A Stochastic Coordinate Descent Primal-Dual Algorithm and Applications

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 Vũ and Condat. Second, we provide a randomized version of the algorithm based on the idea of coordinate descent. Finally, we address two applications of our method: *(i)* In the case of stochastic minibatch optimization, our method can be used to split an 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 objective functions and seeking to find a consensus on the minimum of the aggregate objective. Our method yields a distributed iterative algorithm where agent use both local computations and message passing in an asynchronous manner.**

***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 $M : \mathcal{X} \to \mathcal{Y}$ 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

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

Our contributions are threefold.

1) Assuming that f is differentiable and that its gradient is Lipschitz-continuous, we provide an iterative algorithm for solving (2). 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 with its roots in recents algorithms by Vũ [3] and Condat [4].

2) Based on our previous work [5], we introduce the idea of stochastic coordinate descent on Krasnosel'skii-Mann iterations. Interestingly, ADMM+ as well as many other algorithms (gradient descent, proximal gradient algorithm, ADMM, etc.) are special instances of Krasnosel'skii-Mann iterations [6]. 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. Stochastic coordinate descent has been recently investigated in the case of proximal gradient in [7]–[9].

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: i) we introduce a new stochastic approximation algorithm by applying stochastic coordinate descent on the top of ADMM+; ii) we propose a new asynchronous distributed optimization algorithm.

The first two authors are with the CNRS LTCI; Telecom ParisTech, Paris, France. The third author is with Supelec, Gif-sur-Yvette, France. E-mails: `pascal.bianchi, walid.hachem@telecom-paristech.fr, franck.iutzeler@supelec.fr`. This work was partially granted by the French Defense Agency (DGA) grant ODISSEE, by the Telecom/Eurecom Carnot Institute, and by European Research Council (ERC) Starting Grant sponsored project MORE (Advanced Mathematical Tools for Complex Network Engineering) .

The paper is organized as follows. Section II introduces the ADMM+ algorithm and its relation with Vũ [3] and Condat [4]. In Section III, we provide background on monotone operators and 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 ℓ_1-regularized logistic regression.

II. A Forward Backward Primal Dual Algorithm

A. Problem statement

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 $\mathcal{Y}$, consider the minimization problem

$$\inf_{x\in\mathcal{X}} \left(f(x) + g(x) + h(Mx)\right). \tag{2}$$

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) $g \in \Gamma_0(\mathcal{X})$ *and* $h \in \Gamma_0(\mathcal{Y})$,

(ii) f *is a convex differentiable function on* $\mathcal{X}$*, and its gradient* ∇f *is L-Lipschitz continuous on* $\mathcal{X}$*,* i.e., $\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\|$ *for any* $x, y \in \mathcal{X}$.

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

Assumption 2 *The infimum of Problem* (2) *is attained. Moreover, the following qualification condition holds:*

$$\operatorname{ri}\operatorname{dom} h \cap L(\operatorname{ri}\operatorname{dom} g) \neq \emptyset.$$

The *dual* problem corresponding to the *primal* problem (2) is written

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

where q^* denotes the Legendre-Fenchel transform of a function q. With the assumptions 1 and 2, the classical Fenchel-Rockafellar duality theory [10], [11] shows that

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

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

$$\begin{cases} 0 \in \nabla f(\bar{x}) + \partial g(\bar{x}) + M^*\bar{\lambda} \\ 0 \in -M\bar{x} + \partial h^*(\bar{\lambda}) \end{cases} \tag{3}$$

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

B. The Algorithm

We introduce some scalar parameters $\rho, \tau > 0$ satisfying the following assumption.

Assumption 3 *(i)* $\tau\rho^{-1}\|M\|^2 < 4$
(ii) $\frac{1}{\tau} - \frac{\|M\|^2}{4\rho} \geq L/2$.

For any function $g \in \Gamma_0(\mathcal{X})$ we denote by prox_g its associated proximity operator defined by

$$\operatorname{prox}_g(x) = \arg\min_y g(y) + \frac{1}{2}\|y - x\|^2 .$$

Our *ADMM+* algorithm is described for any initial value (x^0, z^0, λ^0) by the following iterations:

ADMM+

$$x^{k+1} = \operatorname{prox}_{\tau g}\left[x^k - \tau(\nabla f(x^k) + M^*\lambda^k)\right] \tag{4a}$$

$$z^{k+1} = \operatorname{prox}_{\rho h}\left[Mx^{k+1} + \rho\lambda^k\right] \tag{4b}$$

$$\lambda^{k+1} = \lambda^k + \rho^{-1}(Mx^{k+1} - z^{k+1}) . \tag{4c}$$

Theorem 1 *Let Assumptions 1–3 hold true. For any initial value $(x^0, z^0, \lambda^0) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Y}$, apply the above ADMM+ iterations. Then (x^k, λ^k) converges to a primal-dual point $(\bar{x}, \bar{\lambda})$ as $k \to \infty$.*

