is Legendre on $\bigtimes_{i=1}^M \mathcal{X}$; iii) $x \mapsto \frac{1}{M}\sum_{i=1}^M f_i(x_i)$ is proper, lsc., convex, and differentiable on $\operatorname{int}\operatorname{dom} h$; iv) $\operatorname{dom}\iota_C \cap (\operatorname{dom} g)^M \cap \bigtimes_{i=1}^M \operatorname{int}\operatorname{dom} h \neq \emptyset$; and v) the problem $\inf_{x\in\mathbb{X}}\{\iota_C(x) + \frac{1}{M}\sum_{i=1}^M (f(x_i) + g(x_i))\}$ has a non-empty compact solution set. These conditions enable us to apply [4, Lem. 2] which gives that $\mathbb{T}_\gamma(y)$ is non-empty, single-valued, and maps $\bigtimes_{i=1}^M \operatorname{int}\operatorname{dom} h$ to $\bigtimes_{i=1}^M \operatorname{int}\operatorname{dom} h$. In turn, we obtain that $T_\gamma(y)$ is non-empty and single-valued in $\operatorname{int}\operatorname{dom} h$. □

Thus, noticing that in our algorithm, the central machine generates a new iterate x^k, as the point produced by applying T_γ to $y^k = (x^{k-D_1^k}, \ldots, x^{k-D_M^k})$, ie.

$$x^k = T_\gamma(y^k) = \operatorname*{argmin}_{x\in\mathcal{X}} \left\{ \gamma g(x) + h(x) + \left\langle x; \overbrace{\frac{1}{M}\sum_{i=1}^M \underbrace{\left(\gamma\nabla f_i(x^{k-D_i^k}) - \nabla h(x^{k-D_i^k})\right)}_{:=u_i^k}}^{\overline{u}^k} \right\rangle \right\}. \tag{34}$$

Thus Lemma 1 implies that our algorithm is well-defined, from an initial iterate $x^0 \in \operatorname{int}\operatorname{dom} h$. Indeed, the points $(x^{k'})_{k'<k}$ were generated by the central machine and thus belongs to $\operatorname{int}\operatorname{dom} h$; in turn, the input at iteration k is $y^k = (x^{k-D_1^k}, \ldots, x^{k-D_M^k})$ and

belongs to $\times_{i=1}^M \operatorname{int}\operatorname{dom} h$. The algorithm is thus well defined and produces points in $\operatorname{int}\operatorname{dom} h$.

4.2 Convergence result

We now focus on the convergence of the algorithm, established in forthcoming Theorem 1. The proof of convergence consists in carefully combining the contraction results of Bregman operators [13, 4] together with the techniques developed in [18] for the asynchronous proximal algorithms.

Lemma 2 (Contraction)**.** *Let Assumptions 1 to 3 hold. For any stepsize* $\gamma > 0$ *and any vector* $y = (y_1, \dots, y_M) \in \times_{i=1}^M \operatorname{int}\operatorname{dom} h$*, and any* $u \in \mathcal{X}$*,*

$$D_h(u, T_\gamma(y)) \le \frac{1}{M}\sum_{i=1}^M D_h(u, y_i) - \frac{1-\gamma L}{M}\sum_{i=1}^M D_h(T_\gamma(y), y_i) - \gamma(F(T_\gamma(y)) - F(u)). \tag{35}$$

Proof. The proof follows the same reasoning as the one of Lemma 1: working on the product space and deriving properties on the operator $\mathbb{T}_\gamma$. By applying [4, Lem. 5] (whose assumptions are verified as in Lemma 1) with $(u, \dots, u) \in \mathcal{X}^M$ and $\mathbb{T}_\gamma(y)$,

$$\begin{aligned}\frac{1}{M}\sum_{i=1}^M D_h(u, [\mathbb{T}_\gamma(y)]_i) \le \frac{1}{M}\sum_{i=1}^M D_h(u, x_i) - \frac{1-\gamma L}{M}\sum_{i=1}^M D_h([\mathbb{T}_\gamma(y)]_i, y_i) \dots \\ \dots - \gamma\Bigg(\iota_C(\mathbb{T}_\gamma(y)) + \frac{1}{M}\sum_{i=1}^M f_i([\mathbb{T}_\gamma(y)]_i) + g([\mathbb{T}_\gamma(y)]_i) \dots \\ \dots - \iota_C(u, \dots, u) - \frac{1}{M}\sum_{i=1}^M f_i(u) + g(u)\Bigg)\end{aligned} \tag{36}$$

and then, since $\mathbb{T}_\gamma(y) = (T_\gamma(y), \dots, T_\gamma(y))$, we obtain the claimed result. □

