A Stochastic Coordinate Descent Primal-Dual Algorithm and Applications to Large-Scale Composite Optimization

P. Bianchi, W. Hachem and F. Iutzeler

Abstract—First, we introduce a splitting algorithm to minimize a sum of three convex functions. The algorithm is of primal dual kind and is inspired by recent results of Vû and Condat. Second, we provide a randomized version of the algorithm based on the idea of coordinate descent. Third, we address two applications of our method. (i) In the case of stochastic minibatch optimization, the algorithm can be used to split a composite objective function into blocks, each of these blocks being processed sequentially by the computer. (ii) In the case of distributed optimization, we consider a set of $N$ agents having private composite objective functions and seeking to find a consensus on the minimum of the aggregate objective. In that case, our method yields a distributed iterative algorithm where each agent use both local computations and message passing in an asynchronous manner. Numerical results demonstrate the attractive performance of the method in the framework of large scale machine learning applications.

Index Terms—Distributed Optimization, Large-scale Learning, Coordinate Descent, Consensus algorithms, Primal-Dual Algorithm.

I. INTRODUCTION

Let $\mathcal{X}$ and $\mathcal{Y}$ be two Euclidean spaces and let $M : \mathcal{X} \rightarrow \mathcal{Y}$ 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

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

Our contributions are threefold.

1) Assuming that $f$ is differentiable and that its gradient is Lipschitz-continuous, we provide an iterative algorithm for solving (1). We refer to our algorithm as ADMM+ (Alternating Direction Method of Multipliers plus) because it includes the well known ADMM [1], [2] as a special case. The algorithm belongs to the class of primal-dual optimization algorithms. More specifically, it has its roots in algorithms recently developed by Vû [3] and Condat [4]. The algorithm also has some similarities with a recent variation on the ADMM by [5].

2) Based on our previous work [6], we introduce the idea of stochastic coordinate descent on Krasnosel’skii-Mann iterations. In their simplest form, Krasnosel’skii-Mann iterations can be interpreted as fixed point iterations of a given operator having a contraction-like property. Interestingly, ADMM+ as well as many other algorithms (gradient descent, proximal gradient algorithm, ADMM, etc.) are special instances of Krasnosel’skii-Mann iterations [7]. The idea beyond stochastic coordinate descent is to update only a random subset of coordinates at each iteration. This leads to a perturbed version of the initial Krasnosel’skii-Mann iterations which can nevertheless be shown to preserve the sought convergence properties of the initial unperturbed version. Stochastic coordinate descent has been recently investigated in the literature in the special case of proximal gradient algorithms [8]–[10]. We believe that its application to the broader class of Krasnosel’skii-Mann algorithms can potentially lead to various algorithms well suited to large-scale optimization problems. The next point also sustains this claim.

3) We apply our findings to large-scale optimization problems arising in signal processing and machine learning contexts. We show that the general idea of stochastic coordinate descent provides a unified framework allowing to derive stochastic algorithms of different kinds. More precisely, we derive two application examples. First, we introduce a new stochastic approximation algorithm by applying stochastic coordinate descent on the top of ADMM+. The algorithm is referred to as Stochastic Minibatch Primal Dual algorithm (SMPD). Second, we propose a new asynchronous distributed optimization algorithm that we refer to as Distributed Asynchronous Primal Dual algorithm (DAPD). The algorithm can be used to efficiently solve an optimization problem over a network of communicating agents.

The paper is organized as follows. Section II is devoted to the introduction of the ADMM+ algorithm and its relation with the Primal-Dual algorithms of Vû [3] and Condat [4], we also show how the ADMM+ includes both the standard ADMM and the Forward-Backward algorithm (also referred to as proximal gradient algorithm) as special cases. In Section III we provide our main result on the convergence of Krasnosel’skii-Mann iterations with randomized coordinate descent. This enables us to derive, in Section IV a stochastic approximation algorithm from the ADMM+. Section V addresses the problem of asynchronous distributed optimization. Finally, Section VI provides numerical results in the context of large-scale $\ell_1$-regularized logistic regression.
II. A Primal-Dual Algorithm

A. Problem statement

We consider Problem (1). Denoting by $\Gamma_0(\mathcal{X})$ the set of proper lower semicontinuous 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 $\mathcal{S} = \text{Im}(M)$ the image of $\mathcal{M}$ 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 $\nabla$ 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 $\text{dom} q$ the domain of a function $q$ and by $\text{ri} \mathcal{S}$ the relative interior of a set $\mathcal{S}$ in a Euclidean space.

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

$$0 \in \text{ri}(\text{dom} h - M \text{dom} g).$$

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

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

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 [11], [12] shows that

$$\min_{x \in \mathcal{X}} f(x) + g(x) + h(Mx) = -\inf_{\lambda \in \mathcal{Y}} (f + g)^*(-M^*\lambda) + h^*(\lambda), \quad (2)$$

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

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

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+.

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 $(x^k, \lambda^k)$ defined by ADMM+ converges to a primal-dual point $(x^*, \lambda^*)$ of (2) as $k \to \infty$.

Before providing the proof of Theorem 1, let us introduce the following notation. For any function $g \in \Gamma_0(\mathcal{X})$ we denote by $\text{prox}_g$ its proximity operator defined by

$$\text{prox}_g(x) = \arg \min_{w \in \mathcal{X}} \left[ g(w) + \frac{1}{2} \|w - x\|^2 \right]. \quad (4)$$

The ADMM+ has roots in a primal-dual algorithm recently proposed by V"u [3] and Condat [4]:

Theorem 2 ([3], [4]) Given a Euclidean space $\mathcal{E}$, consider the minimization problem $\inf_{x \in \mathcal{E}} f(y) + \tilde{g}(y) + h(y)$ where $\tilde{g}, h \in \Gamma_0(\mathcal{E})$ and where $f$ is convex and differentiable on $\mathcal{E}$ with a $L$-Lipschitz continuous gradient. Assume that the infimum is attained and that $0 \in \text{ri}(\text{dom} h - \text{dom} \tilde{g})$. Let $\tau, \rho > 0$ be such that $\tau^{-1} - \rho^{-1} > L/2$, and consider the iterates

$$\lambda^{k+1} = \text{prox}_{\rho h}(\lambda^k + \rho^{-1} y^k) \quad \text{and} \quad y^{k+1} = \text{prox}_{\tau g}(y^k - \tau \nabla \tilde{f}(y^k) - \tau(2\lambda^{k+1} - \lambda^k)). \quad (5a)$$

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

$$\inf_{y \in \mathcal{E}} \tilde{f}(y) + \tilde{g}(y) + h(y) = -\inf_{x \in \mathcal{E}} (\tilde{f} + \tilde{g})^*(-\lambda) + h^*(\lambda). \quad (6)$$

Elaborating on Theorem 1 we are now ready to establish the Theorem 1.

