A Coordinate Descent Primal-Dual Algorithm and Application to Distributed Asynchronous Optimization

Pascal Bianchi, Walid Hachem, Franck Iutzeler

A Coordinate Descent Primal-Dual Algorithm and Application to Distributed Asynchronous Optimization

P. Bianchi, W. Hachem and F. Iutzeler

***Abstract*— Based on the idea of randomized coordinate descent of α-averaged operators, a randomized primal-dual optimization algorithm is introduced, where a random subset of coordinates is updated at each iteration. The algorithm builds upon a variant of a recent (deterministic) algorithm proposed by Vũ and Condat that includes the well known ADMM as a particular case. The obtained algorithm is used to solve asynchronously a distributed optimization problem. A network of agents, each having a separate cost function containing a differentiable term, seek to find a consensus on the minimum of the aggregate objective. The method yields an algorithm where at each iteration, a random subset of agents wake up, update their local estimates, exchange some data with their neighbors, and go idle. Numerical results demonstrate the attractive performance of the method.**
The general approach can be naturally adapted to other situations where coordinate descent convex optimization algorithms are used with a random choice of the coordinates.

***Index Terms*— Distributed Optimization, Coordinate Descent, Consensus algorithms, Primal-Dual Algorithm.**

I. Introduction

Let $\mathcal{X}$ and $\mathcal{Y}$ be two Euclidean spaces and let $M : \mathcal{X} \to \mathcal{Y}$ be a linear operator. Given two real convex functions f and g on $\mathcal{X}$ and a real convex function h on $\mathcal{Y}$, we consider the minimization problem

$$\inf_{x\in\mathcal{X}} f(x) + g(x) + h(Mx) \tag{1}$$

where f is differentiable and its gradient ∇f is Lipschitz-continuous. Although our theoretical contributions are valid for very general functions f, g and h, the application part of this paper puts a special emphasis on the problem of distributed optimization. In this particular framework, one considers a set of N agents such that each agent $n = 1, \dots, N$ has a private cost of the form $f_n + g_n$ where f_n and g_n are two convex cost function on some (other) space $\overline{\mathcal{X}}$, f_n being differentiable. The aim is to distributively solve

$$\inf_{u\in\overline{\mathcal{X}}} \sum_{n=1}^{N} f_n(u) + g_n(u) \ . \tag{2}$$

The first two authors are with the CNRS LTCI; Telecom ParisTech, Paris, France. The third author is with LJK, Université Joseph Fourier, Grenoble, France. E-mails: pascal.bianchi, walid.hachem@telecom-paristech.fr, franck.iutzeler@imag.fr. This work was granted by the French Defense Agency (DGA) ANR-grant ODISSEE.

In order to construct distributed algorithms a standard approach consists in introducing

$$f(x) = \sum_{n=1}^{N} f_n(x_n) \quad \text{and} \quad g(x) = \sum_{n=1}^{N} g_n(x_n)$$

for all $x = (x_1, \dots, x_N)$ in the product space $\mathcal{X} = \overline{\mathcal{X}}^N$. Obviously, problem (2) is equivalent to the minimization of $f(x) + g(x)$ under the constraint that all components of x are equal *i.e.*, $x_1 = \cdots = x_N$. Therefore, Problem (2) is in fact a special instance of Problem (1) if one chooses $h(Mx)$ as an indicator function, equal to zero if $x_1 = \cdots = x_N$ and to $+\infty$ otherwise. As we shall see, this reformulation of Problem (2) is often a mandatory step in the construction of distributed algorithms.

Our contributions are as follows.

1) Vũ and Condat have separately proposed an algorithm to solve (1) in [1] and [2] respectively. Elaborating on this algorithm, we provide an iterative algorithm for solving (1) which we refer to as ADMM+ (Alternating Direction Method of Multipliers plus) because it includes the well known ADMM [3], [4] as the special case corresponding to $f = 0$. Interestingly, in the framework of the distributed optimization, ADMM+ is provably convergent under weaker assumptions on the step sizes as compared to the original Vũ/Condat algorithm.
2) Based on the idea of the *stochastic coordinate descent* who has been mainly studied in the literature in the special case of proximal gradient algorithms [5]–[7], we develop a *distributed asynchronous* version of ADMM+. As a first step, we borrow from [1]- [2] the idea that their algorithm is an instance of a so-called Krasnosel'skii-Mann iteration applied to an α-averaged operator [8, Section 5.2]. Such operators have contraction-like properties that make the Krasnosel'skii-Mann iterations converge to a fixed point of the operator. The principle of the stochastic coordinate descent algorithms is to update only a random subset of coordinates at each iteration. In this paper, we show in most generality that a randomized coordinate descent version of the Krasnosel'skii-Mann iterations still converges to a fixed point of an α-averaged operator. This provides as a side result a convergence proof of the stochastic coordinate descent versions of the proximal gradient algorithm, since this algorithm can be seen as the application of a $1/2$-averaged operator [8]. More importantly in the context of this paper, this idea

leads to provably convergent asynchronous distributed versions of ADMM+.

3) Putting together both ingredients above, we apply our findings to asynchronous distributed optimization. First, the optimization problem (1) is rewritten in a form where the operator M encodes the connections between the agents within a graph in a manner similar to [9]. Then, a distributed optimization algorithm for solving Problem (2) is obtained by applying ADMM+. Using the idea of coordinate descent on the top of the algorithm, we then obtain a fully asynchronous distributed optimization algorithm that we refer to as Distributed Asynchronous Primal Dual algorithm (DAPD). At each iteration, an independent an identically distributed random subset of agents wake up, apply essentially the proximity operator on their local functions, send some estimates to their neighbors and go idle.

An algorithm that has some formal resemblance with ADMM+ was proposed in [10], who considers the minimization of the sum of two functions, one of them being subjected to noise. This reference includes a linearization of the noisy function in ADMM iterations.

The use of stochastic coordinate descent on averaged operators has been introduced in [11] (see also the recent preprint [12] which uses the same line of thought). Note that the approach of [11] was limited to unrelaxed firmly non expansive (or $1/2$-averaged) operators, well-suited for studying ADMM which was the algorithm of interest in [11].

Asynchronous distributed optimization is a promising framework in order to scale up machine learning problems involving massive data sets (we refer to [13] or the recent survey [14]). Early works on distributed optimization include [15], [16] where a network of processors seeks to optimize some objective function known by all agents (possibly up to some additive noise). More recently, numerous works extended this kind of algorithm to more involved multi-agent scenarios, see [17]–[28].

Note that standard first order distributed optimization methods often rely on the so-called adaptation-diffusion approaches or variants. The agents update their local estimates by evaluating their private gradient and then merge their estimate with their neighbors using a local averaging step. Unfortunately, such methods require the use of a vanishing step size, which results in slow convergence. This paper proposes a first-order distributed optimization method with constant step size, which turns out to outperform standard distributed gradient methods, as shown in the simulations.

To the best of our knowledge, our method is the first distributed algorithm combining the following attractive features:

1) The algorithm is asynchronous at the *node-level*. Only a single node is likely to be active at a given iteration, only broadcasting the result of its computation without expecting any feedback from other nodes. This is in contrast with the asynchronous ADMM studied by [11] and [29] which is only asynchronous at the *edge-level*. In these works, at least two connected nodes are supposed to be active at a common time.
2) The algorithm is a *proximal method*. Similarly to the distributed ADMM, it allows for the use of a proximity operator at each node. This is especially important to cope with the presence of possibly non-differentiable regularization terms. This is unlike the classical adaptation-diffusion methods mentioned above or the more recent first order distributed algorithm EXTRA proposed by [28].
3) The algorithm is a *first-order method*. Similarly to adaptation-diffusion methods, our algorithm allows to compute gradients of the local cost functions. This is unlike the distributed ADMM which only admits implicit steps *i.e.*, agents are required to locally solve an optimization problem at each iteration.
4) The algorithm admits *constant step size*. As remarked in [28], standard adaptation-diffusion methods require the use of a vanishing step size to ensure the convergence to the sought minimizer. In practice, this comes at the price of slow convergence. Our method allows for the use of a constant step size in the gradient descent step.

The paper is organized as follows. Section II is devoted to the the introduction of ADMM+ algorithm and its relation with the Primal-Dual algorithms of Vũ [1] and Condat [2], we also show how ADMM+ includes both the standard ADMM and the Forward-Backward algorithm (also refered to as proximal gradient algorithm) as special cases [8, Section 25.3]. In Section III, we provide our result on the convergence of Krasnosel'skii-Mann iterations with randomized coordinate descent. Section IV addresses the problem of asynchronous distributed optimization. Finally, Section V provides numerical results.