Since any solution $x^\star$ of $(\mathcal{P})$ belongs to $\operatorname{dom} h = \mathcal{X}$, applying Lemma 2 to the iterates produced by our algorithm with $u = x^\star$ shows that for all k,

$$D_h(x^\star, x^k) \le \frac{1}{M}\sum_{i=1}^M D_h(x^\star, x^{k-D_i^k}) - \frac{1-\gamma L}{M}\sum_{i=1}^M D_h(x^k, x^{k-D_i^k}) \dots \tag{37}$$

$$\dots - \gamma\left(F(x^k) - F(x^\star)\right) \tag{38}$$

which is central in the proof of the following convergence result. The pointwise convergence arguments in the proof of this result are based on Opial-type arguments for which the following additional condition is needed. This assumption is verified for many usual divergences, including the Boltzmann-Shannon entropy, the Hellinger distance, etc. It corresponds to assumption **H** in [4].

Assumption 4 (Level boundedness and limit)**.** *For any* $x \in \operatorname{dom} h$ *and* $t \in \mathbb{R}$*, the set* $\{y \in \operatorname{int}\operatorname{dom} h : D_h(x, y) \le t\}$ *is bounded. In addition,* $x^k \to x$ *if and only if* $D_h(x, x^k) \to 0$.

A key feature of the algorithm is that its convergence does not rely on prior knowledge or assumptions on the answering delays. It is only expected that the workers are never dropped, meaning that they all have a finite answering time. Let us introduce this assumption formally.

To do so we need to define the concept of "epoch" denoted $m \in \mathbb{N}$, and of the first iteration k_m of an "epoch" m (drawing from [19]).

- An iteration k belongs to epoch m if it falls in the range $k_m \leq k < k_{m+1}$
- The epoch m starts at 0 (together with k) and is incremented whenever the central node has made at least one *full interaction* with each worker during the current epoch. Namely, whenever all workers have answered at least one query sent after k_m. Since at iteration k, the most recent computation by a worker i was requested at iteration $k - D_i^k$, this condition therefore translates to $k_m \leq k - D_i^k$:

$$\begin{aligned} k_m &= \min\{k : \text{at least 1 full interaction per worker since } k_{m-1}\} && (39)\\ &= \min\{k : k - D_i^k \geq k_{m-1} \text{ for all } i = 1, .., M\}. && (40)\end{aligned}$$

Assumption 5 (Worker participation)**.** *Workers have a finite answering time, in other words,* $m \underset{k\to+\infty}{\longrightarrow} +\infty$.

Note that when Assumption 5 does not hold, we have "a subset of workers ceases to respond" and this leads to a different problem: in such cases, the non-responsive workers can be discarded, and the analysis is restricted to the active workers.

We are now ready to formulate the main result of this paper, namely, the convergence of our asynchronous algorithm.

Theorem 1 (Convergence)**.** *Let Assumptions 1 to 5, hold and fix $\gamma \in (0, 1/L)$. If $(\mathcal{P})$ has a unique minimizer $x^\star$, then the sequence (x^k) generated by Algorithm 1 converges to $x^\star$.*

Proof. Let us start with using (37). Since $\gamma \in (0, 1/L)$, we can disregard the second term involving $D_h(x^k, x^{k-D_i^k})$ which is negative. Then we get that for any $k \geq k_m$ the following bound:

$$D_h(x^\star, x^k) + \gamma\left(F(x^k) - F(x^\star)\right) \leq \max_i D_h(x^\star, x^{k-D_i^k}) \leq \max_{k' \in [k_{m-1}, k)} D_h(x^\star, x^{k'}). \quad (41)$$

In particular, for $k = k_m$, this implies

$$D_h(x^\star, x^{k_m}) \leq \max_{k' \in [k_{m-1}, k_m)} D_h(x^\star, x^{k'}). \quad (42)$$

Applying now (41) for $k = k_m + 1$, we obtain

$$D_h(x^\star, x^{k_m+1}) + \gamma\left(F(x^{k_m+1}) - F(x^\star)\right) \leq \max\left\{D_h(x^\star, x^{k_m}), \max_{k' \in [k_{m-1}, k_m)} D_h(x^\star, x^{k'})\right\} \quad (43)$$

$$\leq \max_{k' \in [k_{m-1}, k_m)} D_h(x^\star, x^{k'}) \quad (44)$$

where we use (42) for the second inequality. Repeating this recursively, we obtain for all $k \in [k_m, k_{m+1})$

$$\big(0 \leq \big)\ D_h(x^\star, x^k) + \gamma \left(F(x^k) - F(x^\star)\right) \leq \max_{k' \in [k_{m-1}, k_m)} D_h(x^\star, x^{k'}). \tag{45}$$