C. Proof of Theorem 7

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 $\tilde{f} = f \circ M^{-1}$, $\tilde{g} = g \circ M^{-1}$ and $h$ satisfy the conditions of Theorem 2. Moreover, since $(\tilde{f} + \tilde{g})^* = (f + g)^* \circ M^*$, one can also notice that $(x^*, \lambda^*)$ is a primal-dual point associated with Eq. (2) if and only if $(Mx^*, \lambda^*)$ is a primal-dual point associated with Eq. (6).

To recover the ADMM+ from the iterations (5a)--(5b), the starting point is Moreau’s identity [12, Th. 14.3] which reads

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

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

$$z^{k+1} = \text{prox}_{\rho h}(y^k + \rho \lambda^k)$$

$$= \arg \min_{w \in \mathcal{Y}} \left[ h(w) + \frac{\|w - (Mx^k + \rho \lambda^k)\|^2}{2\rho} \right],$$
Equation (5a) can be rewritten thanks to Moreau’s identity
\[ \lambda^{k+1} = \lambda^k + \rho^{-1}(Mx^k - z^{k+1}). \]

Now, Equation (5b) can be rewritten as
\[ y^{k+1} = \arg\min_{w \in \mathcal{S}} g(w) + \langle \nabla f(y^k), w \rangle + \frac{1}{2\tau} ||w - y^k + \tau(2\lambda^{k+1} - \lambda^k)||^2. \]

Upon noting that \( g(Mx) = g(x) \) and \( \langle \nabla f(y^k), Mx \rangle = \langle (M^{-1})^* \nabla f(M^{-1}Mx^k), Mx \rangle = \langle \nabla f(x^k), x \rangle \), the above equation becomes
\[ x^{k+1} = \arg\min_{w \in \mathcal{X}} g(w) + \langle \nabla f(x^k), w \rangle + \frac{1}{2\tau} ||Mw - u^{k+1} + \tau\lambda^{k+1}||^2. \]

where
\[ u^{k+1} = Mx^k + \tau(\lambda^k - \lambda^{k+1}) \]
\[ = (1 - \rho^{-1}\tau)Mx^k + \tau\rho^{-1}z^{k+1}. \]

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

D. 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, the ADMM+ boils down to the standard ADMM whose iterations are given by:
\[ z^{k+1} = \arg\min_{w \in \mathcal{Y}} \left[ h(w) + \frac{1}{2\rho} ||w - Mx^k - \lambda^k||^2 \right] \]
\[ \lambda^{k+1} = \lambda^k + \rho^{-1}(Mx^k - z^{k+1}) \]
\[ x^{k+1} = \arg\min_{w \in \mathcal{X}} \left[ g(w) + \frac{1}{2\rho} ||Mw - z^{k+1} + \rho\lambda^{k+1}||^2 \right]. \]

E. 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 \( \lambda^k \) is null for all \( k \geq 1 \) and \( u^k = x^k \). Then, the ADMM+ boils down to the standard Forward-Backward algorithm whose iterations are given by:
\[ x^{k+1} = \arg\min_{w \in \mathcal{X}} g(w) + \frac{1}{2\tau} ||w - (x^k - \tau \nabla f(x^k))||^2 \]
\[ = \text{prox}_{\tau g}(x^k - \tau \nabla f(x^k)). \]

One can remark that \( \rho \) has disappeared thus it can be set as large as wanted so the condition on stepsize \( \tau \) 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 [13].

III. COORDINATE DESCENT

A. Averaged operators and the primal-dual algorithm

Let \( \mathcal{H} \) be a Euclidean space\(^4\). For \( 0 < \alpha \leq 1 \), a mapping \( T : \mathcal{H} \to \mathcal{H} \) is \( \alpha \)-averaged if the following inequality holds for any \( x, y \in \mathcal{H} \):
\[ ||Tx - Ty||^2 \leq ||x - y||^2 - \frac{1 - \alpha}{\alpha} ||(I - T)x - (I - T)y||^2. \]
\(^4\)We refer to [12] for an extension to Hilbert spaces.

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 [12, Proposition 5.15, pp.80].

Lemma 1 (Krasnosel’skii-Mann iterations) Assume that \( T : \mathcal{H} \to \mathcal{H} \) is \( \alpha \)-averaged and that the set \( \text{fix}(T) \) 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 \( x^{k+1} = x^k + \eta_k(Tx^k - x^k) \) converges to some point in \( \text{fix}(T) \).

On the product space \( \mathcal{Y} \times \mathcal{Y} \), consider the operator
\[ V = \left( \tau^{-1}I Y, \frac{1}{\rho}Y \right). \]

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 \( \langle \cdot, \cdot \rangle_Y \) defined as \( \langle \xi, \eta \rangle_Y = \langle \xi, \eta \rangle \) where \( \langle \cdot, \cdot \rangle \) stands for the natural inner product on \( \mathcal{Y} \times \mathcal{Y} \). We denote by \( \mathcal{H}_Y \) the corresponding Euclidean space.

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

Lemma 2 ([3], [4]) Let Assumptions 2 and 3 hold true. Assume that \( \tau^{-1} - \rho^{-1} > L/2 \). Let \( (\lambda^{k+1}, y^{k+1}) = T(\lambda^k, y^k) \) where \( T \) is the transformation described by Equations (5a–5b). Then \( T \) is an \( \alpha \)-averaged operator on \( \mathcal{H}_Y \) 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 \( \alpha \) verifies \( \frac{1}{2} \leq \alpha < 1 \) which matches the definition of \( \alpha \)-averaged operators.

B. Randomized Krasnosel’skii Mann Iterations

Consider the space \( \mathcal{H} = \mathcal{H}_1 \times \cdots \times \mathcal{H}_J \) for some \( J \in \mathbb{N}^* \) where for any \( j, \mathcal{H}_j \) is a Euclidean space. Assume that \( \mathcal{H} \) is equipped with the scalar product \( (x, y)^J = \sum_{j=1}^J (x_j, y_j)_{\mathcal{H}_j} \) where \( (\cdot, \cdot)_{\mathcal{H}_j} \) is the scalar product in \( \mathcal{H}_j \). For \( j \in \{1, \ldots, J\} \), we denote by \( T_j : \mathcal{H} \to \mathcal{H}_j \) the components of the output of operator \( T : \mathcal{H} \to \mathcal{H}_j \) corresponding to \( \mathcal{H}_j \), thus
\[ T_j = \text{prox}_{\tau g_j}(x^k - \tau \nabla f_j(x^k)). \]

Denote by \( 2\tau \) the power set of \( J = \{1, \ldots, J\} \). For any \( k \in 2\tau \), we define the operator \( T^{(k)} : \mathcal{H} \to \mathcal{H} \) by \( T^{(k)}(x) = T_j x \) if \( j \in k \) and \( T^{(k)}(x) = x_j \) otherwise. On some probability space \( (\Omega, \mathcal{F}, \mathbb{P}) \), we introduce a random i.i.d. sequence \((\xi^k)^\infty_{k \in \mathbb{N}}\) such that \( \xi^k : \Omega \to 2\tau \) i.e.
\[ \xi^k(\omega) \text{ is a subset of } J. \]

We assume that the following holds:
\[ \forall j \in J, \exists k \in 2\tau, \ j \in k \text{ and } \mathbb{P}(\xi^k = j) > 0. \quad (7) \]

Let \( T \) be an \( \alpha \)-averaged operator, instead of considering the iterates \( x^{k+1} = x^k + \eta_k(Tx^k - x^k) \), we are now interested in a stochastic coordinate descent version of this algorithm that consists in iterates of the type \( x^{k+1} = x^k + \eta_k(T^{(\xi^{k+1})}x^k - x^k) \).