II. A Primal Dual Algorithm

A. Problem statement

We consider Problem (1). Denoting by $\Gamma_0(\mathcal{X})$ the set of proper lower semi-continuous convex functions on $\mathcal{X} \to (-\infty, \infty]$ and by $\|\cdot\|$ the norm on $\mathcal{X}$, we make the following assumptions:

Assumption 1 *The following facts hold true:*
(i) f *is a convex differentiable function on* $\mathcal{X}$*,*
(ii) $g \in \Gamma_0(\mathcal{X})$ *and* $h \in \Gamma_0(\mathcal{Y})$*.*

We consider the case where M is injective (in particular, it is implicit that $\dim(\mathcal{X}) \leq \dim(\mathcal{Y})$). In the latter case, we denote by $\mathcal{S} = \mathrm{Im}(M)$ the image of M and by M^{-1} the inverse of M on $\mathcal{S} \to \mathcal{X}$. We emphasize the fact that the inclusion $\mathcal{S} \subset \mathcal{Y}$ might be strict. We denote by ∇ the gradient operator.

Assumption 2 *The following facts hold true:*
(i) M *is injective ,*
(ii) $\nabla(f \circ M^{-1})$ *is L-Lipschitz continuous on* $\mathcal{S}$*.*

We denote by $\mathrm{dom}\, q$ the domain of a function q and by $\mathrm{ri}\, S$ the relative interior of a set S in a Euclidean space.

Assumption 3 *The infimum of Problem* (1) *is attained. Moreover, the following qualification condition holds*

$$0 \in \mathrm{ri}(\mathrm{dom}\, h - M\, \mathrm{dom}\, g)$$

where $M\,\mathrm{dom}\,g$ is the image by M of $\mathrm{dom}\,g$.

The *dual* problem corresponding to the *primal* problem (1) is written

$$\inf_{\lambda\in\mathcal{Y}}(f+g)^*(-M^*\lambda)+h^*(\lambda)$$

where q^* denotes the Legendre-Fenchel transform of a function q and where M^* is the adjoint of M. With the assumptions 1 and 3, the classical Fenchel-Rockafellar duality theory [8], [30] shows that

$$\min_{x\in\mathcal{X}} f(x)+g(x)+h(Mx) = -\inf_{\lambda\in\mathcal{Y}}(f+g)^*(-M^*\lambda)+h^*(\lambda), \quad (3)$$

and the infimum at the right hand member is attained. Furthermore, denoting by ∂q the subdifferential of a function $q\in\Gamma_0(\mathcal{X})$, any point $(\bar{x},\bar{\lambda})\in\mathcal{X}\times\mathcal{Y}$ at which the above equality holds satisfies

$$\begin{cases} 0\in\nabla f(\bar{x})+\partial g(\bar{x})+M^*\bar{\lambda} \\ 0\in -M\bar{x}+\partial h^*(\bar{\lambda}) \end{cases}$$

and conversely. Such a point is called a *primal-dual* point.

B. A Primal-Dual Algorithm

We denote by $\langle\cdot,\cdot\rangle$ the inner product on $\mathcal{X}$. We keep the same notation $\|\cdot\|$ to represent the norm on both $\mathcal{X}$ and $\mathcal{Y}$. For some parameters $\rho,\tau>0$, we consider the following algorithm which we shall refer to as ADMM+.

ADMM+

$$z^{k+1}=\operatorname*{argmin}_{z\in\mathcal{Y}}\Big[h(z)+\frac{\|z-(Mx^k+\rho\lambda^k)\|^2}{2\rho}\Big] \quad (4a)$$

$$\lambda^{k+1}=\lambda^k+\rho^{-1}(Mx^k-z^{k+1}) \quad (4b)$$

$$u^{k+1}=(1-\tau\rho^{-1})Mx^k+\tau\rho^{-1}z^{k+1} \quad (4c)$$

$$x^{k+1}=\operatorname*{argmin}_{x\in\mathcal{X}}\Big[g(x)+\langle\nabla f(x^k),x\rangle + \frac{\|Mx-u^{k+1}+\tau\lambda^{k+1}\|^2}{2\tau}\Big] \quad (4d)$$

This algorithm is especially useful in the situations where ∇f and the left hand member of Equation (4d) are both easy to compute, as it is the case when (say) f is quadratic and g is an ℓ_1 regularization term. In such situations, working directly on $f+g$ is often computationally demanding.

Theorem 1 *Let Assumptions 1–3 hold true. Assume that $\tau^{-1}-\rho^{-1}>L/2$. For any initial value $(x^0,\lambda^0)\in\mathcal{X}\times\mathcal{Y}$, the sequence (x^k,λ^k) defined by ADMM+ converges to a primal-dual point $(x^\star,\lambda^\star)$ of* (3) *as $k\to\infty$.*

Remark 1 *In the special case when $f=0$ (that is $L=0$), it turns out that the condition $\tau^{-1}-\rho^{-1}>L/2$ can be further weakened to $\tau^{-1}-\rho^{-1}\geq 0$ (see [2]). It is therefore possible to set $\tau=\rho$ and thus have a single instrumental parameter to tune in the algorithm. Note also that in $f=0$ the algorithm is provably convergent with no need to require the injectivity of M.*

The proof of Theorem 1 is provided in Appendix A. It is based on Theorem 2 below. For any function $g\in\Gamma_0(\mathcal{X})$ we denote by prox_g its proximity operator defined by

$$\mathrm{prox}_g(x)=\arg\min_{w\in\mathcal{X}}\Big[g(w)+\frac{1}{2}\|w-x\|^2\Big]. \quad (5)$$

The ADMM+ is an instance of the primal dual algorithm recently proposed by Vũ [1] and Condat [2], see also [31]:

Theorem 2 ([1], [2]) *Given a Euclidean space $\mathcal{E}$, consider the minimization problem $\inf_{y\in\mathcal{E}}\bar{f}(y)+\bar{g}(y)+h(y)$ where $\bar{g},h\in\Gamma_0(\mathcal{E})$ and where $\bar{f}$ is convex and differentiable on $\mathcal{E}$ with an $L-$Lipschitz continuous gradient. Assume that the infimum is attained and that $0\in\mathrm{ri}(\mathrm{dom}\,h-\mathrm{dom}\,\bar{g})$. Let $\tau,\rho>0$ be such that $\tau^{-1}-\rho^{-1}>L/2$, and consider the iterates*

$$\lambda^{k+1}=\mathrm{prox}_{\rho^{-1}h^*}(\lambda^k+\rho^{-1}y^k) \quad (6a)$$

$$y^{k+1}=\mathrm{prox}_{\tau\bar{g}}(y^k-\tau\nabla\bar{f}(y^k)-\tau(2\lambda^{k+1}-\lambda^k)). \quad (6b)$$

Then for any initial value $(y^0,\lambda^0)\in\mathcal{E}\times\mathcal{E}$, the sequence (y^k,λ^k) converges to a primal-dual point $(y^\star,\lambda^\star)$, i.e., *a solution of the equation*

$$\inf_{y\in\mathcal{E}}\bar{f}(y)+\bar{g}(y)+h(y)=-\inf_{\lambda\in\mathcal{E}}(\bar{f}+\bar{g})^*(-\lambda)+h^*(\lambda). \quad (7)$$

C. The case $f\equiv 0$ and the link with ADMM

In the special case $f\equiv 0$ and $\tau=\rho$, sequence $(u^k)_{k\in\mathbb{N}}$ coincides with $(z^k)_{k\in\mathbb{N}}$. Then, ADMM+ boils down to the standard ADMM whose iterations are given by:

$$z^{k+1}=\operatorname*{argmin}_{z\in\mathcal{Y}}\Big[h(z)+\frac{1}{2\rho}\|z-Mx^k-\rho\lambda^k\|^2\Big]$$
$$\lambda^{k+1}=\lambda^k+\rho^{-1}(Mx^k-z^{k+1})$$
$$x^{k+1}=\operatorname*{argmin}_{x\in\mathcal{X}}\Big[g(x)+\frac{1}{2\rho}\|Mx-z^{k+1}+\rho\lambda^{k+1}\|^2\Big].$$

D. The case $h\equiv 0$ and the link with the Forward-Backward algorithm

In the special case $h\equiv 0$ and $M=I$, it can be easily verified that λ^k is null for all $k\geq 1$ and $u^k=x^k$. Then, ADMM+ boils down to the standard Forward-Backward algorithm whose iterations are given by:

$$x^{k+1}=\operatorname*{argmin}_{x\in\mathcal{X}} g(x)+\frac{1}{2\tau}\|x-(x^k-\tau\nabla f(x^k))\|^2 = \mathrm{prox}_{\tau g}(x^k-\tau\nabla f(x^k)).$$

One can remark that ρ has disappeared thus it can be set as large as wanted so the condition on stepsize τ from Theorem 1 boils down to $\tau<2/L$. Applications of this algorithm with particular functions appear in well known learning methods such as ISTA [32].

E. Comparison to the original Vũ-Condat algorithm

We emphasize the fact that ADMM+ is a variation on the Vũ-Condat algorithm. The original Vũ-Condat algorithm is in general sufficient and, in many contexts, has even better properties than ADMM+ from an implementation point of view. Indeed, whereas the Vũ-Condat algorithm handles the

operator M explicitly, the step (4d) in ADMM+ can be delicate to implement in certain applications, *i.e.*, when M has no convenient structure allowing to easily compute the $\arg\min$ (the same remark holds of course for ADMM which is a special case of ADMM+).

