A Stochastic Primal-Dual algorithm for Distributed Asynchronous Composite Optimization

Pascal Bianchi, Walid Hachem
Telecom ParisTech; CNRS LTCI
Paris, France
{bianchi,hachem}@telecom-paristech.fr

Franck Iutzeler
LANEAS group; Supélec
Gif-sur-Yvette, France
franck.iutzeler@supelec.fr

***Abstract*—Consider a network where each agent has a private composite function (*e.g.* the sum of a smooth and a non-smooth function). The problem we address here is to find a minimizer of the aggregate cost (the sum of the agents functions) in a distributed manner. In this paper, we combine recent results on primal-dual optimization and coordinate descent to propose an asynchronous distributed algorithm for composite optimization.**

***Index Terms*—Distributed optimization, Consensus algorithms, Primal-Dual algorithm, Coordinate Descent.**

I. Introduction

Consider a network of N agents seeking to solve the optimization problem

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

where $\mathcal{X}$ is a Euclidean space (typically $\mathbb{R}$ or $\mathbb{R}^K$ for some $K > 1$) and where f_n, g_n are two private cost functions available at agent n. This formulation based on composite functions enables us to split agent n's cost function into a smooth (*e.g.* quadratic distance, logit) and a non-smooth (*e.g.* ℓ_1 regularization, projection) part.

More precisely, denoting by $\Gamma_0(\mathcal{X})$ the set of proper lower semi-continuous convex functions on $\mathcal{X} \to \overline{\mathbb{R}} \triangleq \mathbb{R}\cup\{+\infty\}$, we make the following assumptions.

Assumption 1. For any $n = 1 \dots N$,

i) $f_n, g_n \in \Gamma_0(\mathcal{X})$;
ii) f_n is differentiable on $\mathcal{X}$;
iii) the gradient ∇f_n of f_n is L-lipschitz continuous on $\mathcal{X}$ *i.e.*, $\|\nabla f_n(x) - \nabla f_n(y)\| \leq L\|x-y\|$ for any $x, y \in \mathcal{X}$.

If there was only one agent, a typical algorithm for finding the minimum of $f_1 + g_1$ is the *Proximal gradient* algorithm [1]. This algorithm is based on the iteration

$$x^{k+1} = \mathrm{prox}_{\gamma g_1}(x^k - \gamma\nabla f_1(x^k)) \tag{2}$$

where $\gamma > 0$ is a fixed stepsize and where prox denotes the *proximity operator* defined as $\mathrm{prox}_h(x) = \arg\min_y h(y) + \frac{1}{2}\|y-x\|^2$ for $h \in \Gamma_0(\mathcal{X})$.

In a networked context, finding a minimizer of the aggregate cost function often imply to use distributed optimization algorithms. These algorithms are based on i) local computations based on the agents' private functions; and ii) local exchanges over some underlying communication network. Their goal is to reach a *consensus*, that is a state where each agent in the network share the same value, over the sought minimizer of the aggregate cost. In addition, in order to add robustness and flexibility, distributed optimization algorithms can be rendered asynchronous/randomized by making only a random set of agents perform computations and/or communications at each iteration.

In the case where the agents functions are smooth (*i.e.* if $\forall n,\ g_n \equiv 0$ in Problem (1)), distributed gradient algorithms are often advocated (see *e.g.* [4], [5], [7], [25]). They consist in two steps: first, a local gradient descent during which some/all agents update their local estimate by performing a gradient descent on their own function weighted by a stepsize γ^k; then, the agents perform a (possibly random) average gossiping step [9], [10]. One of the drawbacks of these algorithms is that, in general, vanishing stepsizes $(\gamma^k)_k$ have to be used which affects greatly the convergence rates.

In the case where the agents functions are not necessarily smooth (*i.e.* if $\forall n,\ f_n \equiv 0$ in Problem (1)), one of the most popular approaches is to use the distributed Alternating Direction Method of Multipliers (ADMM) [11], [12] in which each agent evaluates the proximity operator of its own function and then combines the output with neighboring agents. Asynchronous distributed optimization with the ADMM have also been recently proposed [13]. Even though it is more computationally requiring, the distributed ADMM offers exponential convergence rates for a large class of functions [12], [14].