Theorem 3 Let \( T : \mathcal{H} \to \mathcal{H} \) be \( \alpha \)-averaged and \( \text{fix}(T) \neq \emptyset \). Assume that for all \( k \), sequence \((\eta_k)_{k \in \mathbb{N}}\) satisfies
\[ 0 < \lim \inf \eta_k \leq \lim \sup \eta_k < \frac{1}{\alpha}. \]
Let \((\xi^k)_{k \in \mathbb{N}}\) be a random i.i.d. sequence on \(2^T\) such that Condition (2) holds. Then, almost surely, the iterated sequence
\[
x^{k+1} = x^k + \eta_k (\hat{\mathcal{T}}(x^{k+1})x^k - x^k)
\]
converges to some point in \(\text{fix}(T)\).

**Proof.** The main idea behind the proof can be found in [6]. Define the operator \(U = (1 - \eta_k)I + \eta_k \mathcal{T}\) (we omit the index \(k\) in \(U\) to simplify notations); similarly, define \(U^\kappa = (1 - \eta_k)I + \eta_k \mathcal{T}^\kappa\). Remark that the operator \(U\) is \((\alpha \eta_k)\)-averaged.

The iteration (8) reads \(x^{k+1} = U(x^{k+1})x^k\). Set \(p_n = \mathbb{P}(\xi_1 = \kappa)\) for any \(\kappa \in 2^J\). Denote by \(\|x\|^2 = \langle x, x \rangle\) the squared norm in \(\mathcal{H}\). Define a new inner product \(\langle x, y \rangle = \sum_{j=1}^J q_j (x_j, y_j)_j\) on \(\mathcal{H}\) where \(q_j^{-1} = \sum_{\kappa \in 2^J} p_{\kappa} 1_{\{j \in \kappa\}}\) and let \(\|x\|^2 = \langle x, x \rangle\) be its associated squared norm. Consider any \(x^* \in \text{fix}(T)\). Conditionally to the sigma-field \(\mathcal{F}_k = \sigma(\xi_1, \ldots, \xi_k)\) we have
\[
\mathbb{E}(\|x^{k+1} - x^*\|^2 | \mathcal{F}_k) = \sum_{\kappa \in 2^J} p_{\kappa} \|U(\kappa)x^k - x^*\|^2
\]
\[
= \sum_{\kappa \in 2^J} p_{\kappa} \sum_{j \in \kappa} q_j \|U_j x^k - x^*_j\|^2 + \sum_{\kappa \in 2^J} p_{\kappa} \sum_{j \notin \kappa} q_j \|x^k_j - x^*_j\|^2
\]
\[
= \|x^k - x^*\|^2 + \sum_{\kappa \in 2^J} p_{\kappa} \sum_{j \in \kappa} q_j (\|U_j x^k - x^*_j\|^2 - \|x^k_j - x^*_j\|^2)
\]
\[
= \|x^k - x^*\|^2 + \sum_{j=1}^J (\|U_j x^k - x^*_j\|^2 - \|x^k_j - x^*_j\|^2)
\]
\[
= \|x^k - x^*\|^2 + \sum_{j=1}^J (\|U_j x^k - x^*_j\|^2 - \|x^k - x^*\|^2).
\]
Using that \(U\) is \((\alpha \eta_k)\)-averaged and that \(x^*\) is a fixed point of \(U\), the term enclosed in the parentheses is no larger than \(-\frac{1 - \alpha \eta_k}{\alpha \eta_k}\|1 - U\| x^k\|^2\). As \(1 - U = \eta_k (1 - T)\), we obtain:
\[
\mathbb{E}(\|x^{k+1} - x^*\|^2 | \mathcal{F}_k) \leq \|x^k - x^*\|^2 - \eta_k (1 - \alpha \eta_k)\|1 - T\| x^k\|^2 \tag{9}
\]
which shows that \(\|x^k - x^*\|^2\) 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 \(\text{fix}(T)\), there is a probability one set on which \(\|x^k - x\| \to X_x \in [0, \infty)\) for all \(x \in H\). Let \(x^* \in \text{fix}(T)\), let \(\varepsilon > 0\), and choose \(x \in H\) such that \(\|x - x^*\| \leq \varepsilon\). With probability one, we have
\[
\|x^k - x^*\| \leq \|x^k - x\| + \|x - x^*\| \leq X_x + 2\varepsilon
\]
for \(k\) large enough. Similarly, \(\|x^k - x^*\| \geq X_x - 2\varepsilon\) for \(k\) large enough. Therefore we obtain:

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

Getting back to (9), taking the expectations on both sides of this inequality and iterating over \(k\), we obtain
\[
\sum_{k=0}^\infty \eta_k (1 - \alpha \eta_k)\mathbb{E}(\|1 - T\| x^k\|^2) \leq (x^0 - x^*\|^2.
\]
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 \(\sum_{k=0}^\infty \mathbb{E}(\|1 - T\| x^k\|^2)\) is finite. By Markov’s inequality and Borel Cantelli’s lemma, we therefore obtain:

**C2 :** \((1 - T)x^k \to 0\) almost surely.

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

**Remark 1** At the time of writing the paper, the work [14] was brought to our knowledge. A result similar to Theorem 3 is presented and extended to the case of infinite dimensional Hilbert spaces. The main core of the proof of [14] relies on the same idea as the one developed in [6] and presented above.

By Lemma 2 the ADMM+ iterates are generated by the action of an \(\alpha\)-averaged operator. Theorem 3 shows then that a stochastic coordinate descent version of the ADMM+ converges towards a primal-dual point. This result will be exploited in two directions: first, we describe a stochastic minibatch algorithm, where a large dataset is randomly split into smaller chunks. Second, we develop an asynchronous version of the ADMM+ in the context where it is distributed on a graph.

### IV. APPLICATION TO STOCHASTIC APPROXIMATION

#### A. Problem Setting

Given an integer \(N \geq 1\), consider the problem of minimizing a sum of composite functions
\[
\inf_{x \in \mathcal{X}} \sum_{n=1}^N (f_n(x) + g_n(x)) \tag{10}
\]
where we make the following assumption:

**Assumption 4** For each \(n = 1, \ldots, N\),

(i) \(f_n\) is a convex differentiable function on \(\mathcal{X}\), and its gradient \(\nabla f_n\) is \(L\)-Lipschitz continuous on \(\mathcal{X}\) for some \(L \geq 0\).