This potential drawback is however not an issue in many other scenarios where the structure of M is such that step (4d) is affordable. In Section IV, we shall provide such scenarios where ADMM+ is especially relevant. In particular, ADMM+ is not only easy to implement but it is also provably convergent under weaker assumptions on the step sizes, as compared to the original Vũ-Condat algorithm.

Also, the injectivity assumption on M could be seen as restrictive at first glance. First, the latter assumption is in fact not needed when $f = 0$ as noted above. Second, it is trivially satisfied in the application scenarios which motivate this paper (see the next sections).

As alluded to in the introduction, the primal-dual algorithm of [1], [2] can be geometrically described as a sequence of Krasnosel'skii-Mann iterations applied to an α-averaged operator. In the next section, we briefly present these notions, proceed by introducing the randomized coordinate descent version of these iterations, then state our convergence result.

III. Coordinate Descent

A. Averaged operators and the primal-dual algorithm

Let $\mathcal{H}$ be a Euclidean space[1]. For $0 < \alpha \leq 1$, a mapping $\mathsf{T} : \mathcal{H} \to \mathcal{H}$ is *α-averaged* if the following inequality holds for any x, y in $\mathcal{H}$:

$$\|\mathsf{T}x - \mathsf{T}y\|^2 \leq \|x - y\|^2 - \frac{1-\alpha}{\alpha}\|(\mathsf{I} - \mathsf{T})x - (\mathsf{I} - \mathsf{T})y\|^2 .$$

A 1-averaged operator is said *non-expansive*. A $\frac{1}{2}$-averaged operator is said *firmly non-expansive*. The following Lemma can be found in [8, Proposition 5.15, pp.80].

Lemma 1 (Krasnosel'skii-Mann iterations) *Assume that $\mathsf{T} : \mathcal{H} \to \mathcal{H}$ is α-averaged and that the set $\mathrm{fix}(\mathsf{T})$ of fixed points of T is non-empty. Consider a sequence $(\eta_k)_{k\in\mathbb{N}}$ such that $0 \leq \eta_k \leq 1/\alpha$ and $\sum_k \eta_k(1/\alpha - \eta_k) = \infty$. For any $x^0 \in \mathcal{H}$, the sequence $(x^k)_{k\in\mathbb{N}}$ recursively defined on $\mathcal{H}$ by $x^{k+1} = x^k + \eta_k(\mathsf{T}x^k - x^k)$ converges to some point in $\mathrm{fix}(\mathsf{T})$.*

On the product space $\mathcal{Y} \times \mathcal{Y}$, consider the operator

$$\mathsf{V} = \begin{pmatrix} \tau^{-1}\mathsf{I}_\mathcal{Y} & \mathsf{I}_\mathcal{Y} \\ \mathsf{I}_\mathcal{Y} & \rho\mathsf{I}_\mathcal{Y} \end{pmatrix}$$

where $\mathsf{I}_\mathcal{Y}$ stands for the identity on $\mathcal{Y} \to \mathcal{Y}$. When $\tau^{-1}-\rho^{-1} > 0$, one can easily check that V is positive definite. In this case, we endow $\mathcal{Y} \times \mathcal{Y}$ with an inner product $\langle \cdot, \cdot \rangle_\mathsf{V}$ defined as $\langle \zeta, \varphi \rangle_\mathsf{V} = \langle \zeta, \mathsf{V}\varphi \rangle$ where $\langle \cdot, \cdot \rangle$ stands for the natural inner product on $\mathcal{Y} \times \mathcal{Y}$. We denote by $\mathcal{H}_\mathsf{V}$ the corresponding Euclidean space.

In association with Lemma 1, the following lemma is at the heart of the proof of Theorem 2:

[1]We refer to [8] for an extension to Hilbert spaces.

Lemma 2 ([1], [2]) *Let Assumptions 1–2 hold true. Assume that $\tau^{-1}-\rho^{-1} > L/2$. Let $(\lambda^{k+1}, y^{k+1}) = \mathsf{T}(\lambda^k, y^k)$ where T is the transformation described by Equations* (6a)–(6b)*. Then T is an α-averaged operator on $\mathcal{H}_\mathsf{V}$ with $\alpha = (2-\alpha_1)^{-1}$ and $\alpha_1 = (L/2)(\tau^{-1} - \rho^{-1})^{-1}$.*

Note that $\tau^{-1} - \rho^{-1} > L/2$ implies that $1 > \alpha_1 \geq 0$ and thus that α verifies $\frac{1}{2} \leq \alpha < 1$ which matches the definition of α-averaged operators.

B. Randomized Krasnosel'skii Mann Iterations

Consider the space $\mathcal{H} = \mathcal{H}_1 \times \cdots \times \mathcal{H}_J$ for some integer $J \geq 1$ where for any j, $\mathcal{H}_j$ is a Euclidean space. Assume that $\mathcal{H}$ is equipped with the scalar product $\langle x, y \rangle = \sum_{j=1}^J \langle x_j, y_j \rangle_{\mathcal{H}_j}$ where $\langle \cdot, \cdot \rangle_{\mathcal{H}_j}$ is the scalar product in $\mathcal{H}_j$. For $j \in \{1, \ldots, J\}$, we denote by $\mathsf{T}_j : \mathcal{H} \to \mathcal{H}_j$ the components of the output of operator $\mathsf{T} : \mathcal{H} \to \mathcal{H}$ corresponding to $\mathcal{H}_j$, we thus have $\mathsf{T}x = (\mathsf{T}_1 x, \ldots, \mathsf{T}_J x)$. We denote by $2^\mathcal{J}$ the power set of $\mathcal{J} = \{1, \ldots, J\}$. For any $\kappa \in 2^\mathcal{J}$, we define the operator $\hat{\mathsf{T}}^{(\kappa)} : \mathcal{H} \to \mathcal{H}$ by $\hat{\mathsf{T}}_j^{(\kappa)} x = \mathsf{T}_j x$ if $j \in \kappa$ and $\hat{\mathsf{T}}_j^{(\kappa)} x = x_j$ otherwise. On some probability space $(\Omega, \mathcal{F}, \mathbb{P})$, we introduce a random i.i.d. sequence $(\xi^k)_{k\in\mathbb{N}^*}$ such that $\xi^k : \Omega \to 2^\mathcal{J}$ *i.e.* $\xi^k(\omega)$ is a subset of $\mathcal{J}$. We assume that the following holds:

$$\forall j \in \mathcal{J},\ \exists \kappa \in 2^\mathcal{J} \text{ s.t. } j \in \kappa \text{ and } \mathbb{P}(\xi_1 = \kappa) > 0 . \qquad (8)$$

Let T be an α-averaged operator, instead of considering the iterates $x^{k+1} = x^k + \eta_k(\mathsf{T}x^k - x^k)$, we are now interested in a stochastic *coordinate descent* version of this algorithm that consists in iterates of the type $x^{k+1} = x^k + \eta_k(\hat{\mathsf{T}}^{(\xi^{k+1})}x^k - x^k)$. The proof of Theorem 3 is provided in Appendix B.

Theorem 3 *Let $\mathsf{T} : \mathcal{H} \to \mathcal{H}$ be α-averaged and $\mathrm{fix}(\mathsf{T}) \neq \emptyset$. Assume that for all k, the sequence $(\eta_k)_{k\in\mathbb{N}}$ satisfies*

$$0 < \liminf_k \eta_k \leq \limsup_k \eta_k < \frac{1}{\alpha} .$$

Let $(\xi^k)_{k\in\mathbb{N}^*}$ be a random i.i.d. sequence on $2^\mathcal{J}$ such that Condition (8) *holds. Then, for any deterministic initial value x_0, the iterated sequence*

$$x^{k+1} = x^k + \eta_k(\hat{\mathsf{T}}^{(\xi^{k+1})}x^k - x^k) \qquad (9)$$

converges almost surely to a random variable supported by $\mathrm{fix}(T)$.

Remark 2 *At the time of the writing the paper, the work [12] was brought to our knowledge. A result similar to Theorem 3 is presented in the framework of Hilbert spaces, random summable errors (dealt with by relying on the notion of quasi-Féjer monotonicity) and multiple blocks. The proof of [12] devoted to this result relies on the same idea as the one developed in [11] and presented above. Distributed asynchronous implementations are not considered in [12].*

By Lemma 2, ADMM+ iterates are generated by the action of an α-averaged operator. Theorem 3 shows then that a stochastic coordinate descent version of any α-averaged operator converges towards a primal-dual point. In Theorem 5 below, we apply this result to the operator related to ADMM+,

and develop an asynchronous version of ADMM+ in the context where it is distributed on a graph.

IV. Distributed Optimization

Consider a set of $N > 1$ computing agents that cooperate to solve the minimization problem

$$\inf_{x\in\mathcal{X}} \sum_{n=1}^{N} (f_n(x) + g_n(x)) \tag{10}$$

where f_n and g_n are two private functions available at Agent n. We make here the following assumption:

Assumption 4 *For each $n = 1, ..., N$,*

(i) f_n is a convex differentiable function on $\mathcal{X}$, and its gradient ∇f_n is $\bar{L}$-Lipschitz continuous on $\mathcal{X}$ for some $\bar{L} \geq 0$.
(ii) $g_n \in \Gamma_0(\mathcal{X})$.
(iii) The infimum of Problem (10) is attained.
(iv) $\cap_{n=1}^{N} \operatorname{ri}\operatorname{dom} g_n \neq \emptyset$.