While the distributed gradient and ADMM are respectively primal and dual methods, in this paper, we investigate a primal-dual algorithm for distributed composite optimization. This algorithm is such that each iteration can be decomposed in two steps: i) a local computation step where each agent n computes its new estimate by evaluating the gradient of f_n and the proximity operator of g_n in an operation close to Eq. (2), and ii) a communication step where the agents exchange their estimates through some underlying communication network. On top of this algorithm, we perform coordinate descent [16], [17], [13], [18] on well chosen variables to obtain a new asynchronous algorithm we name Distributed Asynchronous Primal Dual algorithm (DAPD) and detail the communication and computation scheme of this new algorithm. Related works include [13], [15], [6], [2], [3].

We believe these ADMM+ based algorithms provide a useful alternative to popular gradient and ADMM-based distributed algorithms as they combine the good convergence and computational[1] properties of the proximity operators of the ADMM with the computational simplicity of the gradient.

First, in Section II, we introduce our distributed composite optimization problem. Then, we present in Section III-A the ADMM+ algorithm, and in Section III-B, we introduce our distributed composite optimization algorithm based on the ADMM+. After that, we

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) .

[1] *e.g.* the proximity operator of ℓ_1 norm is a simple soft-thresholding

introduce the notion of coordinate descent in Section IV and derive our Distributed Asynchronous Primal Dual algorithm in Section V. Finally, Section VI is dedicated to numerical illustrations.

II. Distributed Optimization problem

Let us consider a network of N agents modeled by a graph $G = (V, E)$ where $V = \{1, \dots, N\}$ is the set of nodes/agents and $E \subset \{1, \dots, N\}^2$ is the set of undirected edges. We will write $m \sim n$ when $\{n, m\} \in E$. In practice, $n \sim m$ means that n and m can communicate with each other. Obviously, to be able to minimize the total aggregate cost, the network has to be connected.

Assumption 2. G is connected and has no self loop.

Now, give each agent n two private functions f_n and g_n verifying Assumption 1. The goal of this network is to solve Problem (1), however, this problem does not take into account i) the fact that the function are private and local to one particular agent; and ii) the network (and thus the communication) structure.

To address the first point, let us define new functions f and g on $\mathcal{X}^N$ as

$$\begin{array}{rcccrcc} f : \mathcal{X}^N & \longrightarrow & \overline{\mathbb{R}} & \text{and } g : \mathcal{X}^N & \longrightarrow & \overline{\mathbb{R}} \\ x & \longmapsto & \sum_{n=1}^N f_n(x_n) & x & \longmapsto & \sum_{n=1}^N g_n(x_n) \end{array} \tag{3}$$

where for $x \in \mathcal{X}^N$, we note $x = (x_1, ..., x_N)$ the splitting of x on the product space. Obviously, Problem (1) is equivalent the minimization of $f(x) + g(x)$ under the constraint that all components x are the same.

$$\begin{array}{rl} \inf\limits_{x \in \mathcal{X}^N} & f(x) + g(x) \\ \text{s.t.} & x_1 = ... = x_N \end{array} \tag{4}$$