Proof: The proof is a consequence of Theorem 2 and Lemma 2 given in the next section. ■

III. Monotone Operators

A. Theoretical Background

An operator T on an Euclidean[1] space $\mathcal{Y}$ is a set valued mapping $\mathsf{T} : \mathcal{Y} \to 2^{\mathcal{Y}}$. Its *domain* is the set of $x \in \mathcal{Y}$ such that $\mathsf{T}x$ is non-empty. An operator can be equivalently identified as a subset of $\mathcal{Y} \times \mathcal{Y}$, and we write $(x, y) \in \mathsf{T}$ when $y \in \mathsf{T}x$. Given two operators T_1 and T_2 on $\mathcal{Y}$ and two real numbers α_1 and α_2, the operator $\alpha_1\mathsf{T}_1 + \alpha_2\mathsf{T}_2$ is defined as $\alpha_1\mathsf{T}_1 + \alpha_2\mathsf{T}_2 = \{(x, \alpha_1 y_1 + \alpha_2 y_2) : (x, y_1) \in \mathsf{T}_1, (x, y_2) \in \mathsf{T}_2\}$. The identity operator is denoted by $\mathsf{I}_{\mathcal{Y}} = \{(x, x) : x \in \mathcal{Y}\}$ or simply by I. The inverse of the operator T is $\mathsf{T}^{-1} = \{(x, y) : (y, x) \in \mathsf{T}\}$. The following notation will be convenient in this paper. If $\mathcal{Y}_1$ and $\mathcal{Y}_2$ are two Euclidean spaces and, for any $i, j \in \{1, 2\}^2$, $\mathsf{T}_{ij} : \mathcal{Y}_j \to 2^{\mathcal{Y}_i}$ is a set-valued mapping, we shall refer to

$$\begin{pmatrix} \mathsf{T}_{11} & \mathsf{T}_{12} \\ \mathsf{T}_{21} & \mathsf{T}_{22} \end{pmatrix}$$

as the operator on $\mathcal{Y}_1 \times \mathcal{Y}_2$ such that for any $x = (x_1, x_2) \in \mathcal{Y}_1 \times \mathcal{Y}_2$, $\mathsf{T}x$ is the set of vectors $(y_{11} + y_{12}, y_{21} + y_{22})$ where $y_{ij} \in \mathsf{T}_{ij}x_j$.

An operator T is said *monotone* if

$$\forall\ (x, y), (x', y') \in \mathsf{T},\ \langle x - x', y - y' \rangle \geq 0.$$

A monotone operator is said *maximal* if it is not strictly contained in any monotone operator (as a subset of $\mathcal{Y} \times \mathcal{Y}$). The typical example of a maximal monotone operator is the subdifferential ∂f of a function $f \in \Gamma_0(\mathcal{X})$.

An operator T is said single-valued if $\mathsf{T}x$ is a singleton for any x in its domain. In that case, we identify T with a function $\mathsf{T} : D \to \mathcal{Y}$ where D is the domain of T. For $0 < \alpha \leq 1$, a single-valued operator T *α-averaged* if the following inequality holds for any x, y in D:

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

[1]We refer to [11] 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*.

Lemma 1 (Krasnosel'skii-Mann iterations) [2] *Assume that $T : \mathcal{Y} \to \mathcal{Y}$ is α-averaged and that the set $\operatorname{fix}(\mathsf{T})$ 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 $x^0 \in \mathcal{Y}$, the sequence $(x^k)_{k\in\mathbb{N}}$ recursively defined on $\mathcal{Y}$ by $x^{k+1} = x^k + \eta^k(\mathsf{T}x^k - x^k)$ converges to some point in $\operatorname{fix}(\mathsf{T})$.*