Our purpose is to design a random distributed (or decentralized) iterative algorithm where, at a each iteration, each active agent updates a local estimate in the parameter space $\mathcal{X}$ based on the sole knowledge of its private functions and on information it received from its neighbors through some communication network. Eventually, the local estimates will converge to a common consensus value which is a minimizer of the aggregate function of Problem (10).

Instances of this problem appear in learning applications where massive training data sets are distributed over a network and processed by distinct machines [33], [34], in resource allocation problems for communication networks [21], or in statistical estimation problems by sensor networks [20], [35].

A. Network Model and Problem Formulation

To help the reader, the notations that will be introduced progressively are summarized in the following table.

d_n	:	Degree of node (agent) n,
$\epsilon = \{n, m\}$	:	Graph edge between n and m,
E	:	Set of graph edges,
$f(x) = \sum f_n(x_n)$	:	Differentiable term in obj. fct.,
$g(x) = \sum g_n(x_n)$	:	Other Γ_0 term in obj. fct.,
h	:	Consensus ensuring function,
λ^k	:	$= ((\lambda_\epsilon^k(n), \lambda_\epsilon^k(m)))_{\epsilon=\{n,m\}\in E}$ $\mathcal{X}^{2\lvert E\rvert}$ vector of dual variables, $\lambda_\epsilon^k(n)$ is updated by Agent n,
$n \sim m$	:	Stands for $\{n, m\} \in E$,
Subscript n	:	Agent number,
Superscript k	:	Time index,
$V = \{1, \ldots, N\}$	:	Set of graph nodes (agents),
$x^k = (x_n^k)_{n\in V}$	:	$\mathcal{X}^N$ vector of primal variables, updated by Eq. (4d),
$z^k = ((\bar{z}_\epsilon^k, \tilde{z}_\epsilon^k))_{\epsilon\in E}$	:	$\mathcal{X}^{2\lvert E\rvert}$ vector given by Eq. (4a).

We represent the network as an undirected graph $G = (V, E)$ where $V = \{1, \ldots, N\}$ is the set of agents/nodes and E is the set of edges. Representing an edge by a set $\{n, m\}$ with $n, m \in V$, we write $m \sim n$ whenever $\{n, m\} \in E$. Practically, $n \sim m$ means that Agents n and m can communicate with each other.

Assumption 5 *G is connected and has no self-loops ($n \neq m$ for all $\{n, m\} \in E$).*

Let us introduce some notation. For any $x \in \mathcal{X}^N$, we denote by x_n the n^{th} component of x, *i.e.*, $x = (x_n)_{n\in V}$. We introduce the functions f and g on $\mathcal{X}^N \to (-\infty, +\infty]$ as $f(x) = \sum_{n\in V} f_n(x_n)$ and $g(x) = \sum_{n\in V} g_n(x_n)$. Clearly, Problem (10) is equivalent to the minimization of $f(x)+g(x)$ under the constraint that all components of x are equal. Here, one can rephrase the optimization problem (10) as

$$\min_{x\in\mathcal{X}^N} \sum_{n=1}^{N} (f_n(x_n) + g_n(x_n)) + \iota_{\mathcal{C}}(x)$$

where ι_A is the indicator function of a set A (null on A and equal to $+\infty$ outside this set), and $\mathcal{C}$ is the space of vectors $x \in \mathcal{X}^N$ such that $x_1 = \cdots = x_N$. This problem is an instance of Problem (1) where $h = \iota_{\mathcal{C}}$ and M as the identity operator. However, simply setting $h = \iota_{\mathcal{C}}$ and M as the identity would not lead to a distributed algorithm. Loosely speaking, we must define h and M in such a way that it encodes the communication graph. Our goal will be to ensure global consensus through local consensus over every edge of the graph.

For any $\epsilon \in E$, say $\epsilon = \{n, m\}$, we define the linear operator $M_\epsilon : \mathcal{X}^N \to \mathcal{X}^2$ as $M_\epsilon(x) = (x_n, x_m)$ assuming $n < m$ to avoid any ambiguity on the definition of M. We construct the linear operator $M : \mathcal{X}^N \to \mathcal{Y} \triangleq \mathcal{X}^{2|E|}$ as $Mx = (M_\epsilon(x))_{\epsilon\in E}$ where we assume some (irrelevant) ordering on the edges. Any vector $y \in \mathcal{Y}$ will be written as $y = (y_\epsilon)_{\epsilon\in E}$ where, writing $\epsilon = \{n, m\} \in E$, the component y_ϵ will be represented by the couple $y_\epsilon = (y_\epsilon(n), y_\epsilon(m))$ with $n < m$. Note that this notation is abusive since it tends to indicate that y_ϵ has more than two components. However, it will turn out to be convenient in the sequel. We also introduce the subspace of $\mathcal{X}^2$ defined as $\mathcal{C}_2 = \{(x, x) : x \in \mathcal{X}\}$. Finally, we define $h : \mathcal{Y} \to (-\infty, +\infty]$ as

$$h(y) = \sum_{\epsilon\in E} \iota_{\mathcal{C}_2}(y_\epsilon)\,. \tag{11}$$

We consider the following problem:

$$\min_{x\in\mathcal{X}^N} f(x) + g(x) + h(Mx)\ . \tag{12}$$

Lemma 3 *Let Assumption 5 hold true. The minimizers of* (12) *are the tuples $(x^\star, \cdots, x^\star)$ where $x^\star$ is any minimizer of* (10)*.*

Proof: Assume that Problem (12) has a minimizer $\underline{x} = (x_1, \ldots, x_N)$. Then

$$h(M\underline{x}) = \sum_{\epsilon=\{n,m\}\in E} \iota_{\mathcal{C}_2}((x_n, x_m)) = 0.$$

Since the graph G is connected, this equation is satisfied if and only if $\underline{x} = (x^\star, \ldots, x^\star)$ for some $x^\star \in \mathcal{X}$. The result follows. ∎

B. Instantiating ADMM+

We now apply ADMM+ to solve the problem (12). Since the newly defined function h is separable with respect to the $(y_\epsilon)_{\epsilon\in E}$, we get

$$\mathrm{prox}_{\rho h}(y) = (\mathrm{prox}_{\rho \iota_{\mathcal{C}_2}}(y_\epsilon))_{\epsilon\in E} = \Big((\bar{y}_\epsilon, \bar{y}_\epsilon)\Big)_{\epsilon\in E}$$

where $\bar{y}_\epsilon = (y_\epsilon(n) + y_\epsilon(m))/2$ if $\epsilon = \{n,m\}$. With this at hand, the update equation (4a) of ADMM+ is written as $z^{k+1} = ((\bar{z}_\epsilon^{k+1}, \bar{z}_\epsilon^{k+1}))_{\epsilon\in E}$ where $\bar{z}_\epsilon^{k+1} = (x_n^k + x_m^k)/2 + \rho(\lambda_\epsilon^k(n) + \lambda_\epsilon^k(m))/2$ for any $\epsilon = \{n,m\} \in E$. Plugging this equality into Eq. (4b), it can be seen that $\lambda_\epsilon^k(n) = -\lambda_\epsilon^k(m)$. Therefore, $\bar{z}_\epsilon^{k+1} = (x_n^k + x_m^k)/2$ for any $k \geq 1$. Moreover, $\lambda_\epsilon^{k+1}(n) = \lambda_\epsilon^k(n) + (x_n^k - x_m^k)/(2\rho)$.
Let us now instantiate Equations (4c) and (4d). Observe that the n^{th} component of the vector M^*Mx coincides with $d_n x_n$ where d_n is the degree (*i.e.*, the number of neighbors) of node n. From Eq. (4d), the n^{th} component of x^{k+1} is written

$$x_n^{k+1} = \mathrm{prox}_{\tau g_n/d_n}\Big[\frac{(M^*(u^{k+1} - \tau\lambda^{k+1}))_n - \tau\nabla f_n(x_n^k)}{d_n}\Big]$$

where for any $y \in \mathcal{Y}$,

$$(M^*y)_n = \sum_{m:\{n,m\}\in E} y_{\{n,m\}}(n)$$

is the n^{th} component of $M^*y \in \mathcal{X}^N$. Plugging Eq. (4c) together with the expressions of $\bar{z}_{\{n,m\}}^{k+1}$ and $\lambda_{\{n,m\}}^{k+1}(n)$ in the argument of $\mathrm{prox}_{\tau g_n/d_n}$, we get after a small calculation

$$x_n^{k+1} = \mathrm{prox}_{\tau g_n/d_n}\Big[(1-\tau\rho^{-1})x_n^k - \frac{\tau}{d_n}\nabla f_n(x_n^k) + \frac{\tau}{d_n}\sum_{m:\{n,m\}\in E}(\rho^{-1}x_m^k - \lambda_{\{n,m\}}^k(n))\Big].$$

The *Distributed ADMM+* (DADMM+) algorithm is described by the following procedure:

DADMM+
Initialization: (x^0, λ^0) s.t. $\lambda_{\{n,m\}}^0(n) = -\lambda_{\{n,m\}}^0(m)$ for all $m \sim n$.
Do

- For all $n \in V$, Agent n has in its memory the variables x_n^k, $\{\lambda_{\{n,m\}}^k(n)\}_{m\sim n}$, and $\{x_m^k\}_{m\sim n}$. It performs the following operations:
  - For all $m \sim n$, do
$$\lambda_{\{n,m\}}^{k+1}(n) = \lambda_{\{n,m\}}^k(n) + \frac{x_n^k - x_m^k}{2\rho},$$
  - $x_n^{k+1} = \mathrm{prox}_{\tau g_n/d_n}\Big[(1-\tau\rho^{-1})x_n^k - \frac{\tau}{d_n}\nabla f_n(x_n^k) + \frac{\tau}{d_n}\sum_{m:\{n,m\}\in E}(\rho^{-1}x_m^k - \lambda_{\{n,m\}}^k(n))\Big]$.
- For all $n \in V$, Agent n sends the parameter x_n^{k+1} to its neighbors,
- Increment k.