To address the second point, we use the idea of [19] to reformulate this constraint into an indicator function taking the graph structure into account. The idea is to ensure consensus separately over all the edges of the graph; that way, the constraints are localized but thanks to the network connectivity assumption, this is obviously equivalent to have global consensus over the network. Mathematically, for any $e = \{n, m\} \in E$, let M_e be the linear operator from $\mathcal{X}^N$ to $\mathcal{X}^2$ such that $M_e x = (x_n, x_m)$ for any $x \in \mathcal{X}^N$. Then, define M as the linear operator generated by stacking vertically the $(M_e)_{e \in E}$, M thus goes from $\mathcal{X}^N$ to $\mathcal{Y} \triangleq (\mathcal{X}^2)^{|E|}$. For $x \in \mathcal{X}^N$, $y = Mx \in \mathcal{Y}$ can be split into its $|E|$ components in $\mathcal{X}^2$ such that $y_e = M_e x = (x_n, x_m)$. We will often use this splitting in the following and we will split any variable $y \in \mathcal{Y}$ into $|E|$ chunks[2] such that $e = \{n, m\}$-th element writes $y_e = (y_e(n), y_e(m))$. Finally, define a new function h on $\mathcal{Y}$ such that for $y \in \mathcal{Y}$, $h(y) = 0$ if for every $e \in E$, y_e has the form $[a, a]$ for some $a \in \mathcal{X}$ and $+\infty$ elsewhere.

$$\begin{array}{rcl} h : \mathcal{Y} & \longrightarrow & \overline{\mathbb{R}} \\ y & \longmapsto & \sum_{e \in E} \iota_{\mathcal{C}_2}(y_e) \end{array} \tag{5}$$

where ι_C is the indicator function of set C equal to 0 if its argument is in C and $+\infty$ elsewhere; $\mathcal{C}_2 = \{[a, a] : a \in \mathcal{X}\}$ is the wanted consensus space.

We are finally able to formulate our *distributed composite optimization problem* as

$$\inf_{x \in \mathcal{X}^N} f(x) + g(x) + h(Mx) \ . \tag{6}$$

[2] we suppose some implicit ordering of the links of E and identify an element e in E with its index.

Lemma 1. *Let Assumption 2 hold true. The minimizers of* (6) *are of the form* $(x^\star, ..., x^\star)$ *where* $x^\star$ *is a minimizer of* (1).

Proof: If (1) has a minimizer, then a minimizer $\bar{x}$ of (6) verify $h(M\bar{x}) = 0$. As the network is connected by Assumption 2, we have $\bar{x} = [a, ..., a]$ and plugging this into (6) imply that a is a minimizer of (1). ∎

III. Distributed Optimization with ADMM+

A. ADMM+

The ADMM+ algorithm was introduced in [15] to deal with problems of the form of (6) when f is smooth but not necessarily g nor h (see Assumption 1).

Let us denote by $\langle \cdot, \cdot \rangle$ the inner product on $\mathcal{X}^N$ and by $\| \cdot \|$ the norm on $\mathcal{X}^N$ or $\mathcal{Y}$. For some free hyper-parameters $\rho, \tau > 0$, the ADMM+ consists in the following iterations for solving $\min_x f(x) + g(x) + h(Mx)$.

ADMM+

$$z^{k+1} = \operatorname*{argmin}_{w \in \mathcal{Y}} \left\{ h(w) + \frac{\|w - (Mx^k + \rho\lambda^k)\|^2}{2\rho} \right\} \tag{7a}$$

$$\lambda^{k+1} = \lambda^k + \rho^{-1}(Mx^k - z^{k+1}) \tag{7b}$$

$$u^{k+1} = (1 - \tau\rho^{-1})Mx^k + \tau\rho^{-1} z^{k+1} \tag{7c}$$

$$x^{k+1} = \operatorname*{argmin}_{w \in \mathcal{X}^N} \left\{ g(w) + \langle \nabla f(x^k), w \rangle + \frac{\|Mw - u^{k+1} + \tau\lambda^{k+1}\|^2}{2\tau} \right\} \tag{7d}$$

Assumption 3. The infimum of (1) is attained. Moreover, $\cap_{n=1}^N \operatorname{ri} \operatorname{dom} g_n \neq \emptyset$ where $\operatorname{dom} g$ is the domain of the function g and where $\operatorname{ri} C$ is the relative interior of the set C.

Theorem 1 ([15])**.** *Let Assumptions 1–3 hold true. Assume that* $\tau^{-1} - \rho^{-1} > L/2$. *For any initial value* $(x^0, \lambda^0) \in \mathcal{X}^N \times \mathcal{Y}$, *the sequence* $(x^k)_k$ *defined by ADMM+ converges to a minimizer of* (6) *as* $k \to \infty$.