(ii) \(g_n \in \Gamma_0(\mathcal{X})\).

(iii) The infimum of Problem (10) is attained.

(iv) \(\cap_{n=1}^N \text{ri dom } g_n = \emptyset\).

This problem arises for instance in large-scale learning applications where the learning set is too large to be handled as a single block. Stochastic minibatch approaches consist in splitting the data set into \(N\) chunks and to process each chunk in some order, one at a time. The quantity \(f_n(x) + g_n(x)\) measures the inadequacy between the model (represented by parameter \(x\)) and the \(n\)-th chunk of data. Typically, \(f_n\) stands for a data fitting term whereas \(g_n\) is a regularization term which penalizes the occurrence of erratic solutions. As an example, the case where \(f_n\) is quadratic and \(g_n\) is the \(\ell_1\)-norm reduces to the popular LASSO problem [15].
B. Instantiating the ADMM+

We derive our stochastic minibatch algorithm as an instance of the ADMM+ coupled with a randomized coordinate descent. To that end let us first rephrase Problem (10) as

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

where the notation $x_n$ represents the $n$-th component of any $x \in \mathcal{X}^N$, $\iota_{\mathcal{A}}$ is the indicator function of a set $\mathcal{A}$ (null on $A$ and infinite outside this set), and $\mathcal{C}$ is the space of vectors $x \in \mathcal{X}^N$ such that $x_1 = \cdots = x_N$. On the space $\mathcal{X}^N$, we set $f(x) = \sum_n f_n(x_n)$, $g(x) = \sum_n g_n(x_n)$, $h(x) = \iota_{\mathcal{C}}$ and $M = 1_{\mathcal{X}^N}$ the identity matrix. Problem (11) can be rewritten as

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

We define the natural scalar product on $\mathcal{X}^N$ as $\langle x, y \rangle = \sum_{n=1}^{N} \langle x_n, y_n \rangle$. Applying the ADMM+ to Problem (12) leads to the following procedure:

$$
\begin{align*}
\lambda^{k+1}_n &= \text{proj}_C[\|x^n - \rho \lambda^k\|^2] - \rho \lambda^k + \sum_{n=1}^{N} x^n \\
\lambda^{k+1} &= \lambda^k + \rho^{-1}(\lambda^{k+1}_n - x^{k+1}_n) \\
u^{k+1} &= (1 - \tau \rho^{-1})x^k + \tau \rho^{-1}x^{k+1} \\
x_n^{k+1} &= \arg\min_{w \in \mathcal{X}} \left[ g_n(w) + \langle \nabla f_n(x^k_n), w \rangle + \frac{\|w - x^{k+1}_n + \tau \lambda^{k+1}_n\|^2}{2\tau} \right]
\end{align*}
$$

where $\text{proj}_C$ is the orthogonal projection onto $\mathcal{C}$ (observe that the $\text{proj}$ of an indicator function on a closed convex set coincides with the orthogonal projection on that set). Note that for any $x \in \mathcal{X}^N$, $\text{proj}_C(x)$ is equal to $(\bar{x}, \cdots, \bar{x})$ where $\bar{x}$ is the average of vector $x$, i.e., $\bar{x} = N^{-1} \sum x_n$. Consequently, the components of $x^{k+1}$ are equal and coincide with $\bar{x} + \rho \bar{\lambda}$ where $\bar{x}$ and $\bar{\lambda}$ are the averages of $x^k$ and $\lambda^k$ respectively. By inspecting the $\lambda^k$-update equation above, we notice that the latter equality simplifies even further by noting that $\bar{x}^{k+1}$ is equal or, equivalently, $\bar{\lambda}^k = 0$ for all $k \geq 1$ if the algorithm is started with $\bar{x}^0 = 0$. Finally, for any $n$ and $k \geq 1$, the above iterations reduce to

$$
\begin{align*}
\bar{x}^k &= \frac{1}{N} \sum_{n=1}^{N} x_n^k \\
\lambda^{k+1}_n &= \lambda^k + \rho^{-1}(\lambda^{k+1}_n - \bar{x}^k) \\
u^{k+1} &= (1 - \tau \rho^{-1})x^k + \tau \rho^{-1} \bar{x}^k \\
x_n^{k+1} &= \text{prox}_{\tau g_n}[x^{k+1}_n - \tau(\nabla f_n(x^k_n) + \lambda^{k+1}_n)]
\end{align*}
$$

where we recall the notation $\text{prox}$ defined in (4). These iterations can be written more compactly as

Minibatch ADMM+

Initialization: $(x^0, \lambda^0)$ s.t. $\sum_n \lambda^0_n = 0$.

Do

- $\bar{x}^k = \frac{1}{N} \sum_{n=1}^{N} x_n^k$
- $\lambda^{k+1} = \lambda^k + \rho^{-1}(\lambda^{k+1}_n - \bar{x}^k)$
- $u^{k+1} = (1 - \tau \rho^{-1})x^k + \tau \rho^{-1} \bar{x}^k$
- $x_n^{k+1} = \text{prox}_{\tau g_n}[x^{k+1}_n - \tau(\nabla f_n(x^k_n) + \lambda^{k+1}_n)]$

The following result is a straightforward consequence of Theorem 1.

**Theorem 4** Let Assumption 2 hold true and assume that $\tau - \rho^{-1} > L/2$. Then for any initial point $(x^0, \lambda^0)$ such that $\bar{\lambda}^0 = 0$, there exists a primal-dual point $(x^*, \lambda^*)$ of Problem (12) such that $(\bar{x}^k, \bar{\lambda}^k)$ converges to $(x^*, \lambda^*)$.

At each step $k$, the iterations given above involve the whole set of functions $f_n, g_n$ ($n = 1, \ldots, N$). Our aim is now to propose an algorithm which involves a single couple of functions $(f_n, g_n)$ per iteration.

C. A Stochastic Minibatch Primal Dual algorithm

We are now in position to state the main algorithm of this section. The proposed Stochastic Minibatch Primal Dual algorithm (SMPD) is obtained upon applying the randomized coordinate descent on the minibatch ADMM+:

**SMPD Algorithm:**

**Initialization:** $(x^0, \lambda^0)$.

Do

- Define $\bar{x}^k$ and $\bar{\lambda}^k$ as $\bar{x}^k = \frac{1}{N} \sum_{n=1}^{N} x_n^k$, $\bar{\lambda}^k = \frac{1}{N} \sum_{n=1}^{N} \lambda_n^k$.
- Pick up the value of $\xi^{k+1}$.
- For Batch $n = \xi^{k+1}$, set
  $$
  \begin{align*}
  \lambda^{k+1}_n &= \lambda^k_n - \bar{x}^k + \frac{x_n^k - \bar{x}^k}{\rho} \\
  x_n^{k+1} &= \text{prox}_{\tau g_n}[1 - (1 - 2\tau \rho^{-1})x_n^k - \tau \lambda^{k+1}_n - \tau(\nabla f_n(x_n^k) + \lambda^{k+1}_n)]
  \end{align*}
  $$
- For all batches $n \neq \xi^{k+1}$, set $\lambda^{k+1}_n = \lambda^k_n$, $x_n^{k+1} = \bar{x}^k$.
- Increment $k$.