Observe that this bound yields

$$\max_{k \in [k_m, k_{m+1})} D_h(x^\star, x^k) \leq \max_{k' \in [k_{m-1}, k_m)} D_h(x^\star, x^{k'}). \tag{46}$$

In words, the non-negative sequence $\max_{k \in [k_m, k_{m+1})} D_h(x^\star, x^k)$ is non-increasing and thus converges to a non-negative value.

Consider now a sequence of time indices (l^m) where the above maximum is attained:

$$l^m \in \arg\max_{k \in [k_m, k_{m+1})} D_h(x^\star, x^k).$$

Since $l^m \in [k_m, k_{m+1})$, we have

$$D_h(x^\star, x^{l^m}) + \gamma \left(F(x^{l^m}) - F(x^\star)\right) \leq \max_{k \in [k_m, k_{m+1})} D_h(x^\star, x^k) + \gamma \left(F(x^k) - F(x^\star)\right) \tag{47}$$

which gives, with the help of (45),

$$D_h(x^\star, x^{l^m}) + \gamma \left(F(x^{l^m}) - F(x^\star)\right) \leq \max_{k' \in [k_{m-1}, k_m)} D_h(x^\star, x^{k'}) \ = \ D_h(x^\star, x^{l^{m-1}}). \tag{48}$$

By letting $m \to \infty$ in this inequality, we obtain

$$\lim_{m \to \infty} F(x^{l^m}) = F(x^\star). \tag{49}$$

Since we have $D_h(x^\star, x^{l^m}) \leq D_h(x^\star, x^0)$, Assumption 4 yields that (x^{l^m}) is bounded, so that we can extract a sub-sequence that converges to a point $\tilde{x} \in \operatorname{dom} h$. From (49), $\tilde{x}$ must be a minimizer for F, and by uniqueness, we have $\tilde{x} = x^\star$. Thus we have $D_h(x^\star, x^{l^m}) \longrightarrow_{m \to \infty} 0$. This allows us to conclude from (45): the right-hand side vanishes, so that $D_h(x^\star, x^k) \to 0$ and $F(x^k) \to F(x^\star)$.

$\square$

5 Numerical illustrations

In this section, we take a look at the practical behavior of our asynchronous Bregman algorithm on a simple synthetic distributed problem. We provide an illustration of the convergence result and a comparison with competing algorithms. A complete computational study is out of the scope of this paper. The numerical illustrations are operated in `Julia` [10] on a personal computer. We have packaged a toolbox implementing the algorithms and the experiments; it is publicly available at `github.com/Selim78/distributed-bregman`.

5.1 Experimental set-up

We illustrate our algorithm on a regularized non-negative linear regression problem, which is a distributed variant of the problem of [4, Sec. 5.3] (see also [14]). It consists in the minimization of the function of Example 1 with an additional ℓ_1-regularization to ensure that the problem has a unique minimizer. Let $A \in \mathbb{R}_+^{m \times n}$ be a sensing matrix and $b \in \mathbb{R}_{++}^m$ a positive output vector. Assume that we split the m samples of this dataset on M workers (for simplicity, we consider in our experiments an even partition of m/M examples). With a_j representing the j-th row of A, the objective function writes

$$\min_{x\in\mathbb{R}^n_+} \frac{1}{M}\sum_{i=1}^{M} \underbrace{\left(\sum_{j=\frac{(i-1)m}{M}+1}^{\frac{im}{M}} \langle a_j, x\rangle \log\langle a_j, x\rangle - (\log b_j + 1)\langle a_j, x\rangle + b_j\right)}_{f_i(x)} + \lambda\|x\|_1. \tag{50}$$

In practice, the data is generated as follows. We take $n = 100$ and $m = 200$; the rows of matrix A are drawn from a uniform distribution in $[0, 1)$. The vector b is generated as $b = A\bar{x} + \epsilon$ with a random positive $\bar{x}$ plus a Poisson noise with rate 1. This problem is distributed on $M = 10$ processes living in separate memory domains using `Julia`'s Distributed module, which allows the control of process generation and communication between processes. To simulate some heterogeneity between the workers, we artificially slow down two workers by a factor 5 and 10 respectively.