The proof, provided in [15], is based on recent results on primal-dual algorithms by Vũ [20] and Condat [21].

B. Distributed ADMM+

Now, let us apply ADMM+ to solve Problem (6) with the functions defined in Eqs. (3) and (5) in order to derive a distributed optimization algorithm.

First, let us explicit the proximal operation related to h stressing the fact that h is separable in $|E|$ chunks in $\mathcal{X}^2$.

$$\begin{aligned} \operatorname{prox}_{\rho h}(y) &= \operatorname*{argmin}_{w \in \mathcal{Y}} \left\{ h(w) + \frac{1}{2\rho}\|w - y\|^2 \right\} \\ &= \operatorname*{argmin}_{w \in \mathcal{Y}} \left\{ \sum_{e \in E} \iota_{\mathcal{C}_2}(w_e) + \frac{1}{2\rho}\|w_e - y_e\|^2 \right\} \end{aligned}$$

We remark that the operation can be split: for $e = \{n, m\} \in E$, the e-th chunk of $\operatorname{prox}_{\rho h}(y)$ only depends on y_e and writes

$$\begin{aligned} \big(\operatorname{prox}_{\rho h}(y_e)\big)_e &= \operatorname*{argmin}_{w_e \in \mathcal{X}^2} \left\{ \iota_{\mathcal{C}_2}(w_e) + \frac{1}{2\rho}\|w_e - y_e\|^2 \right\} \\ &= \left(\frac{y_e(n) + y_e(m)}{2}, \frac{y_e(n) + y_e(m)}{2} \right) \end{aligned}$$

thus for any $e=\{n,m\}\in E$, we have

$$z_e^{k+1}(n)=z_e^{k+1}(m)=\frac{x_n^k+x_m^k}{2}+\rho\frac{\lambda_e^k(n)+\lambda_e^k(m)}{2}$$

by definition of operator M.

Plugging this equality into Eq. (7b) and, for any $e=\{n,m\}\in E$, looking at λ_e^{k+1}, we get that

$$\begin{aligned}\lambda_e^{k+1}(n)+\lambda_e^{k+1}(m)&=\lambda_e^k(n)+\lambda_e^k(m)-2\frac{\lambda_e^k(n)+\lambda_e^k(m)}{2}\\&\quad+\rho^{-1}\left(x_n^k+x_m^k-2\frac{x_n^k+x_m^k}{2}\right)\\&=0\end{aligned}$$

$$\text{and so } z_e^{k+1}(n)=z_e^{k+1}(m)=\frac{x_n^k+x_m^k}{2}$$

$$\text{thus we have } \lambda_e^{k+1}(n)=-\lambda_e^{k+1}(m)=\lambda_e^k(n)+\rho^{-1}\frac{x_n^k-x_m^k}{2}$$

and with Eq. (7c), $u_e^{k+1}(n)=x_n^k-\tau\rho^{-1}\frac{x_n^k-x_m^k}{2}$

Finally, let us concentrate on Eq. (7d) to put it in a proximal gradient form (see Eq. (2)) for each component $n=1,...,N$.

$$\begin{aligned}x^{k+1}&=\operatorname*{argmin}_{w\in\mathcal{X}^N}\left\{\sum_{n=1}^N g_n(w_n)+\langle\nabla f_n(x_n^k),w_n\rangle\right.\\&\quad\left.+\frac{1}{2\tau}\sum_{m:n\sim m}\left\|w_n-u_{\{n,m\}}^{k+1}(n)+\tau\lambda_{\{n,m\}}^{k+1}(n)\right\|^2\right\}\\&=\operatorname*{argmin}_{w\in\mathcal{X}^N}\left\{\sum_{n=1}^N g_n(w_n)+\langle\nabla f_n(x_n^k),w_n\rangle\right.\\&\quad\left.+\frac{d_n}{2\tau}\left\|w_n-\sum_{m:n\sim m}\frac{u_{\{n,m\}}^{k+1}(n)-\tau\lambda_{\{n,m\}}^{k+1}(n)}{d_n}\right\|^2\right\}\\&=\operatorname*{argmin}_{w\in\mathcal{X}^N}\left\{\sum_{n=1}^N g_n(w_n)\right.\\&\quad\left.+\frac{d_n}{2\tau}\left\|w_n+\frac{\tau\nabla f_n(x_n^k)}{d_n}-\sum_{m:n\sim m}\frac{u_{\{n,m\}}^{k+1}(n)-\tau\lambda_{\{n,m\}}^{k+1}(n)}{d_n}\right\|^2\right\}\end{aligned}$$