**Assumption 5** The random sequence $(\xi^k)_{k \in \mathbb{N}^+}$ is i.i.d. and satisfies $\mathbb{P}[\xi^k = n] > 0$ for all $n = 1, \ldots, N$.

**Theorem 5** Let Assumptions 2 and 5 hold true. Assume that $\tau - \rho^{-1} > L/2$. For any initial point $(x^0, \lambda^0)$, the sequence $\bar{x}^k$ generated by the SMPD algorithm converges almost surely to a minimizer $x^*$ of Problem (12).

**Proof:** Let us define $(f, g, h, M) = (f, g, h, I_{\mathcal{X}^N})$ where the functions $f$, $g$, and $h$ are the ones defined in Section IV-B. Then the iterates $(\lambda^{k+1}_n)_{n=1}^{N}$, $(x^{k+1}_n)_{n=1}^{N}$ described by Equations (13) coincide with the iterates $(\lambda^{k+1}, y^{k+1})$ described by Equations (5). If we write these equations more compactly as $(\lambda^{k+1}, x^{k+1}) = T(\lambda^k, x^k)$ where the operator $T$ acts in...
the space $\mathcal{H} = \mathcal{X}^N \times \mathcal{X}^N$, then Lemma 2 shows that $T$ is $\alpha$-averaged. Defining the selection operator $S_n$ on $\mathcal{H}$ as $S_n(\lambda, x) = (\lambda_n, x_n)$, we obtain that $\mathcal{H} = S_1(\mathcal{H}) \times \cdots \times S_N(\mathcal{H})$ up to an element reordering. To be compatible with the notations of Section II-B we assume that $J = N$ and that the random sequence $\xi^s$ driving the SMPD algorithm is set valued in $\{1, \ldots, [N]\} \subset 2^J$. In order to establish Theorem 5 we need to show that the iterates $(\lambda^{k+1}, x^{k+1})$ provided by the SMPD algorithm are those who satisfy the equation $(\lambda^{k+1}, x^{k+1}) = T(\xi^{k+1}, (\lambda^k, x^k))$. Theorem 5 follows then by the direct application of Theorem 4.

Let us start with the update equation. Since $h = \iota_C$, its Legendre-Fenchel transform is $h^* = \iota_{C^\perp}$ where $C^\perp$ is the orthogonal complement of $C$ in $\mathcal{X}^N$. Consequently, if we write $(n^{k+1}, g^{k+1}) = T(\lambda^{k+1}, x^{k+1})$, then by Eq. (33),

$$\eta^{k+1} = \lambda^{k+1} - x^k + \frac{x^k - x^k}{\rho} \quad n = 1, \ldots , N$$

Notice that in general, $\lambda^{k+1} \neq 0$ because in the SMPD algorithm, only one component is updated at a time. If $\{n\} = \xi^{k+1}$, then $\lambda^{k+1} = \eta^{k+1}$ which is Eq. (14). All other components of $\lambda^k$ are carried over to $\lambda^{k+1}$.

By Equation 5(b) we also get

$$g^{k+1} = \text{prox}_{\rho g_n} \left[ x^k - \nabla f_n(x_n^k) - \tau (2\lambda^{k+1} - \lambda^k) \right].$$

If $\{n\} = \xi^{k+1}$, then $x^{k+1} = g^{k+1}$ can easily be shown to be given by Eq. (15).

V. DISTRIBUTED OPTIMIZATION

Consider a set of $N > 1$ computing agents that cooperate to solve the minimization problem (10). Here, $f_n, g_n$ are two private functions available at Agent $n$. Our purpose is to design a random distributed (or decentralized) iterative algorithm where, at 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) if any.

Instances of this problem appear in learning applications where massive training data sets are distributed over a network and processed by distinct machines [16], [17], in resource allocation problems for communication networks [18], or in statistical estimation problems by sensor networks [19], [20].

A. NETWORK MODEL AND PROBLEM FORMULATION

We represent the network as a graph $G = (V, E)$ where $V = \{1, \ldots , N\}$ is the set of agents/nodes and $E \subset \{1, \ldots , N\}^2$ is the set of undirected edges. 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 6 $G$ is connected and has no self loop.

Let us introduce some notations. For any $x \in \mathcal{X}^V$, we denote by $x_n$ the components of $x$, i.e., $x = (x_n)_{n \in V}$. We introduce the functions $f$ and $g$ on $\mathcal{X}^V \rightarrow (-\infty, +\infty]$ 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.

The next step is to rewrite the latter constraint in a way that involves the graph $G$. We replace the global consensus constraint by a modified version of the function $\iota_C$ (introduced in Eq. (11)). Our goal will be to ensure global consensus through local consensus over every edge of the graph.

For any $\epsilon \in E$, say $\epsilon = \{n, m\} \in V$, we define the linear operator $M_\epsilon : \mathcal{X}^V \rightarrow \mathcal{X}^2$ as $M_\epsilon(x) = (x_n, x_m)$ where we assume some ordering on the nodes (that will be irrelevant in the sequel) to avoid any ambiguity on the definition of $M$. We construct the linear operator $M : \mathcal{X}^V \rightarrow \mathcal{X}^2$ as $M(x) = (M_\epsilon(x))_{\epsilon \in E}$ where we also assume some 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 $y_\epsilon$ will be represented by the couple $y_\epsilon = (y_\epsilon(n), y_\epsilon(m))$ with $n < m$. We also introduce the subspace of $\mathcal{X}^2$ defined as $C_2 = \{x, x : x \in \mathcal{X}\}$. Finally, we define $h : \mathcal{Y} \rightarrow (-\infty, +\infty]$ as

$$h(y) = \sum_{\epsilon \in E} \iota_{C_2}(y_\epsilon).$$

We consider the following problem:

$$\inf_{x \in \mathcal{X}^V} f(x) + g(x) + h(Mx).$$

Lemma 3 Let Assumptions 2 hold true. The minimizers of (17) are the tuples $(x^*, \cdots , x^*)$ where $x^*$ is any minimizer of (10).