B. Randomized Krasnosel'skii Mann Iterations

Consider the space $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_J$ for some $J \in \mathbb{N}^*$ where for any j, $\mathcal{Y}_j$ is a Euclidean space. Assume that $\mathcal{Y}$ is equipped with the scalar product $\langle x, y \rangle = \sum_{j=1}^J \langle x_j, y_j \rangle_{\mathcal{Y}_j}$ where $\langle\, .\, , .\, \rangle_{\mathcal{Y}_j}$ is the scalar product in $\mathcal{Y}_j$. For $j \in \{1, \ldots, J\}$, we denote by $\mathsf{T}_j : \mathcal{Y} \to \mathcal{Y}_j$ the components of the output of operator $\mathsf{T} : \mathcal{Y} \to \mathcal{Y}$ corresponding to $\mathcal{Y}_j$, we thus have $\mathsf{T}(x) = (\mathsf{T}_1(x), \ldots, \mathsf{T}_J(x))$. We denote by $2^{\mathcal{J}}$ the set of subsets of $\mathcal{J} = \{1, \ldots, J\}$. For any $\kappa \in 2^{\mathcal{J}}$, we define the operator $\hat{\mathsf{T}}^{(\kappa)} : \mathcal{Y} \to \mathcal{Y}$ 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 $(\xi^k)_{k>0}$ 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:

$$\forall j \in \mathcal{J},\ \exists\kappa \in 2^{\mathcal{J}},\ j \in \kappa \text{ and } \mathbb{P}(\xi_1 = \kappa) > 0 . \tag{5}$$

Theorem 2 *Let $\mathsf{T} : \mathcal{Y} \to \mathcal{Y}$ be α-averaged and $\operatorname{fix}(\mathsf{T}) \neq \emptyset$. Assume that for all k, 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 (5) holds. Then, almost surely, the iterated sequence

$$x^{k+1} = x^k + \eta^k(\hat{\mathsf{T}}^{(\xi^{k+1})}(x^k) - x^k) \tag{6}$$

converges to some point in $\operatorname{fix}(T)$.

Proof: Define the operator $\mathsf{U} = (1 - \eta^k)\mathsf{I} + \eta^k\mathsf{T}$ (we omit the index k to simplify notations); 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 (6) 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 $\mathcal{Y}$. Define a new inner product $x \bullet y = \sum_{j=1}^J q_j \langle x_j, y_j \rangle_j$ on $\mathcal{Y}$ 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 $\mathcal{F}^k = \sigma(\xi_1, \ldots, \xi^k)$ we have

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

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 $-\frac{1-\alpha\eta^k}{\alpha\eta^k}\|(\mathsf{I} -$

[2] [11, Proposition 5.15, pp.80]

$\mathsf{U})(x^k)\|^2$. As $\mathsf{I}-\mathsf{U}=\eta^k(\mathsf{I}-\mathsf{T})$, we obtain:

$$\mathbb{E}[\left|\!\left|\!\left|x^{k+1}-x^\star\right|\!\right|\!\right|^2 \mid \mathcal{F}^k] \leq \left|\!\left|\!\left|x^k-x^\star\right|\!\right|\!\right|^2 - \eta^k(1-\alpha\eta^k)\|(\mathsf{I}-\mathsf{T})(x^k)\|^2 \quad (7)$$

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. Since $x^\star$ was chosen arbitrarily, the latter statement can be generalized. Using the same arguments as in [5], one can easily prove that the following holds:

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

Getting back to the inequality (7) and integrating both sides w.r.t. $\mathbb{P}$, we obtain on the otherhand that $\sum_k \eta^k(\frac{1}{\alpha}-\eta^k)\mathbb{E}(\|(\mathsf{I}-\mathsf{T})(x^k)\|^2)<\infty$. This implies that, almost surely, $\sqrt{\eta^k(\frac{1}{\alpha}-\eta^k)}(\mathsf{T}x^k-x^k)\to 0$ which reduces to

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

using the stated hypotheses on sequence η^k. The end of the proof directly follows from **C1** and **C2** by the same arguments as in [5]. ■

C. The ADMM+ algorithm as Krasnosel'skii-Mann iterations

In this paragraph, we consider the ADMM+ algorithm of Section II-B. We show that it can be interpreted as a fixed point algorithm associated with an α-averaged operator T. This result has two consequences. First, it proves Theorem 1. Second, by Theorem 2, it gives the possibility to devise an randomized coordinate descent version of this algorithm.

Assume that the product space $\mathcal{X}\times\mathcal{Y}$ is endowed with a new inner product $\langle\,.\,,\,.\,\rangle_{\mathsf{V}}$ defined as $\langle\zeta,\varphi\rangle_{\mathsf{V}}=\langle\zeta,\mathsf{V}\varphi\rangle$ where $\langle\,.\,,\,.\,\rangle$ stands for the natural inner product on $\mathcal{X}\times\mathcal{Y}$ and where

$$\mathsf{V}=\begin{pmatrix}\frac{1}{\tau}\mathsf{I}_{\mathcal{X}} & -\frac{1}{2}M^* \\ -\frac{1}{2}M & \rho\mathsf{I}_{\mathcal{Y}}\end{pmatrix}.$$

We denote by $\mathcal{H}_{\mathsf{V}}$ the corresponding Euclidean space.