The proof of the following result is provided in Appendix C.

Theorem 4 *Let Assumptions 4 and 5 hold true. Assume that*

$$\tau^{-1} - \rho^{-1} > \frac{\bar{L}}{2d_{\min}} \tag{13}$$

where $d_{\min}$ is the minimum of the nodes' degrees in the graph G. For any initial value (x^0, λ^0), let $(x^k)_{k\in\mathbb{N}}$ be the sequence produced by the Distributed ADMM+. Then there exists a minimizer $x^\star$ of Problem (10) *such that for all $n \in V$, $(x_n^k)_{k\in\mathbb{N}}$ converges to $x^\star$.*

C. A Distributed Asynchronous Primal Dual Algorithm

In the distributed *synchronous* case, at each clock tick, a central scheduler activates all the nodes of the network simultaneously and monitors the communications that take place between these nodes once they have finished their prox$(\cdot)$ and gradient operations. The meaning we give to "distributed asynchronous algorithm" is that there is no central scheduler and that any node can wake up randomly at any moment independently of the other nodes. This mode of operation brings clear advantages in terms of complexity and flexibility.

The proposed *Distributed Asynchronous Primal Dual* algorithm (DAPD) is obtained by applying the randomized coordinate descent on the above algorithm. As opposed to the latter, the resulting algorithm has the following attractive property: at each iteration, a single agent, or possibly a *subset* of agents chosen at random, are activated. More formally, let $(\xi^k)_{k\in\mathbb{N}}$ be a sequence of i.i.d. random variables valued in 2^V. The value taken by ξ^k represents the agents that will be activated and perform a prox on their x variable at moment k. The asynchronous algorithm goes as follows:

DAPD Algorithm:
Initialization: (x^0, λ^0).
Do

- Select a random set of agents $\xi^{k+1} = \mathcal{A}$,
- For all $n \in \mathcal{A}$, Agent n performs the following operations:
  - For all $m \sim n$, do
$$\lambda_{\{n,m\}}^{k+1}(n) = \frac{\lambda_{\{n,m\}}^k(n) - \lambda_{\{n,m\}}^k(m)}{2} + \frac{x_n^k - x_m^k}{2\rho},$$
  - $x_n^{k+1} = \mathrm{prox}_{\tau g_n/d_n}\Big[(1-\tau\rho^{-1})x_n^k - \frac{\tau}{d_n}\nabla f_n(x_n^k) + \frac{\tau}{d_n}\sum_{m\sim n}(\rho^{-1}x_m^k + \lambda_{\{n,m\}}^k(m))\Big]$,
  - For all $m \sim n$, send $\{x_n^{k+1}, \lambda_{\{n,m\}}^{k+1}(n)\}$ to Neighbor m.
- For all agents $n \notin \mathcal{A}$, $x_n^{k+1} = x_n^k$, and $\lambda_{\{n,m\}}^{k+1}(n) = \lambda_{\{n,m\}}^k(n)$ for all $m \sim n$.
- Increment k.

Assumption 6 *The collections of sets $\{\mathcal{A}_1, \mathcal{A}_2, \ldots\}$ such that $\mathbb{P}[\xi^1 = \mathcal{A}_i]$ is positive satisfies $\bigcup \mathcal{A}_i = V$.*

In other words, any agent is selected with a positive probability. The following theorem is proven in Appendix D.

Theorem 5 *Let Assumptions 4, 5, and 6 hold true. Assume that condition (13) holds true. Let $(x_n^{k+1})_{n\in V}$ be the output of the DAPD algorithm. For any initial value (x^0, λ^0), the sequences $x_1^k, \ldots, x_N^k$ converge almost surely as $k \to \infty$ to a random variable $x^\star$ supported by the set of minimizers of Problem* (10).

Before turning to the numerical illustrations, we note that the very recent paper [36] also deals with asynchronous primal-dual distributed algorithms by relying on the idea of random coordinate descent.

V. Numerical illustrations

We address the problem of the so called ℓ_2-regularized logistic regression. Denoting by m the number of observations and by p the number of features, our optimization problem is written

$$\min_{\mathbf{x}\in\mathbb{R}^p} \frac{1}{m}\sum_{t=1}^{m} \log\left(1+\mathrm{e}^{-y_t\mathbf{a}_t^{\mathrm{T}}\mathbf{x}}\right) + \mu\|\mathbf{x}\|^2$$

where the $(y_t)_{t=1}^m$ are in $\{-1,+1\}$, the $(\mathbf{a}_t)_{t=1}^m$ are in $\mathbb{R}^p$, and $\mu>0$ is a scalar.
We consider the case where the dataset is scattered over a network. Indeed, massive data sets are often distributed on different physical machines communicating together by means of an interconnection network [37, Chap. 2.5] and many algorithms have been implemented for independent threads or processes running on distant cores, closer to the data (see *e.g.* [24], [13] for MapReduce implementation of ADMM, [26] for Spark implementation). Formally, denoting by $\{\mathcal{B}_n\}_{n=1}^N$ a partition of $\{1,\ldots,m\}$, we assume that Agent n holds in its memory the data in $\mathcal{B}_n$. Denoting by $G=(V,E)$ the graph that represents the connections between the agents, the regularized logistic regression problem is written in its distributed form as

$$\min_{\mathbf{x}\in\mathbb{R}^{Np}} \sum_{n=1}^{N}\left(\sum_{t\in\mathcal{B}_n}\frac{1}{m}\log\left(1+\mathrm{e}^{-y_t\mathbf{a}_t^{\mathrm{T}}\mathbf{x}_n}\right)+\frac{\mu}{2N}\|\mathbf{x}_n\|_2^2\right) + \sum_{\epsilon\in E}\iota_{\mathcal{C}_2}(y_\epsilon).$$

Clearly, this is an instance of Problem (12).

Our simulations will be performed on the following classical datasets:

name	m	p	density
`covtype`	581012	54	dense
`alpha`	500000	500	dense
`realsim`	72309	20958	sparse
`rcv1`	20242	47236	sparse

The datasets `covtype`, `realsim`, and `rcv1` are taken from the LIBSVM website[2] and `alpha` was from the Pascal 2008 Large Scale Learning challenge[3]. We preprocessed the dense datasets so that each feature has zero mean and unit variance. The global Lipschitz constant for the gradient of the logistic function was estimated by its classical upper bound $\hat{L}=0.25\max_{n=1,\ldots,N}\|\mathbf{a}_n\|_2^2$. Finally, the regularization parameter μ was set to 10^{-4}.

In our simulations, we also compared the DAPD algorithm presented in this paper with some known algorithms that lend themselves to a distributed implementation. These are:

- DGD: the synchronous distributed algorithm [38]. Here, each agent performs a gradient descent then exchanges with its neighbors according to the Metropolis rule.
- ABG: the asynchronous broadcast gradient [39]. In this setup, one agent wakes up and sends its information to its neighbors. Any of these neighbors replaces its current value with the mean of this value and the received value then performs a gradient descent.
- PWG: the pairwise gossip gradient [16], [40]. In this setup, one agent wakes up, and selects one neighbor. Then each of the two agents performs a gradient descent, then exchanges and replaces its value by taking the mean between the former and the received value.

For the DGD, the ABG, and the PWG, the stepsizes have been taken decreasing as $\gamma_0/k^{0.75}$. The other parameters (including γ_0) were chosen automatically in sets of the form $parameter_{theory}\times 10^i, i=\{1,..,10\}$ (where $parameter_{theory}$ is computed from the best theoretic bound with Lipschitz constant estimate $\hat{L}$) by running in parallel multiple instances of the algorithm with the different constants over 50 iterations and choosing the constant giving the lowest functional cost.