Proof: Assume that Problem (17) has a minimizer $x = (x_1, \ldots , x_V)$. Then

$$h(Mx) = \sum_{\epsilon = (n, m) \in E} \iota_{C_2}(x_n, x_m) = 0.$$ 

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

B. INSTANTIATING THE ADMM+

We now apply the ADMM+ to solve the problem (17). Since the newly defined function $h$ is separable with respect to the $(y_\epsilon)_{\epsilon \in E}$, we get

$$\text{prox}_{\rho h}(y) = (\text{prox}_{\rho \iota_{C_2}}(y_\epsilon))_{\epsilon \in E} = \left( (\bar{y}_\epsilon, \bar{y}_\epsilon) \right)_{\epsilon \in E}$$

where $y_\epsilon = (y_\epsilon(n), y_\epsilon(m))$ if $\epsilon = \{n, m\}$. With this at hand, the update equation (33) of the ADMM+ is written as $z^{k+1} = ((z^{k+1}_n, z^{k+1}_m))_{\epsilon \in E}$ where $z^{k+1}_\epsilon = (x^{k}_n + x^{k}_m)/2 + \rho (\lambda^{k+1}_n + \lambda^{k+1}_m)/2$ for any $\epsilon = \{n, m\} \in E$. Plugging this equality into Eq. (33), it can be seen that $\lambda^{k+1}_n = -\lambda^{k}_n$. Therefore, $z^{k+1}_\epsilon = (x^{k}_n + x^{k}_m)/2$ for any $k \geq 1$. Moreover, $\lambda^{k+1}_n = \lambda^{k}_n + (x^{k}_n - x^{k}_m)/(2\rho)$. Let us now instantiate Equations (33) and (34). 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. \ref{3d}, the \( n \)th component of \( x^{k+1} \) is written

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

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 \( M^* y \in \mathcal{X}^{|V|} \). Plugging Eq. \ref{3d} together with the expressions of \( \tilde{z}_m^{k+1} \) and \( \tilde{\lambda}_m^{k+1} \) (in the argument of \( \text{prox}_{\tau g_n/d_n} \)), we get after a small calculation

\[
x_n^{k+1} = \text{prox}_{\tau g_n/d_n} \left[ (1 - \tau \rho^{-1})x_n^{k} - \frac{\tau}{d_n} \nabla f_n(x_n^{k}) + \frac{\tau}{d_n} \sum_{m:\{n,m\} \in E} (\rho^{-1} x_n^{k} - \lambda^{k}_{m,n}(n)) \right]
\]

The algorithm is finally described by the following procedure: Prior to the clock tick \( k+1 \), the node \( n \) has in its memory the variables \( x_n^{k}, \{\lambda^{k}_{m,n}(n)\}_{m\sim n} \), and \( \{x_n^{k}\}_{m\sim n} \).

**Distributed ADMM+**

Initialization: \((x^0, \lambda^0)\) s.t. \( \sum_n \lambda_n^0 = 0 \).

Do

- For any \( n \in V \), Agent \( n \) performs the following operations:

\[
\lambda^{k+1}_{m,n}(n) = \lambda^{k}_{m,n}(n) + \frac{x_n^{k} - x_m^{k}}{2 \rho} \quad \text{for all} \quad m \sim n
\]

\[
x_n^{k+1} = \text{prox}_{\tau g_n/d_n} \left[ (1 - \tau \rho^{-1})x_n^{k} - \frac{\tau}{d_n} \nabla f_n(x_n^{k}) + \frac{\tau}{d_n} \sum_{m:\{n,m\} \in E} (\rho^{-1} x_n^{k} - \lambda^{k}_{m,n}(n)) \right]
\]

- Agent \( n \) sends the parameter \( x_n^{k+1} \) to its neighbors,
- Increment \( k \).

Under Assumption \ref{5} it can be shown after some algebra the Lipschitz constant of \( \nabla (f \circ M^{-1}) \) is no larger than \( L \). Then, the following result is a straightforward consequence of Theorem \ref{1}.

**Theorem 6** Let Assumptions \ref{2} and \ref{6} hold true. Assume that \( \tau^{-1} - \rho^{-1} > L/2 \). For any initial value \((x^0, \lambda^0)\), let \((x^k)_{k \in \mathbb{N}}\) be the sequence produced by the Distributed ADMM+. Then, there exists a minimizer \( x^* \) of Problem \ref{10} such that for all \( n \in V \), \((x^k)_{k \in \mathbb{N}}\) converges to \( x^* \).

**C. A Distributed Asynchronous Primal Dual Algorithm**

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 \( \xi^k \) 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: \((x^0, \lambda^0)\).

Do

- Select a random set of agents \( \xi^{k+1} = \mathcal{A} \).
- For any \( n \in \mathcal{A} \), Agent \( n \) performs the following operations:
  - For all \( m \sim n \), do
    \[
    \lambda_{m,n}^{k+1}(n) = \lambda_{m,n}^{k}(n) - \lambda_{m,n}^{k}(m) + \frac{x_n^{k} - x_m^{k}}{2 \rho};
    \]
    \[
    x_n^{k+1} = \text{prox}_{\tau g_n/d_n} \left[ (1 - \tau \rho^{-1})x_n^{k} - \frac{\tau}{d_n} \nabla f_n(x_n^{k}) + \frac{\tau}{d_n} \sum_{m:\{n,m\} \in E} (\rho^{-1} x_n^{k} + \lambda_{m,n}^{k}(m)) \right].
    \]
  - For all \( m \sim n \), send \( \{x_n^{k+1}, \lambda_{m,n}^{k+1}(n)\} \) to Neighbor \( m \).
- For any Agent \( n \not\in \mathcal{A} \), \( x_n^{k+1} = x_n^{k} \), and \( \lambda_{m,n}^{k+1}(n) = \lambda_{m,n}^{k}(n) \) for all \( m \sim n \).
- Increment \( k \).

**Assumption 7** The collections of sets \( \{A_1, A_2, \ldots\} \) such that \( \mathcal{P} \{\xi^k = A_i\} \) is positive satisfies \( \bigcup A_i = V \).

In other words, any agent is selected with a positive probability.

**Theorem 7** Let Assumptions \ref{2} and \ref{6} hold true. Assume that \( \tau^{-1} - \rho^{-1} > L/2 \). Let \((x^k)_{k \in \mathbb{N}}\) be the output of the DAPD algorithm. For any initial value \((x^0, \lambda^0)\), the sequences \(x^1, \ldots, x^k\) converge almost surely as \( k \to \infty \) to a minimizer \( x^* \) of Problem \ref{10}.

**Proof:** Let \((\tilde{f}, \tilde{g}, h) = (f \circ M^{-1}, g \circ M^{-1}, h) \) where \( f, g, h \) and \( M \) are those of Problem \ref{17}. For these functions, write Equations \ref{5} as \((\lambda^{k+1}, y^{k+1}) = T(\lambda^k, y^k) \). By Lemma \ref{2} the operator \( T \) is an \( \alpha \)-averaged operator acting on the space \( \mathcal{H} = Y \times S \), where \( S \) is the image of \( \mathcal{X}^{|V|} \) by \( M \). For any \( n \in V \), let \( S_n \) be the selection operator on \( \mathcal{H} \) defined as \( S_n(\lambda, Mx) = (\lambda(n))_{n \in E; n \not\in \xi^k, x_n} \). Then it is easy to see that up to an element reordering, \( \mathcal{H} = S_1(\mathcal{H}) \times \cdots \times S_{|V|}(\mathcal{H}) \). Identifying the set \( \mathcal{J} \) introduced before the statement of Theorem \ref{3} with \( V \), the operator \( T(\xi^k) \) is defined as follows: if \( n \in \xi^k \), then \( S_n(T(\xi^k)(\lambda, Mx)) = S_n(\lambda, Mx) \) while if \( n \not\in \xi^k \), then \( S_n(T(\xi^k)(\lambda, Mx)) = S_n(\lambda, Mx) \). We know by Theorem \ref{3} that the sequence \((\lambda^{k+1}, Mx^{k+1}) = T(\xi^{k+1})(\lambda^k, Mx^k) \) converges almost surely to a primal-dual point of Problem \ref{6}. This implies by Lemma \ref{3} that the sequence \( x^k \) converges almost surely to \((x^*, \ldots, x^*)\) where \( x^* \) is a minimizer of Problem \ref{10}.