Lemma 2 *Let Assumptions 1–3 hold true. For some $\alpha\in[0,1)$, there exists an α-averaged operator T on $\mathcal{H}_{\mathsf{V}}$ such that the iterations of the ADMM+ algorithm verify $(x^k,\lambda^{k+1})=\mathsf{T}(x^{k-1},\lambda^k)$ for any $k\geq 1$.*

Proof: The proof is provided in [12] and relies itself on a recent result of Vũ [3] and Condat [4]. ■

IV. Stochastic Minibatch Algorithm

A. Problem Setting

Given an integer $N>1$, consider the problem of minimizing a sum of functions

$$\inf_{x\in\mathcal{X}}\sum_{n=1}^N(f_n(x)+g_n(x)) \quad (8)$$

where we make the following assumption:

Assumption 4 *For each n,*

(i) f_n is a convex differentiable function on $\mathcal{X}$, and its gradient ∇f_n is L-Lipschitz continuous on $\mathcal{X}$,
(ii) $g_n\in\Gamma_0(\mathcal{X})$.
(iii) The infimum of Problem (8) is attained.
(iv) $\cap_{n=1}^N$ridomg$_n\neq\emptyset$

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 minibatches approaches consist in splitting the data set into N chunks and to process each chunk sequentially, one at a time. The quantity $f_n(x)+g_n(x)$ measures the inadequacy between the model (represented by parameter x) and the nth 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 ℓ_1-norm reduces to the popular lasso problem [13].

B. Instanciating the ADMM+ Algorithm

We derive our stochastic minibatch algorithm as an instance of the ADMM+ algorithm coupled with randomized coordinate descent. To that end let us first rephrase Problem (8) as

$$\inf_{x\in\mathcal{X}^N}\sum_{n=1}^N(f_n(x_n)+g_n(x_n))+\iota_{\mathcal{C}}(x) \quad (9)$$

where the notation x_n represents the nth component of any $x\in\mathcal{X}^N$, ι_A is the characteristic function of a set A (equal to one on A and to zero 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 product 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 $M=\mathsf{I}_N$ the identity matrix. Straightforward application of the ADMM+ algorithm to (9) leads to the following iterations:

$$x_n^{k+1}=\text{prox}_{\tau g}\left[x_n^k-\tau(\nabla f_n(x_n^k)+\lambda_n^k)\right] \quad (10a)$$

$$z^{k+1}=\text{proj}_{\mathcal{C}}\left[x^{k+1}+\rho\lambda^k\right] \quad (10b)$$

$$\lambda_n^{k+1}=\lambda_n^k+\rho^{-1}(x_n^{k+1}-z_n^{k+1}) \quad (10c)$$

where $\text{proj}_{\mathcal{C}}$ is the Euclidean projection onto $\mathcal{C}$. Note that for any $x\in\mathcal{X}^N$, $\text{proj}_{\mathcal{C}}(x)$ is equal to $(\bar{x},\cdots,\bar{x})$ where $\bar{x}$ is the average of vector x *i.e.* $\bar{x}=N^{-1}\sum_n x_n$. As a consequence, components of z^{k+1} are equal and coincide with $\bar{x}^{k+1}+\rho\bar{\lambda}^k$ where $\bar{x}^{k+1}$ and $\bar{\lambda}^k$ are the average of x^{k+1} and λ^k respectively. Using (10c), the latter equality simplifies even further by noting that $\bar{\lambda}^{k+1}=0$ or, equivalently, $\bar{\lambda}^k=0$ for all $k\geq 1$. Finally, for any n and $k\geq 1$, the above iterations reduce to

$$x_n^{k+1}=\text{prox}_{\tau g}\left[x_n^k-\tau(\nabla f_n(x_n^k)+\lambda_n^k)\right] \quad (11a)$$

$$\bar{x}^{k+1}=\frac{1}{N}\sum_{n=1}^N x_n^{k+1} \quad (11b)$$

$$\lambda_n^{k+1}=\lambda_n^k+\rho^{-1}(x_n^{k+1}-\bar{x}^{k+1}). \quad (11c)$$

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 function (f_n,g_n) per iteration. This can be achieved by applying the idea of randomized coordinate descent on top of the ADMM+ algorithm.