where d_n is the degree of node n defined as the cardinal of the set of neighbors $\{m:n\sim m\}$. Now, remarking that the above argmin can be done component-wise and reduces to a proximity operator, we obtain for all $n=1,...,N$

$$\begin{aligned}x_n^{k+1}&=\operatorname{prox}_{\tau g_n/d_n}\left(-\frac{\tau}{d_n}\nabla f_n(x_n^k)+\sum_{m:n\sim m}\frac{u_{\{n,m\}}^{k+1}(n)-\tau\lambda_{\{n,m\}}^{k+1}(n)}{d_n}\right)\\&=\operatorname{prox}_{\tau g_n/d_n}\left(\left(1-\tau\rho^{-1}\right)x_n^k-\frac{\tau}{d_n}\nabla f_n(x_n^k)\right.\\&\quad\left.+\frac{\tau}{d_n}\sum_{m:n\sim m}\left(\rho^{-1}x_m^k-\lambda_{\{n,m\}}^k(n)\right)\right)\end{aligned}$$

where the second equality used the previously derived updates for u^{k+1} and λ^{k+1}.

We are now able to explicit our Distributed ADMM+. Remark that in order to perform the previously derived updates at time $k+1$, every agent n must have knowledge of x_n^k, $\{\lambda_{\{n,m\}}^k(n)\}_{m\sim n}$, and $\{x_m^k\}_{m\sim n}$. While the first two are previously computed variables, the knowledge of the neighbors' estimates is also required which means that after each iteration, the sensors must send their estimates to all their neighbors.

Distributed ADMM+

Initialization: (x^0,λ^0) s.t. for all $e=\{n,m\}\in E$, $\lambda_e^0(n)+\lambda_e^0(m)=0$

Do

- For any $n=1,...,N$, agent n performs the following operations:

$$\lambda_{\{n,m\}}^{k+1}(n)=\lambda_{\{n,m\}}^k(n)+\frac{x_n^k-x_m^k}{2\rho}\quad\text{for all } m\sim n \tag{8a}$$

$$\begin{aligned}x_n^{k+1}=\operatorname{prox}_{\tau g_n/d_n}\left(\left(1-\tau\rho^{-1}\right)x_n^k-\frac{\tau}{d_n}\nabla f_n(x_n^k)\right.\\\left.+\frac{\tau}{d_n}\sum_{m:n\sim m}\left(\rho^{-1}x_m^k-\lambda_{\{n,m\}}^k(n)\right)\right)\end{aligned} \tag{8b}$$

- Agent n sends its estimate x_n^{k+1} to its neighbors
- Increment k

The following result is a direct consequence of Theorem 1.

Corollary 1. *Let Assumptions 1–3 hold true. Assume that $\tau^{-1}-\rho^{-1}>L/2$. For any initial value (x^0,λ^0), let $(x^k)_{k\in\mathbb{N}}$ be the sequence produced by the Distributed ADMM+. Then, there exists a minimizer $x^\star$ of Problem* (1) *such that for all $n\in V$, $(x_n^k)_k$ converges to $x^\star$.*