We therefore need to prove that the operator \( T(\xi^{k+1}) \) is translated into the DAPD algorithm. The definition \ref{16} of
\[ h^*(\phi) = \sum_{e \in E} \lambda \phi_2(x_e) \]

where \( C^2_+ = \{(x, -x) : x \in \mathcal{X}\} \). Therefore, writing
\[ (\eta^{k+1}, q^{k+1}) = M \nu^{k+1} = T(\lambda^k, y^k = Mx^k), \]

Equation (5b) shows that
\[ \eta_k^{k+1} = \text{proj}_{C^2_+}(\lambda^k + \rho^{-1} y^k). \]

Notice that contrary to the case of the synchronous algorithm \( (18) \), there is no reason here for which \( \text{proj}_{C^2_+}(\lambda^k) = 0 \).

Getting back to \( (\lambda^{k+1}, Mx^{k+1}) = T^{k+1}(\lambda^k, y^k = Mx^k) \), we therefore obtain that for all \( n \in \xi^{k+1} \) and all \( m \sim n \),
\[ \lambda^{k+1}_{n,m}(n) = \frac{\lambda^k_{n,m}(n) - \lambda^k_{n,m}(m)}{2} + \frac{y^k_{n,m}(n) - y^k_{n,m}(m)}{2\rho} \]
\[ = \frac{\lambda^k_{n,m}(n) - \lambda^k_{n,m}(m)}{2} + \frac{x^k_n - x^k_m}{2\rho}. \]

Recall now that Eq. (5b) can be rewritten as
\[ q^{k+1} = \arg\min_{w \in \mathcal{X}} \bar{g}(w) + \langle \nabla \bar{f}(y^k), w \rangle + \frac{\|w - y^k + \tau(2\lambda^{k+1} - \lambda^k)\|^2}{2\tau}. \]

Upon noting that \( \bar{g}(Mx) = g(x) \) and \( \langle \nabla \bar{f}(y^k), Mx \rangle = (M^{-1})^* \nabla f(M^{-1}Mx^k, Mx) = \langle \nabla f(x^k), x \rangle \), the above equation becomes
\[ v^{k+1} = \arg\min_{w \in \mathcal{X}} g(w) + \langle \nabla f(x^k), w \rangle + \frac{\|M(w - x^k) + \tau(2\lambda^{k+1} - \lambda^k)\|^2}{2\tau}. \]

Recall that \( (M^*Mx)_n = d_n x_n \). Hence, for all \( n \in \xi^{k+1} \), we get after some computations
\[ x^{k+1}_n = \text{prox}_{\tau y_n, d_n} \left[ x^k_n - \frac{\tau}{d_n} \nabla f_n(x^k_n) \right] \]
\[ - \frac{\tau}{d_n} \left( M^* \left( 2\lambda^{k+1} - \lambda^k \right) \right)_n. \]

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.

**VI. NUMERICAL ILLUSTRATIONS**

**Problem.** We address the problem of \( \ell_1 \)-regularized logistic regression. Denoting by \( m \) the number of observations and by \( p \) the number of features, the optimization problem writes
\[ \min_{x \in \mathbb{R}^p} \frac{1}{m} \sum_{t=1}^m \log \left( 1 + e^{-y_t a_t x} \right) + \mu \|x\|_1 \]
where \( (y_t)_{t=1}^m \) are in \( \{-1, +1\} \), the \( (a_t)_{t=1}^m \) are in \( \mathbb{R}^p \), and \( \mu > 0 \) is a scalar. Let \( (B_n)_{n=1}^N \) denote a partition of \( \{1, \ldots, m\} \). The optimization problem then writes
\[ \min_{x \in \mathbb{R}^p} \sum_{n=1}^N \sum_{t \in B_n} \frac{1}{m} \log \left( 1 + e^{-y_t a_t x} \right) + \mu \|x\|_1 \]

or, splitting the problem between the batches
\[ \min_{x \in \mathbb{R}^p} \sum_{n=1}^N \left( \frac{1}{m} \sum_{t \in B_n} \log \left( 1 + e^{-y_t a_t x} \right) + \frac{\mu}{N} \|x\|_1 \right) + \iota \mathcal{C}(x) \]

where \( x = (x_1, \ldots, x_N) \) is now in \( \mathbb{R}^{Np} \). Obviously Problems (19), (20) and (21) are equivalent and Problem (21) is in the form of (11).

**Dataset.** In the whole section, we will perform our simulations on the classical \texttt{covtype} dataset available from the LIBSVM website.

This dense dataset has \( m = 581012 \) observations and \( p = 54 \) features; we preprocessed it so that the features have zero mean and unit variance.

We will consider two different setups:

**A. Minibatch.** At each iteration, a random batch of data is processed. In this setup, we apply the SMPD algorithm described in Section IV.

**B. Distributed Optimization.** At each iteration, one (or multiples) random agents process their own batch of data and communicate with its neighbors. In this setup, we apply the DAPD algorithm described in Section V.

**A. Minibatch**

**Setup.** Consider Problem (20), the goal of minibatch processing is to find a minimum of this problem by using one randomly selected batch per iteration. We processed the \texttt{covtype} dataset through 581 batches of 1000 observations and setted the regularization parameter \( \mu \) to \( 10^{-3} \).

**SMPD Algorithm for Problem (21):**

**Initialization:** \( (x^0, \lambda^0) \).

Do

- Define \( \bar{x}^k \) and \( \bar{\lambda}^k \) as
\[ \bar{x}^k = \frac{N}{n=1} \sum_{n=1}^N x^k_n, \quad \bar{\lambda}^k = \frac{N}{n=1} \sum_{n=1}^N \lambda^k_n, \]

- Pick up the value of \( \xi^{k+1} \).

For Batch \( n = \xi^{k+1} \), set
\[ \lambda_{n}^{k+1} = \lambda_{n}^{k} - \bar{\lambda}^{k} + \frac{x_n^k - \bar{x}^k}{\mu}, \]
\[ x_n^{k+1} = \text{SoftTh} \left[ (1 - 2\tau \rho^{-1}) x_n^k - \tau \nabla l_n(x_n^k) - \tau \lambda_n^k + 2\tau (\rho^{-1} x_n^k + \bar{\lambda}^k), \frac{\tau \mu}{N} \right]. \]

- For all batches \( n \neq \xi^{k+1} \), set \( \lambda_{n}^{k+1} = \lambda_{n}^{k} \) and \( x_n^{k+1} = x_n^{k} \).