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 by applying randomized coordinate descent on the above iterations (11a)-(11b)-(11c).

```
SMPD Algorithm:
Initialization: $(x^0,\lambda^0)$ s.t. $\sum_n \lambda_n^0=0$
Do
    Select a random batch $n^k$ in $\{1,\ldots,N\}$
    For $n=n^k$, set $x_n^{k+1},\bar{x}^{k+1},\lambda_n^{k+1}$ by (11a)-(11b)-(11c)
    For all $m\neq n^k$, set $x_m^{k+1}=x_m^k$ and $\lambda_m^{k+1}=\lambda_m^k$
    $k\leftarrow k+1$
```

```
until happy
Output: $\bar{x}^{k+1}$
```

Assumption 5 *The random sequence n^k is iid and satisfies $\mathbb{P}[n^1 = m] > 0$ for all $m = 1, ..., N$.*

Theorem 3 *Let Assumptions 4 and 5 hold true. Let $\bar{x}^k$ be the output of the SMPD algorithm at iteration k. For any initial values (x^0, λ^0), there exists almost surely a minimizer x^* of (8) such that $\bar{x}^k$ converges to x^*.*

Proof: As the SMPD algorithm is an instance of the ADMM+ algorithm coupled with randomized coordinate descent, it can obviously be seen as randomized Krasnosel'skii-Mann iterations applied to the operator T of Lemma 2. It is easy to check that batch selection sequence $(n^k)_{k>0}$ leads to randomized iterations of T verifying the condition in Eq. (5). Thus, Theorem 2 gives us the claimed result. ■

V. Distributed Optimization

Consider a set of $N > 1$ computing agents that cooperate to solve the minimization problem (8). Here, f_n, g_n are two private cost functions available at Agent n. 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 this agent's private cost 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 cost function of problem (8) 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 [14], [15], in resource allocation problems for communication networks [16], or in statistical estimation problems by sensor networks [17], [18].

A. Network Model

We represent the network as a graph $G = (V, E)$ where $V = \{1, \dots, N\}$ is the set of agents/nodes and $E \subset \{1, \dots, N\}^2$ 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.*

B. Problem Reformulation

In order to formulate a distributed optimization problem leading to fully decentralized algorithm, we a introduce a set V' of $|E|$ *virtual* nodes, each of them corresponding to an edge in E. As opposed to virtual nodes, the elements of V will be referred to as *physical* nodes and we will note $\mathcal{V} = V \bigcup V'$. We refer to Figure 1 for an illustration. We endow $\mathcal{V}$ with a set of edges $\mathcal{E}$ which are represented by the green segments in Figure 1 that is, for each edge $\{n, m\} \in E$ in the initial graph, we generate two edges $\{n, v\}$ and $\{v, m\}$ where v stands for the virtual node between n and m. In particular, $\mathcal{E}$ has cardinality $2|E|$. To be more formal, let $\varphi : E \to V'$ is the bijection associating each edge of the initial graph to the virtual node on that edge. Then $\mathcal{E}$ is the set of couples $\{n, \varphi(\{n, m\})\}$ for all $n \sim m$.

Let us introduce some notations. For any $x \in \mathcal{X}^{\mathcal{V}}$, we denote by x_v the components of x *i.e.*, $x = (x_v)_{v \in \mathcal{V}}$. We introduce the function f and g on $\mathcal{X}^{\mathcal{V}} \to (-\infty, +\infty]$ as $f(x) = \sum_{n \in V} f_n(x_n)$ and $g(x) = \sum_{n \in V} g_n(x_n)$. Note that the sum is done over the set of physical nodes V and not over the extended set $\mathcal{V}$. Otherwise stated, $f(x)$ and $g(x)$ depend on x only through the components

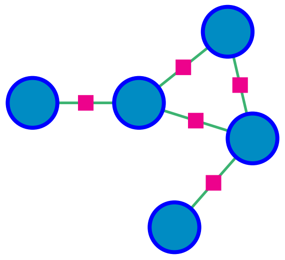
Fig. 1. Extended graph: the round nodes correspond to V and the square ones correspond to the additional set V'.

of x corresponding to the physical nodes. Clearly, Problem (8) 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 extended graph $(\mathcal{V}, \mathcal{E})$. We replace the global consensus constraint by a modified version of the function $\iota_{\mathcal{C}}$ (introduced in Eq. 9). Our goal will be to ensure global consensus through local consensus over every edge of the graph.