Whereas DAPD can allow for multiple agents to wake up at each iteration, we considered only the single active agent case as it does not change much the practical implementation. It is thus underperforming compared to a multiple awaking agents scenario. Similarly to the previous algorithms, the stepsizes of DAPD have been chosen automatically in sets of the form $parameter_{theory}\times 10^i$ for τ and $\rho=2\tau$ for fairness in terms of number of step sizes explored.

The (total) functional cost was evaluated with the value at agent 1 (the agents are indistinguishable from a network point of view) and plotted versus the number of local gradients used.

In Figure 1, we plot the ℓ_2-regularized logistic cost at some agent versus the number of local gradients used. We solved this problem for each dataset on a 10×10 2D toroidal grid (100 agents) by assigning the same number of observations per agent. We observe that the DAPD is significantly faster than the other stochastic gradient methods. Finally, we also remark that the quantity of information exchanged per iteration for DAPD is roughly a vector of length shorter than $2Np$ ($8p$ with our graph) which means that the number of transmissions is in general quite small compared to the size of the whole dataset (roughly Tp).

In Figure 2, we plot the same quantities for the `rcv1` dataset but now the same number of observations are dispatched over i) a 5×5 toroidal grid (25 agents) and ii) a 50-nodes complete network.

VI. Conclusions and Perspectives

This paper introduced a general framework for stochastic coordinate descent. The framework was used on a new algorithm called ADMM+ which has roots in a recent work by

[2] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`
[3] `http://largescale.ml.tu-berlin.de`

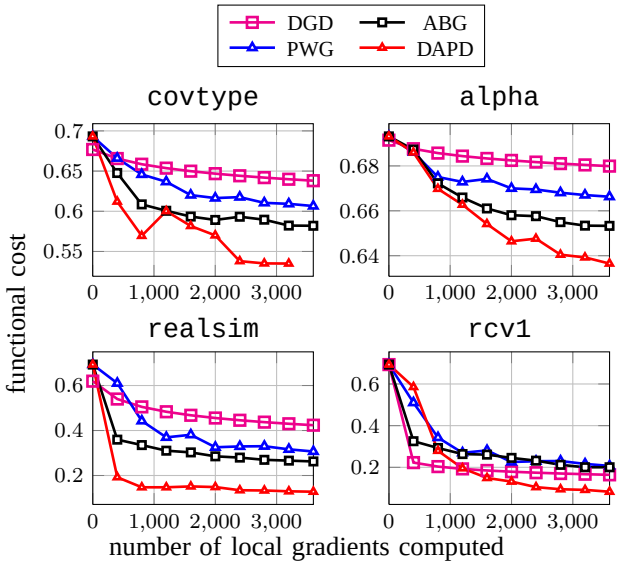

Fig. 1. Comparison of distributed algorithms on a 5×5 grid.

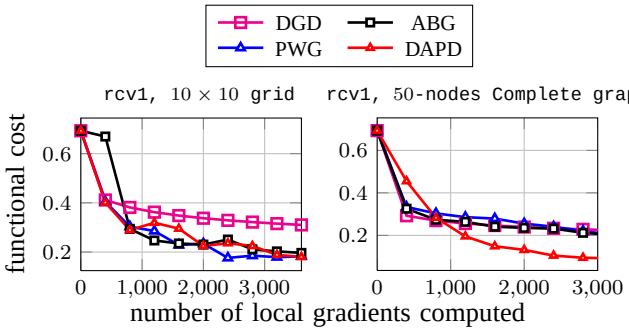

Fig. 2. Comparison between different networks on `rcv1` dataset.

Vũ and Condat. As a byproduct, we obtained an asynchronous distributed algorithm which enables the processing of distinct blocks on different machines. Future works include an analysis of the convergence rate of our algorithms along with efficient stepsizes strategies.

APPENDIX

A. Proof of Theorem 1

By setting $\mathcal{E} = \mathcal{S}$ and by assuming that $\mathcal{E}$ is equipped with the same inner product as $\mathcal{Y}$, one can notice that the functions $\bar{f} = f \circ M^{-1}$, $\bar{g} = g \circ M^{-1}$ and h satisfy the conditions of Theorem 2. Moreover, since $(\bar{f} + \bar{g})^* = (f + g)^* \circ M^*$, one can also notice that $(x^\star, \lambda^\star)$ is a primal-dual point associated with Eq. (3) if and only if $(Mx^\star, \lambda^\star)$ is a primal-dual point associated with Eq. (7).

To recover ADMM+ from the iterations (6a)–(6b), the starting point is Moreau's identity [8, Th. 14.3] which reads

$$\operatorname{prox}_{\rho^{-1}h^*}(x) + \rho^{-1} \operatorname{prox}_{\rho h}(\rho x) = x\,.$$

Setting $x^k = M^{-1}y^k$ and

$$\begin{aligned} z^{k+1} &= \operatorname{prox}_{\rho h}(y^k + \rho\lambda^k) \\ &= \operatorname*{argmin}_{w \in \mathcal{Y}} \Big[h(w) + \frac{\|w - (Mx^k + \rho\lambda^k)\|^2}{2\rho} \Big]\,, \end{aligned}$$

Equation (6a) can be rewritten thanks to Moreau's identity

$$\lambda^{k+1} = \lambda^k + \rho^{-1}(Mx^k - z^{k+1})\,.$$

Now, Equation (6b) can be rewritten as

$$y^{k+1} = \operatorname*{argmin}_{w \in \mathcal{S}} \bar{g}(w) + \langle \nabla \bar{f}(y^k), w \rangle + \frac{\|w - y^k + \tau(2\lambda^{k+1} - \lambda^k)\|^2}{2\tau}$$

Upon noting that $\bar{g}(Mx) = g(x)$ and $\langle \nabla \bar{f}(y^k), Mx \rangle = \langle (M^{-1})^* \nabla f(M^{-1}Mx^k), Mx \rangle = \langle \nabla f(x^k), x \rangle$, the above equation becomes

$$x^{k+1} = \operatorname*{argmin}_{w \in \mathcal{X}} g(w) + \langle \nabla f(x^k), w \rangle + \frac{\|Mw - u^{k+1} + \tau\lambda^{k+1}\|^2}{2\tau}$$

where

$$\begin{aligned} u^{k+1} &= Mx^k + \tau(\lambda^k - \lambda^{k+1}) \\ &= (1 - \rho^{-1}\tau)Mx^k + \tau\rho^{-1}z^{k+1}. \end{aligned}$$

The iterates $(z^{k+1}, \lambda^{k+1}, u^{k+1}, x^{k+1})$ are those of the ADMM+.

B. Proof of Theorem 3

The main idea behind the proof can be found in [11]. Define the operator $\mathsf{U} = (1 - \eta_k)\mathsf{I} + \eta_k \mathsf{T}$ (we omit the index k in U to simplify the notation); similarly, define $\mathsf{U}^{(\kappa)} = (1 - \eta_k)\mathsf{I} + \eta_k \hat{\mathsf{T}}^{(\kappa)}$. Remark that the operator U is $(\alpha\eta_k)$-averaged.

The iteration (9) reads $x^{k+1} = \mathsf{U}^{(\xi^{k+1})}x^k$. Set $p_\kappa = \mathbb{P}(\xi_1 = \kappa)$ for any $\kappa \in 2^{\mathcal{J}}$. Denote by $\|x\|^2 = \langle x, x \rangle$ the squared norm in $\mathcal{H}$. Define a new inner product $x \bullet y = \sum_{j=1}^J q_j \langle x_j, y_j \rangle_j$ on $\mathcal{H}$ where $q_j^{-1} = \sum_{\kappa \in 2^{\mathcal{J}}} p_\kappa \mathbf{1}_{\{j \in \kappa\}}$ and let $|||x|||^2 = x \bullet x$ be its associated squared norm. Consider any $x^\star \in \operatorname{fix}(\mathsf{T})$. Conditionally to the sigma-field $\mathcal{F}^k = \sigma(\xi_1, \dots, \xi^k)$ we have

$$\begin{aligned} \mathbb{E}[|||x^{k+1} - x^\star|||^2 \mid \mathcal{F}^k] &= \sum_{\kappa \in 2^{\mathcal{J}}} p_\kappa \left|\left|\left| \hat{\mathsf{U}}^{(\kappa)}x^k - x^\star \right|\right|\right|^2 \\ &= \sum_{\kappa \in 2^{\mathcal{J}}} p_\kappa \sum_{j \in \kappa} q_j \|\mathsf{U}_j x^k - x_j^\star\|^2 + \sum_{\kappa \in 2^{\mathcal{J}}} p_\kappa \sum_{j \notin \kappa} q_j \|x_j^k - x_j^\star\|^2 \\ &= |||x^k - x^\star|||^2 + \sum_{\kappa \in 2^{\mathcal{J}}} p_\kappa \sum_{j \in \kappa} q_j \left(\|\mathsf{U}_j x^k - x_j^\star\|^2 - \|x_j^k - x_j^\star\|^2 \right) \\ &= |||x^k - x^\star|||^2 + \sum_{j=1}^J \left(\|\mathsf{U}_j x^k - x_j^\star\|^2 - \|x_j^k - x_j^\star\|^2 \right) \\ &= |||x^k - x^\star|||^2 + \left(\|\mathsf{U}x^k - x^\star\|^2 - \|x^k - x^\star\|^2 \right)\,. \end{aligned}$$

Using that U is $(\alpha\eta_k)$-averaged and that $x^\star$ is a fixed point of U, the term enclosed in the parentheses is no larger than $-\frac{1 - \alpha\eta_k}{\alpha\eta_k} \|(\mathsf{I} - \mathsf{U})x^k\|^2$. As $\mathsf{I} - \mathsf{U} = \eta_k(\mathsf{I} - \mathsf{T})$, we obtain:

$$\begin{aligned} \mathbb{E}[|||x^{k+1} - x^\star|||^2 \mid \mathcal{F}^k] \leq |||x^k - x^\star|||^2 \\ - \eta_k(1 - \alpha\eta_k)\|(\mathsf{I} - \mathsf{T})x^k\|^2 \end{aligned} \quad (14)$$

which shows that $|||x^k - x^\star|||^2$ is a nonnegative supermartingale with respect to the filtration $(\mathcal{F}_k)$. As such, it converges with probability one towards a random variable that is finite almost everywhere.