We solve this problem with the three following Bregman algorithms using the Boltzmann-Shannon entropy $h(x) = \sum_{j=1}^n x_j \log x_j$ (since the first part of the objective function is L-smooth with respect to it with L as in (22) as discussed in Example 1):

- the synchronous algorithm based on the iteration (5) (called Synchronous),
- the asynchronous variant based on the iteration (28) (called Bregman-PIAG),
- our asynchronous variant (Algorithm 1).

We note that our asynchronous algorithm, as well as the synchronous one, is guaranteed to converge with the standard stepsize $\gamma < 1/L$. In contrast, Bregman-PIAG has no convergence guarantee, as recalled at the end of section 3.2. In practice, we run all algorithms with, both, a theory-complying $0.99/L$ stepsize and a tuned stepsize.

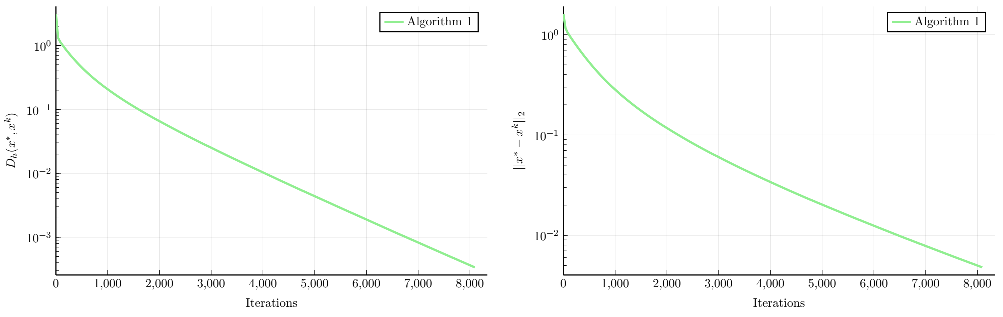

FIGURE 2. Convergence with respect to the number of iterations, illustrating Theorem 1

5.2 Experimental results

In Figure 2, we show convergence of the iterates of Algorithm 1 (with $\gamma = 0.99/L$), as guaranteed by Theorem 1. The two plots display a decrease of, respectively, $D_h(x^k, x^\star)$

and $\|x^k - x^\star\|^2$ over iterations[5]. We see on the plots that this decrease is mostly monotonous; we note that the proof of Theorem 1 establishes monotonicity for the worst Bregman divergence over epochs (46).

In Figure 3, we report the performance for the three algorithms described in the previous section. Since iterations have two different meanings for asynchronous or synchronous algorithms, we compare the convergence speed in terms of wallclock time. On the left-hand plot, we use the theoretical stepsize γ for the three algorithms, and on the right-hand plot, we use tuned stepsizes[6]. We notice a clear gain in performance with our approach, in the two cases. Note finally that the instance of the problem, generated as described in Section 5.1, is only weakly heterogeneous. We choose it on purpose to compare our algorithm with two other algorithms in a situation that is not, apriori, the best for our algorithm. Even better performances can be obtained for stronger heterogeneity of the data or the system. A complete computational study is out of the scope of this paper, which is mainly methodological.

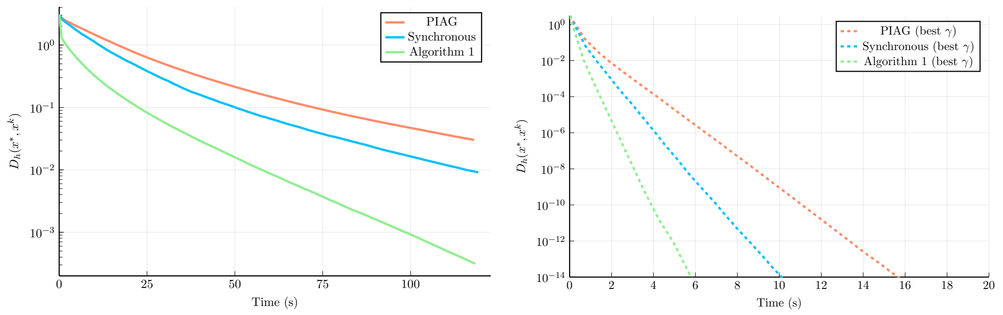

FIGURE 3. Comparison of our algorithm with two existing ones, with respect to wall-clock time

6 Conclusions, perspectives

In this paper, we provided an asynchronous version of the Bregman proximal gradient method. Building on efficient asynchronous methods for the Euclidean case and on smoothness models adapted to Bregman geometries, we derive and analyze a method that can handle any kind of delays, with a simple implementation using the same step size as in the synchronous case. The light assumption on the delays combined with the subtlety of Bregman geometry make the analysis rather involved: we were able to show the convergence of our method whenever there is a unique minimizer to our problem; nevertheless, we believe that finer results could be derived using the same proof template. For instance, it may be possible to obtain a local convergence rate in the strongly convex case using the Legendre exponent reasoning of [3].