For any $\epsilon \in \mathcal{E}$, say $\epsilon = \{n, n'\} \in V \times V'$, we define the linear operator $M_\epsilon : \mathcal{X}^{|\mathcal{V}|} \to \mathcal{X}^2$ as $M_\epsilon(x) = (x_n, x_{n'})$. We contruct the linear operator $M : \mathcal{X}^{|\mathcal{V}|} \to \mathcal{Y} \triangleq (\mathcal{X}^2)^{|\mathcal{E}|}$ as $Mx = (M_\epsilon(x))_{\epsilon \in \mathcal{E}}$. We define $\mathcal{C}_2 = \{(x, x) : x \in \mathcal{X}\}$. Finally, we define $h : \mathcal{Y} \to (-\infty, +\infty]$ for any $y = (y_\epsilon)_{\epsilon \in \mathcal{E}}$ as

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

We consider the following problem:

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

Lemma 3 *Let Assumptions 4 and 6 hold true. The minimizers of (12) are the tuples $(x^*, \dots, x^*)$ where x^* is any minimizer of (8).*

C. Instantiating the ADMM+ Algorithm

We now instantiate the ADMM+ algorithm to the problem (12) – only replacing $\mathcal{X}$ by $\mathcal{X}^{\mathcal{V}}$ in (2). For a given $\epsilon = \{n, n'\} \in \mathcal{E}$, the component λ_ϵ of any vector $\lambda \in \mathcal{Y}$ will be represented by $\lambda_\epsilon = (\lambda_\epsilon(n), \lambda_\epsilon(n'))$.

As the newly defined function h is separable between the $(y_\epsilon)_{\epsilon \in \mathcal{E}}$, the computation of its $\mathrm{prox}_{\rho h}$ can also be split this way. In particular, each component z_ϵ^{k+1} has the form $(\bar{z}_\epsilon^{k+1}, \bar{z}_\epsilon^{k+1})$ where $\bar{z}_\epsilon^{k+1}$ is equal to $(x_n^{k+1} + x_{n'}^{k+1})/2 + \rho(\lambda_\epsilon^k(n) + \lambda_\epsilon^k(n'))/2$ for any $\epsilon = \{n, n'\} \in \mathcal{E}$. Plugging this equality in (4b), we obtain $\lambda_\epsilon^k(n) + \lambda_\epsilon^k(n') = 0$. Therefore, for any $k \geq 1$, $\bar{z}_\epsilon^{k+1} = (x_n^{k+1} + x_{n'}^{k+1})/2$. Moreover, $\lambda_{n,m}^{k+1}(n) = \lambda_{n,m}^k(n) + (x_n^{k+1} - x_{n'}^{k+1})/(2\rho)$. In order to explicit (4a), remark that for any $n \in V$ and for any $\lambda \in \mathcal{Y}$, the nth component of $M^*\lambda$ is equal to $(M^*\lambda)_n = \sum_{n' : \{n,n'\} \in \mathcal{E}} \lambda_{\{n,n'\}}(n)$. As a consequence, (4a) implies that for any physical node $n \in V$,

$$x_n^{k+1} = \mathrm{prox}_{\tau g_n}\left[x_n^k - \tau \left(\nabla f_n(x_n^k) + \sum_{n' : \{n,n'\} \in \mathcal{E}} \lambda_{n,n'}^k(n) \right) \right] .$$

On the otherhand, for any virtual node $n' \in V'$, $(M^*\lambda^k)_{n'} = \sum_{n : \{n,n'\} \in \mathcal{E}} \lambda_{\{n,n'\}}^k(n')$. Because any virtual node $n' = \varphi(\{n, m\})$ is connected to two nodes n and m, the latter sum simplifies to $(M^*\lambda^k)_{n'} = \lambda_{\{n,n'\}}^k(n') + \lambda_{\{m,n'\}}^k(n') = -\lambda_{\{n,n'\}}^k(n) - \lambda_{\{m,n'\}}^k(m)$. As a consequence, $x_{n'}^{k+1} = x_{n'}^k +$

$\tau\left(\lambda^k_{\{n,n'\}}(n) + \lambda^k_{\{m,n'\}}(m)\right)$. We are now in position to state the algorithm. To lighten notations, we introduce the notation $\Lambda^k_n(m) = \lambda^k_{\{n,\varphi(\{n,m\})\}}(n)$ for any $\{n,m\} \in E$. We also introduce the notation $x^k_{\{n,m\}}$ instead of $x^k_{\varphi(\{n,m\})}$. We obtain the following update equations.