Given a countable dense subset H of $\operatorname{fix}(\mathsf{T})$, there is a probability one set on which $|||x^k - \boldsymbol{x}||| \to X_{\boldsymbol{x}} \in [0, \infty)$ for

all $\boldsymbol{x} \in H$. Let $x^\star \in \mathrm{fix}(\mathsf{T})$, let $\varepsilon > 0$, and choose $\boldsymbol{x} \in H$ such that $|||x^\star - \boldsymbol{x}||| \leq \varepsilon$. With probability one, we have

$$|||x^k - x^\star||| \leq |||x^k - \boldsymbol{x}||| + |||\boldsymbol{x} - x^\star||| \leq X_{\boldsymbol{x}} + 2\varepsilon$$

for k large enough. Similarly, $|||x^k - x^\star||| \geq X_{\boldsymbol{x}} - 2\varepsilon$ for k large enough. We therefore obtain:

C1 : There is a probability one set on which $|||x^k - x^\star|||$ converges for every $x^\star \in \mathrm{fix}(\mathsf{T})$.

Getting back to (14), taking the expectations on both sides of this inequality and iterating over k, we obtain

$$\sum_{k=0}^{\infty} \eta_k(1-\alpha\eta_k)\mathbb{E}\|(\mathsf{I}-\mathsf{T})x^k\|^2 \leq |||x^0 - x^\star|||^2 .$$

Using the assumption on $(\eta_k)_{k\in\mathbb{N}}$, it is straightforward to see that $\sum_{k=0}^{\infty} \eta_k(1-\alpha\eta_k) = +\infty$ and thus that $\sum_{k=0}^{\infty} \mathbb{E}\|(\mathsf{I}-\mathsf{T})x^k\|^2$ is finite. By Markov's inequality and Borel Cantelli's lemma, we therefore obtain:

C2 : $(\mathsf{I}-\mathsf{T})x^k \to 0$ almost surely.

We now consider an elementary event in the probability one set where **C1** and **C2** hold. On this event, since $|||x^k - x^\star|||$ converges for $x^\star \in \mathrm{fix}(\mathsf{T})$, the sequence $(x^k)_{k\in\mathbb{N}}$ is bounded. Since T is α-averaged, it is continuous, and **C2** shows that all the accumulation points of $(x^k)_{k\in\mathbb{N}}$ are in $\mathrm{fix}(\mathsf{T})$. It remains to show that these accumulation points reduce to one point. Assume that $x_1^\star$ is an accumulation point. By **C1**, $|||x^k - x_1^\star|||$ converges. Therefore, $\lim |||x^k - x_1^\star||| = \liminf |||x^k - x_1^\star||| = 0$, which shows that $x_1^\star$ is unique.

C. Proof of Theorem 4

The proof simply consists in checking that the assumptions of Theorem 1 are satisfied. To that end, we compute the Lipschitz constant L of $\nabla(f\circ M^{-1})$ as a function of $\bar{L}$. Recall that $\mathcal{S}$ is the image of M. For any $y \in \mathcal{S}$, note that

$$\nabla(f\circ M^{-1})(y) = M(M^*M)^{-1}\nabla f(M^{-1}y)\,. \tag{15}$$

Using the definition of M, the operator M^*M is diagonal. More precisely, for any $x \in \mathbb{R}^N$, say $x = (x_n)_{n\in V}$, the nth component of $(M^*M)x$ coincides with $d_n x_n$ where $d_n = \mathrm{card}\{m \in V \,:\, n \sim m\}$ is the degree of node n in the graph G. Thus, $\|M(M^*M)^{-1}x\|^2 = \sum_{n\in V} d_n^{-1}\|x_n\|^2$. As a consequence of the latter equality and (15), for any $(y, y') \in \mathcal{S}^2$, say $y = Mx$ and $y' = Mx'$, one has

$$\begin{aligned}\|\nabla(f\circ M^{-1})(y) - \nabla(f\circ M^{-1})(y')\|^2 \\ = \sum_n d_n^{-1}\|\nabla f_n(x_n) - \nabla f_n(x_n')\|^2\,.\end{aligned}$$

Under the stated hypotheses, we can write for all n, $\|\nabla f_n(x_n) - \nabla f_n(x_n')\|^2 \leq \bar{L}^2\|x_n - x_n'\|^2$. Thus,

$$\|\nabla(f\circ M^{-1})(y) - \nabla(f\circ M^{-1})(y')\|^2 \leq (\bar{L}^2/d_{\min})\|x - x'\|^2 \tag{16}$$

where $d_{\min} = \min(d_n : n \in V)$. On the otherhand, $\|y - y'\|^2 = \|M(x-x')\|^2 = \sum_n d_n\|x_n - x_n'\|^2 \geq d_{\min}\|x - x'\|^2$. Plugging the latter inequality into (16), we finally obtain $\|\nabla(f\circ M^{-1})(y) - \nabla(f\circ M^{-1})(y')\|^2 \leq (\bar{L}/d_{\min})^2\|x - x'\|^2$. This proves that $\nabla(f\circ M)$ is Lipschitz continuous with constant $L = \bar{L}/d_{\min}$. The final result follows by immediate application of Theorem 1.

D. Proof of Theorem 5

Let $(\bar{f}, \bar{g}, h) = (f\circ M^{-1}, g\circ M^{-1}, h)$ where f, g, h and M are those of Problem (12). For these functions, write Equations (6) as $(\lambda^{k+1}, y^{k+1}) = \mathsf{T}(\lambda^k, y^k)$. By Lemma 2, the operator T is an α-averaged operator acting on the space $\mathcal{H} = \mathcal{Y}\times\mathcal{S}$, where $\mathcal{S}$ is the image of $\mathcal{X}^N$ by M. For any $n \in V$, let S_n be the selection operator on $\mathcal{H}$ defined as $S_n(\lambda, Mx) = ((\lambda_\epsilon(n))_{\epsilon\in E\,:\, n\in\epsilon}, x_n)$. Then it is easy to see that up to an element reordering, $\mathcal{H} = S_1(\mathcal{H})\times\cdots\times S_N(\mathcal{H})$. Identifying the set $\mathcal{J}$ introduced before the statement of Theorem 3 with V, the operator $\mathsf{T}^{(\xi^k)}$ is defined as follows: if $n \in \xi^k$, then $S_n(\mathsf{T}^{(\xi^k)}(\lambda, Mx)) = S_n(\mathsf{T}(\lambda, Mx))$ while if $n \notin \xi^k$, then $S_n(\mathsf{T}^{(\xi^k)}(\lambda, Mx)) = S_n(\lambda, Mx)$. We know by Theorem 3 that the sequence $(\lambda^{k+1}, Mx^{k+1}) = \mathsf{T}^{(\xi^{k+1})}(\lambda^k, Mx^k)$ converges almost surely to a primal-dual point of Problem (7). This implies by Lemma 3 that the sequence x^k converges almost surely to $(x^\star, \ldots, x^\star)$ where $x^\star$ is a minimizer of Problem (10).