- Increment \( k \)

where \( \text{SoftTh}[: \lambda] \) denotes the soft-thresholding operator with parameter \( \lambda \) and \( \nabla l_n \) is the gradient of \( \sum_{t \in B_n} \frac{1}{m} \log(1 + e^{-y_t a_t x}) \) with respect to \( x \).

**Compared Algorithms.** As our goal is to show the performance of randomized optimization algorithms derived from ADMM+ or ADMM*, we compare ourselfs with algorithms using a gradient step for the data fitting function and we do not

\[ \text{http://www.csie.ntu.edu.tw/~cjlin/libsvm/} \]
consider here acceleration techniques nor stepsize selection procedures. The considered algorithms are:

- **SGD**: the stochastic (sub-)gradient descent (see [21] and references therein) applied to Problem (20) with 1/k stepsize.
- **MISO**: the MISO algorithm for composite optimization [22], [23] applied to Problem (20) with 1/L stepsize, where L is set to the maximum of the upper bounds of the Lipschitz constants per batch \( L = 0.25 \max_{n=1,...,N} \|a_n\|_2^2 \).
- **SMPD**: our SMPD algorithm described above with parameters \( \rho \) and \( \tau \) set manually for good performance while satisfying the condition of Theorem 5.

In Figure 1 we plot the \( \ell_1 \)-regularized logistic loss versus the number of passes over the data (estimated as the number of iterations divided by the number of batches). We observe that the SMPD performs significantly better than the stochastic subgradient. It also performs well compared to MISO. It is worth noticing that, unlike for MISO (for which a null initialization often leads to a stalling algorithm), we did not observe bad starting points for SMPD leading to a bad stationary point of the algorithm.

![Figure 1](image_url)

**Comparison of minibatch algorithms.**

### B. Distributed Optimization

**Setup.** Now, we consider the case where the dataset is scattered over a network and the agents want to solve it in a decentralized manner. To do so, we introduce a graph \( G = (V, E) \) representing the connections between the agents. To solve Problem (20) distributively, we rewrite it in the form of Problem (17) using the same function \( h \) and matrix \( M \) as in Section V.

\[
\min_{x \in \mathbb{R}^n} \sum_{n=1}^{N} \left( \sum_{t \in B_n} \frac{1}{m} \log \left( 1 + e^{-y_{nt} \langle a_{nt} \rangle} \right) + \frac{\mu}{N} \|x_n\|_2^2 \right) + \sum_{e \in E} \ell_E(y_e).
\]

We solved this problem on a 10 \times 10 2D toroidal grid by assigning 1000 observations taken in the **covertype** dataset to each agent so that the whole network processes 100000 observations. The regularization parameter \( \mu \) was also set to \( 10^{-3} \). We remind the fact that in our DAPD algorithm, a agent randomly activates, processes its data, and sends information to its neighbors: no coordinator/fusion center is present to collect or manage data.

#### DAPD Algorithm for Problem (22):

**Initialization:** \((x^0, \lambda^0)\).

- Select a random set of agents \( \xi^{k+1} = \Xi \).
- For any \( n \in \Xi \), Agent \( n \) performs the following operations:
  - For all \( m \sim n \), do
    \[
    \lambda^{k+1}_{k,n,m}(n) = \frac{\lambda^{k}_{k,n,m}(n) - \lambda^{k}_{k,n,m}(m)}{2} + \frac{x^k_n - x^k_m}{2\rho}.
    \]
  - Then compute
    \[
    x^{k+1}_n = \text{SoftTh} \left[ (1 - \tau \rho^{-1})x^k_n - \tau \frac{\partial h_n(x^k_n)}{\partial x_n} \right] + \frac{\tau \rho}{\partial h_n} \sum_{m \sim n} (\rho^{-1}x^k_m + \lambda^{k}_{k,n,m}(m)) - \frac{\tau \rho}{N} d_n
    \]
  - For all \( m \sim n \), send \( \{x^{k+1}_n, \lambda^{k+1}_{k,n,m}(n)\} \) to Neighbor \( m \).
- For any Agent \( n \notin \Xi \), \( x^{k+1}_n = x^k_n \), and \( \lambda^{k+1}_{k,n,m}(n) = \lambda^{k}_{k,n,m}(n) \) for all \( m \sim n \).
- Increment \( k \).

**Compared Algorithms.** As before, we focus on the optimization algorithm itself and now compare ourselves with distributed algorithms using a gradient step for the data fitting function. The considered algorithms are:

- **ABG**: the asynchronous broadcast (sub-)gradient (see [24] and references therein) applied to Problem (20) with 1/k stepsize.
- **DSGD**: the distributed stochastic (sub-)gradient descent (see [20] and references therein) applied to Problem (20) with 1/k stepsize. Contrary to the other algorithms, at each iteration this algorithm demands a gradient descent for each agent followed by a randomly chosen communication. For fairness in the comparison in terms of communications, we used simple Broadcast Gossip Algorithm [25] as communication.
- **DAPD**: our DAPD algorithm described above with parameters \( \rho \) and \( \tau \) set manually for good performance while satisfying the condition of Theorem 7.

In Figure 2 we plot the \( \ell_1 \)-regularized logistic loss at some agent (they are indistinguishable from a network point of
view) versus the number of iterations (i.e. the number of agent activations). We observe that the DAPD is significantly quicker than the distributed stochastic subgradient, itself way quicker than the asynchronous broadcast gradient. In particular, this is due to the fact that the DAPD can be seen as a Primal-Dual algorithm distributed on a graph, which performs significantly better than a stochastic gradient especially on regularized regression problem for a comparable computational cost.

![Comparison of distributed optimization algorithms.](image)

We remark that the quantity of information exchanged per iteration 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 $Tp$).

VII. 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 Vû and Condat. As a byproduct, we obtained a stochastic approximation algorithm which can be used to handle distinct data blocks sequentially. We also 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 stepizes strategies.

REFERENCES

[1] Daniel Gabay and Bertrand 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.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] P. L. Combettes and J.-C. Pesquet, “Proximal splitting methods in signal processing,” in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pp. 185–212. Springer, 2011.

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

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

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

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

[12] 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.

[13] 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.

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

[15] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[16] 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.

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

[18] 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.

[19] 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.

[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] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *International Conference on Computational Statistics (COMPSTAT)*, 2010, pp. 177–186, Springer.

[22] J. Mairal, “Optimization with first-order surrogate functions,” in *International Conference on Machine Learning (ICML)*, 2013, pp. 783–791.

[23] J. Mairal, “Incremental majorization-minimization optimization with application to large-scale machine learning,” *arXiv preprint arXiv:1402.4419*, 2014.

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

[25] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, “Broadcast gossip algorithms for consensus,” *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2748–2761, 2009.