$$x^{k+1}_n = \mathrm{prox}_{\tau g_n}\left[x^k_n - \tau\left(\nabla f_n(x^k_n) + \sum_{m\sim n}\Lambda^k_n(m)\right)\right] \quad (13a)$$

$$x^{k+1}_{\{n,m\}} = x^k_{\{n,m\}} + \tau\left(\Lambda^k_n(m) + \Lambda^k_m(n)\right) \quad (13b)$$

$$\Lambda^{k+1}_n(m) = \lambda^k_n(m) + (x^{k+1}_n - x^{k+1}_{\{n,m\}})/(2\rho)\,. \quad (13c)$$

D. A Decentralized Stochastic Primal Dual algorithm

The proposed *Decentralized Stochastic Primal Dual* algorithm (DSPD) is obtained by applying randomized coordinate descent on the above iterations (13a)-(13b)-(13c).

```
DSPD Algorithm:
Initialization: $(x^0, \Lambda^0)$
Do
    Select a random set of nodes $\mathcal{A}^k \subseteq V$
    For each node $n \in \mathcal{A}^k$, do
        Update $x^{k+1}_n$ by (13a)
        Update $x^{k+1}_{\{n,m\}}, \Lambda^{k+1}_n(m)$ by (13b)-(13c), $\forall m \sim n$
        Send $x^{k+1}_{\{n,m\}}$ and $\Lambda^{k+1}_n(m)$ to nodes $m \sim n$
    Leave all other variables unmodified.
    $k \leftarrow k+1$
until happy
Output: $(x^{k+1}_n)_{n\in V}$
```

Assumption 7 *The set of active agents $\mathcal{A}^k$ forms an iid sequence on 2^V such that for any n, $\mathbb{P}[n \in V^1] > 0$.*

Theorem 4 *Let Assumptions 4 6 and 7 hold true. Let $(x^{k+1}_n)_{n\in V}$ be the output of the DSPD algorithm. For any initial values (x^0, Λ^0), there exists almost surely a minimizer x^* of (8) such that for all $n \in V$, x^k_n converges to x^*.*

Proof: The proof follows the same reasoning as the one of Theorem 3. ■

VI. Numerical illustrations

Problem. We address the problem of ℓ_1-regularized logistic regression. Denoting by m the number of observations and by p the number of features, the optimization problem writes

$$\min_{\mathbf{x}\in\mathbb{R}^p} \frac{1}{m}\sum_{t=1}^{m}\log\left(1 + \mathrm{e}^{-y_t \mathbf{a}_t^{\mathrm{T}}\mathbf{x}}\right) + \mu\|\mathbf{x}\|_1 \quad (14)$$

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. Let $(\mathcal{B}_n)_{n=1}^N$ denote a partition of $\{1,\ldots,m\}$. The optimization problem then writes

$$\min_{\mathbf{x}\in\mathbb{R}^p} \sum_{n=1}^{N}\sum_{t\in\mathcal{B}_n}\frac{1}{m}\log\left(1 + \mathrm{e}^{-y_t \mathbf{a}_t^{\mathrm{T}}\mathbf{x}}\right) + \mu\|\mathbf{x}\|_1 \quad (15)$$

or, splitting the problem between the batches

$$\min_{\mathbf{x}\in\mathbb{R}^{Np}} \sum_{n=1}^{N}\left(\sum_{t\in\mathcal{B}_n}\frac{1}{m}\log\left(1 + \mathrm{e}^{-y_t \mathbf{a}_t^{\mathrm{T}}\mathbf{x}_n}\right) + \frac{\mu}{N}\|\mathbf{x}_n\|_1\right) + \iota_{\mathcal{C}}(\mathbf{x}) \quad (16)$$

where $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_N)$ is now in $\mathbb{R}^{Np}$. Obviously Problems (14), (15) and (16) are equivalent and Problem (16) is in the form of (9).

Dataset. In the whole section, we will perform our simulations on the classical `covtype` dataset available from the LIBSVM website[3]. 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 communicates with its neighbors. In this setup, we apply the DSPD algorithm described in Section V.

A. Minibatch

Setup. Consider Problem (15), the goal of minibatch processing is to find a minimum of this problem by using one randomly selected batch per iteration. We processed the `covtype` dataset through 581 batches of 1000 observations and setted the regularization parameter μ to 10^{-3}.