We therefore need to prove that the operator $\mathsf{T}^{(\xi^{k+1})}$ is translated into the DAPD algorithm. The definition (11) of h shows that

$$h^*(\phi) = \sum_{\epsilon\in E} \iota_{\mathcal{C}_2^\perp}(\phi_\epsilon)$$

where $\mathcal{C}_2^\perp = \{(x, -x) \,:\, x \in \mathcal{X}\}$. Therefore, writing

$$(\eta^{k+1}, q^{k+1} = Mv^{k+1}) = \mathsf{T}(\lambda^k, y^k = Mx^k),$$

Equation (6a) shows that

$$\eta_\epsilon^{k+1} = \mathrm{proj}_{\mathcal{C}_2^\perp}(\lambda_\epsilon^k + \rho^{-1}y_\epsilon^k).$$

Notice that contrary to the case of the synchronous algorithm DADMM+, there is no reason here for which $\mathrm{proj}_{\mathcal{C}_2^\perp}(\lambda_\epsilon^k) = 0$. Getting back to $(\lambda^{k+1}, Mx^{k+1}) = \mathsf{T}^{(\xi^{k+1})}(\lambda^k, y^k = Mx^k)$, we therefore obtain that for all $n \in \xi^{k+1}$ and all $m \sim n$,

$$\begin{aligned}\lambda_{\{n,m\}}^{k+1}(n) &= \frac{\lambda_{\{n,m\}}^k(n) - \lambda_{\{n,m\}}^k(m)}{2} \\ &\quad + \frac{y_{\{n,m\}}^k(n) - y_{\{n,m\}}^k(m)}{2\rho} \\ &= \frac{\lambda_{\{n,m\}}^k(n) - \lambda_{\{n,m\}}^k(m)}{2} + \frac{x_n^k - x_m^k}{2\rho}.\end{aligned}$$

Recall now that Eq. (6b) can be rewritten as

$$q^{k+1} = \operatorname*{argmin}_{w\in\mathcal{S}} \bar{g}(w) + \langle\nabla\bar{f}(y^k), w\rangle + \frac{\|w - y^k + \tau(2\lambda^{k+1} - \lambda^k)\|^2}{2\tau}$$

Upon noting that $\bar{g}(Mx) = g(x)$ and $\langle\nabla\bar{f}(y^k), Mx\rangle = \langle(M^{-1})^*\nabla f(M^{-1}Mx^k), Mx\rangle = \langle\nabla f(x^k), x\rangle$, the above equation becomes

$$\begin{aligned}v^{k+1} = \operatorname*{argmin}_{w\in\mathcal{X}}\, & g(w) + \langle\nabla f(x^k), w\rangle \\ & + \frac{\|M(w - x^k) + \tau(2\lambda^{k+1} - \lambda^k)\|^2}{2\tau}.\end{aligned}$$

Recall that $(M^*Mx)_n = d_n x_n$. Hence, for all $n \in \xi^{k+1}$, we get after some computations

$$x_n^{k+1} = \text{prox}_{\tau g_n/d_n}\Big[x_n^k - \frac{\tau}{d_n}\nabla f_n(x_n^k) - \frac{\tau}{d_n}(M^*(2\lambda^{k+1} - \lambda^k))_n\Big].$$

Using the identity $(M^*y)_n = \sum_{m:\{n,m\}\in E} y_{\{n,m\}}(n)$, one can check that this equation coincides with the $x-$update equation in the DAPD algorithm.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their useful comments.

REFERENCES

[1] B. C. Vũ, "A splitting algorithm for dual monotone inclusions involving cocoercive operators," *Advances in Computational Mathematics*, vol. 38, no. 3, pp. 667–681, 2013.

[2] L. Condat, "A primal-dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms," *Journal of Optimization Theory and Applications*, vol. 158, no. 2, pp. 460–479, 2013.

[3] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Computers & Mathematics with Applications*, vol. 2, no. 1, pp. 17–40, 1976.

[4] D. Gabay, "Application of the method of multipliers to variational inequalities," in M. Fortin and R. Glowinski, editors, *Augmented Lagrangian Methods: Applications to the solution of Boundary-Value Problems*. North-Holland, Amsterdam, 1983.

[5] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[6] O. Fercoq and P. Richtárik, "Accelerated, parallel and proximal coordinate descent," *arXiv preprint arXiv:1312.5799*, 2013.

[7] M. Bačák, "The proximal point algorithm in metric spaces," *Israel Journal of Mathematics*, vol. 194, no. 2, pp. 689–701, 2013.

[8] H. H. Bauschke and P. L. Combettes, *Convex analysis and monotone operator theory in Hilbert spaces*, CMS Books in Mathematics/Ouvrages de Mathématiques de la SMC. Springer, New York, 2011.

[9] I.D. Schizas, A. Ribeiro, and G.B. Giannakis, "Consensus in ad hoc WSNs with noisy links - Part I: Distributed estimation of deterministic signals," *IEEE Trans. on Signal Processing*, vol. 56, no. 1, pp. 350–364, 2008.

[10] H. Ouyang, N. He, L. Tran, and A. Gray, "Stochastic Alternating Direction Method of Multipliers," in *International Conference on Machine Learning*, 2013, pp. 80–88.

[11] F. Iutzeler, P. Bianchi, Ph. Ciblat, and W. Hachem, "Asynchronous distributed optimization using a randomized Alternating Direction Method of Multipliers," in *Proc. IEEE Conf. Decision and Control (CDC)*, Florence, Italy, Dec. 2013.

[12] P. L. Combettes and J.-C. Pesquet, "Stochastic quasi-Fejér block-coordinate fixed point iterations with random sweeping," *arXiv preprint arXiv:1404.7536*, 2014.

[13] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the Alternating Direction Method of Multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[14] V. Cevher, S. Becker, and M. Schmidt, "Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics," *Signal Processing Magazine, IEEE*, vol. 31, no. 5, pp. 32–43, 2014.

[15] J. Tsitsiklis, *Problems in Decentralized Decision Making and Computation*, Ph.D. thesis, Massachusetts Institute of Technology, 1984.

[16] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *Automatic Control, IEEE Transactions on*, vol. 31, no. 9, pp. 803 – 812, sep 1986.

[17] H. J. Kushner and G. Yin, "Asymptotic properties of distributed and communicating stochastic approximation algorithms," *SIAM J. Control Optim.*, vol. 25, pp. 1266 – 1290, 1987.

[18] C. Lopes and A.H. Sayed, "Distributed processing over adaptive networks," in *Adaptive Sensor Array Processing Workshop*, June 2006, pp. 1–5.

[19] A. Nedic, A. Ozdaglar, and P.A. Parrilo, "Constrained Consensus and Optimization in Multi-Agent Networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, April 2010.

[20] P. Bianchi, G. Fort, and W. Hachem, "Performance of a distributed stochastic approximation algorithm," *IEEE Transactions on Information Theory*, vol. 59, no. 11, pp. 7405 – 7418, November 2013.

[21] P. Bianchi and J. Jakubowicz, "Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization," *IEEE Transactions on Automatic Control*, vol. 58, no. 2, pp. 391 – 405, February 2013.

[22] A. Nedic and A. Olshevsky, "Distributed optimization over time-varying directed graphs," in *IEEE conf. on Decision and Control*, Florence, Italy, 2013.

[23] F. Iutzeler, P. Bianchi, Ph. Ciblat, and W. Hachem, "Explicit convergence rate of a distributed alternating direction method of multipliers," *arXiv preprint arXiv:1312.1085*, 2013.

[24] K. Tsianos and M. Rabbat, "Efficient distributed online prediction and stochastic optimization with approximate distributed mini-batches," *arXiv preprint arXiv:1403.0603*, 2014.

[25] A. Mokhtari, Q. Ling, and A. Ribeiro, "An approximate newton method for distributed optimization," in *ICASSP (submitted)*, 2015.

[26] M. Jaggi, V. Smith, M. Takáč, J. Terhorst, T. Hofmann, and M. I. Jordan, "Communication-efficient distributed dual coordinate ascent," *arXiv preprint arXiv:1409.1458*, 2014.

[27] G. Morral, P. Bianchi, and G. Fort, "Success and failure of adaptation-diffusion algorithms for consensus in multi-agent networks," *arXiv preprint arXiv:1410.6956*, 2014.

[28] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *arXiv preprint arXiv:1404.6264*, 2014.

[29] E. Wei and A. Ozdaglar, "On the $O(1/k)$ convergence of asynchronous distributed Alternating Direction Method of Multipliers," *arXiv preprint arXiv:1307.8254*, 2013.

[30] R. T. Rockafellar, *Convex analysis*, Princeton Mathematical Series, No. 28. Princeton University Press, Princeton, N.J., 1970.

[31] B. He and X. Yuan, "Convergence analysis of primal-dual algorithms for a saddle-point problem: From contraction perspective," *SIAM Journal on Imaging Sciences*, vol. 5, no. 1, pp. 119–149, 2012.

[32] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on pure and applied mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.

[33] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *The Journal of Machine Learning Research*, vol. 99, pp. 1663–1707, 2010.

[34] A. Agarwal, O. Chapelle, M. Dudík, and J. Langford, "A reliable effective terascale linear learning system," *arXiv preprint arXiv:1110.4198*, 2011.

[35] S.S. Ram, V.V. Veeravalli, and A. Nedic, "Distributed and recursive parameter estimation in parametrized linear state-space models," *IEEE Trans. on Automatic Control*, vol. 55, no. 2, pp. 488–492, 2010.

[36] J.-C. Pesquet and A. Repetti, "A Class of Randomized Primal-Dual Algorithms for Distributed Optimization," *ArXiv e-prints*, June 2014.

[37] T. Rauber and G. Rünger, *Parallel programming: For multicore and cluster systems*, Springer Science & Business, 2013.

[38] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *Automatic Control, IEEE Transactions on*, vol. 54, no. 1, pp. 48–61, Jan 2009.

[39] A. Nedic, "Asynchronous broadcast-based convex optimization over a network," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1337–1351, 2011.

[40] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah, "Randomized gossip algorithms," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2508–2530, 2006.