Acknowledgements

This work has been supported by MIAI Grenoble Alpes (ANR-19-P3IA-0003).

[5]The optimal solution $x^\star$ is obtained by running the synchronous algorithm for a long time.

[6]For each algorithm, γ is chosen among the exponential of $1, 1.5, \ldots, 2.5, 3$. The γ retained is the one which produces the last iterate with the minimum Bergman distance to x^* after a 100 second run.

References

[1] Auslender, A., Teboulle, M.: Interior gradient and proximal methods for convex and conic optimization. SIAM Journal on Optimization **16**(3), 697–725 (2006)

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

[3] Azizian, W., Iutzeler, F., Malick, J., Mertikopoulos, P.: The last-iterate convergence rate of optimistic mirror descent in stochastic variational inequalities. In: Conference on Learning Theory, pp. 326–358. PMLR (2021)

[4] Bauschke, H.H., Bolte, J., Teboulle, M.: A descent lemma beyond lipschitz gradient continuity: first-order methods revisited and applications. Mathematics of Operations Research **42**(2), 330–348 (2017)

[5] Bauschke, H.H., Borwein, J.M., et al.: Legendre functions and the method of random bregman projections. Journal of convex analysis **4**(1), 27–67 (1997)

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

[7] Beck, A.: First-order methods in optimization. SIAM (2017)

[8] Bertero, M., Boccacci, P., Desiderà, G., Vicidomini, G.: Image deblurring with poisson data: from cells to galaxies. Inverse Problems **25**(12), 123006 (2009)

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

[10] Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing. SIAM review **59**(1), 65–98 (2017)

[11] Bolte, J., Sabach, S., Teboulle, M., Vaisbourd, Y.: First order methods beyond convexity and lipschitz gradient continuity with applications to quadratic inverse problems. SIAM Journal on Optimization **28**(3), 2131–2151 (2018)

[12] Bregman, L.M.: The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. USSR computational mathematics and mathematical physics **7**(3), 200–217 (1967)

[13] Chen, G., Teboulle, M.: Convergence analysis of a proximal-like minimization algorithm using bregman functions. SIAM Journal on Optimization **3**(3), 538–543 (1993)

[14] Csiszar, I.: Why least squares and maximum entropy? an axiomatic approach to inference for linear inverse problems. The annals of statistics **19**(4), 2032–2066 (1991)

[15] Hannah, R., Yin, W.: More iterations per second, same quality–why asynchronous algorithms may drastically outperform traditional ones. arXiv:1708.05136 (2017)

[16] Hannah, R., Yin, W.: On unbounded delays in asynchronous parallel fixed-point algorithms. Journal of Scientific Computing **76**(1), 299–326 (2018)

[17] Lu, H., Freund, R.M., Nesterov, Y.: Relatively-smooth convex optimization by first-order methods, and applications (2017)

[18] Mishchenko, K., Iutzeler, F., Malick, J.: A distributed flexible delay-tolerant proximal gradient algorithm. SIAM Journal on Optimization **30**(1), 933–959 (2020)

[19] Mishchenko, K., Iutzeler, F., Malick, J., Amini, M.R.: A delay-tolerant proximal-gradient algorithm for distributed learning. In: Proceedings of the 35th international conference on machine learning (ICML) (2018)

[20] Nesterov, Y.: Introductory lectures on convex optimization: A basic course, vol. 87. Springer Science & Business Media (2013)

[21] Rockafellar, R.T.: Convex analysis. Princeton university press (2015)

[22] Sun, T., Hannah, R., Yin, W.: Asynchronous coordinate descent under more realistic assumptions. In: Advances in Neural Information Processing Systems, pp. 6182–6190 (2017)

[23] Vanli, N.D., Gürbüzbalaban, M., Ozdaglar, A.: Global convergence rate of proximal incremental aggregated gradient methods. SIAM Journal on Optimization **28**(2), 1282–1300 (2018)