Compared Algorithms. As our goal is to show the performance of randomized optimization algorithms derived from ADMM+, we compare ourselves with algorithms using a gradient step for the data fitting function and we do not consider here acceleration techniques nor stepsize selection procedures. The considered algorithms are:

- SGD: the stochastic (sub-)gradient descent (see [19] and references therein) applied to Problem (15) with $1/k$ stepsize.
- MISO: the MISO algorithm for composite optimization [20], [21] applied to Problem (15) with $1/L$ stepsize, where L is set to the maximum of the upper bounds of the Lipschitz constants per batch $L = 0.25\max_{n=1,\ldots,N}\|\mathbf{a}_n\|_2^2$.
- SMPD: our SMPD algorithm described above with parameters ρ and τ set manually for good performance while satisfying the condition of Theorem 3.

In Figure 2, we plot the ℓ_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[4] 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.

B. Distributed Optimization

Setup. Now, we consider the case where the dataset is scattered over the network depicted in Fig. 1. More precisely, each agent has 500 observations taken in the `covertype` dataset, it randomly activates, processes its data, and sends information to its neighbors. Here, no coordinator/fusion center is present to collect or manage data.

$$\min_{\mathbf{x}\in\mathbb{R}^{Np}} \sum_{n=1}^{N}\left(\sum_{t\in\mathcal{B}_n}\frac{1}{m}\log\left(1 + \mathrm{e}^{-y_t \mathbf{a}_t^{\mathrm{T}}\mathbf{x}_n}\right) + \frac{\mu}{N}\|\mathbf{x}_n\|_1\right) + \sum_{\epsilon\in E}\iota_{\mathcal{C}_2}(y_\epsilon). \quad (17)$$

[3] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

[4] the performance of stochastic optimizations methods for learning depends vastly on the choice of the stepsizes. In this paper, we do not provide illustrations and methods for different stepsize selections as our focus is on the optimization algorithm itself.

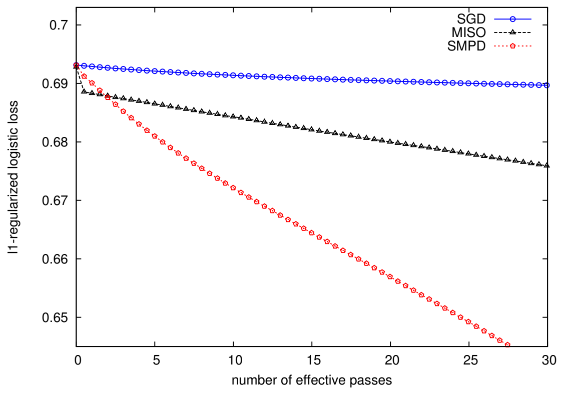

Fig. 2. Comparison of minibatch algorithms.

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:

- DSGD: the distributed stochastic (sub-)gradient descent (see [18], [22] and references therein) applied to Problem (15) with $1/n$ stepsize. For fairness in the comparison in terms of communications, we used the Broadcast Gossip Algorithm [23] as an exchange algorithm.
- DSPD: our DSPD algorithm described above with parameters ρ and τ set manually for good performance while satisfying the condition of Theorem 4.

In Figure 3, we plot the ℓ_1-regularized logistic loss at the agent at the left versus the number of iterations (*i.e.* the number of agent activations). We observe that the DSPD is significantly quicker than the distributed stochastic subgradient. In particular, this is due to the fact that the DSPD 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.

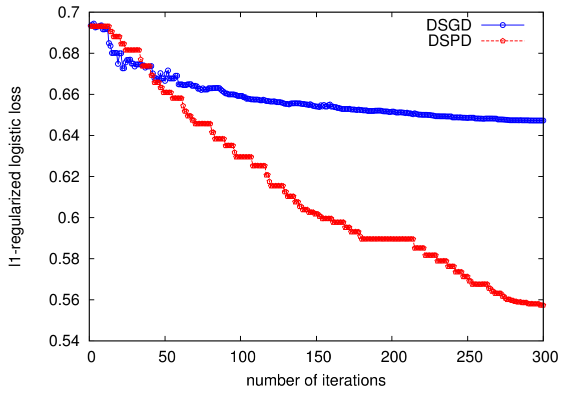

Fig. 3. Comparison of distributed optimization algorithms.

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. Conclusion and Perspectives

In this paper, starting from the general ADMM+ algorithm, we derived randomized optimization algorithms suited for the minimization of the sum of three functions, one of them being smooth. These algorithms are perfectly suited for learning of big datasets, possibly scattered over a network.

Future works include a precise study of the convergence rate of these algorithms along with efficient stepsizes strategies.

References

[1] 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.

[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. Vũ, “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] 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.

[6] 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.

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

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

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

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

[11] 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.

[12] P. Bianchi and W. Hachem, “A Primal-Dual algorithm for Distributed Optimization,” submitted, 2014.

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

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

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

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

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

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

[23] 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.