IV. Randomized Coordinate Descent

Let us get back to the ADMM+ algorithm. Looking at the iterations, one can see that only x^k and λ^k are needed to generate all the variables at time $k+1$. Actually, it was shown in [15] that the hypervariable $\zeta^k\triangleq(\lambda^k,Mx^k)$ enabled to generate subsequent values; in addition, it was proven that there was some operator T from $\mathcal{Y}\times\mathcal{Y}$ to $\mathcal{Y}\times\mathcal{Y}$ such that

$$\zeta^{k+1}=\mathsf{T}\zeta^k. \tag{9}$$

Then, T was shown to have some contraction property called *averaging*[3] that ensured that the above iteration converged to a fixed point $\zeta^\star=(\lambda^\star,Mx^\star)$ of T as it boils down to a Krasnoselskiĭ-Mann algorithm [22, Prop 5.15]. This is the core of the proof of Theorem 1 along with the fact that the reached couple $(\lambda^\star,x^\star)$ is a primal-dual optimal point for the considered problem (see [15]).

Now, it was recently proven in [13] that if, instead of performing the whole update in Eq. (9), one only updates randomly chosen coordinates, the iterations still converged to a fixed point of operator T. More precisely, let us split the space $\mathcal{Y}\times\mathcal{Y}$ into a product of J subspaces $\mathcal{H}_1\times...\times\mathcal{H}_J$ and introduce a random i.i.d. subset selection sequence $(\xi^k)_k$ valued in $\{0,1\}^J$. Obviously, we must assume that each subset of coordinates is chosen with some positive probability.

Assumption 4. The subset selection sequence $(\xi^k)_k$ is i.i.d. and for all $j=1,...,J$, $\mathbb{P}[\xi_j^1=1]>0$.

Finally, for κ in $\{0,1\}^J$ define the operator $\hat{\mathsf{T}}^{(\kappa)}$ as $(\hat{\mathsf{T}}^{(\kappa)}\zeta)_j=(\mathsf{T}\zeta)_j$ if $\kappa_j=1$ and $(\hat{\mathsf{T}}^{(\kappa)}\zeta)_j=\zeta_j$ elsewhere. Obviously, $(\hat{\mathsf{T}}^{(\xi^k)})_k$ is the wanted sequence of randomly updated operators. The following convergence theorem is a simple extension of [13, Th. 2].

Theorem 2. *Let $\mathsf{T}:\mathcal{H}\to\mathcal{H}$ be an averaged operator and $\operatorname{fix}(\mathsf{T})\neq\emptyset$. Let $(\xi^k)_k$ verify Assumption 4. Then the iterated sequence*

$$\zeta^{k+1}=\hat{\mathsf{T}}^{(\xi^{k+1})}\zeta^k \tag{10}$$

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

[3] $\mathsf{T}:\mathcal{H}\to\mathcal{H}$ is *averaged* if there is $\alpha\in]0,1[$ such that for $x,y\in\mathcal{H}$, $\|\mathsf{T}x-\mathsf{T}y\|^2\leq\|x-y\|^2-\frac{1-\alpha}{\alpha}\|x-\mathsf{T}x-(y-\mathsf{T}y)\|^2$

V. A Distributed Asynchronous Primal Dual Algorithm

Starting from ADMM+ and functions defined in Section II, our goal now is to design an asynchronous distributed algorithm *i.e.* an algorithm where the computations and communications are done by some agents chosen at random. Theorem 2 tells us that when applying ADMM+, one can keep only the updates related to some agents as long as the selection is i.i.d. and that every agent can be selected.

First, let us remark that $x_n^k, \lambda_{\{n,m\}}^k$ are owned by agent n before time $k+1$ in Distributed ADMM+. Thus, let us define a group $\mathcal{V}^{k+1}$ of active agents for time $k+1$ and compute their update in the same manner as for Distributed ADMM+ in Section III-B but now only the components related to the agents of $\mathcal{V}^{k+1}$ (*i.e.* x_n^{k+1} and $\lambda_{\{n,m\}}^{k+1}(n)$ for $n \in \mathcal{V}^{k+1}$ and $m \sim n$) are updated, the others remain still. This means that some simplifications as the fact that $\lambda_e^{k+1}(n)+\lambda_e^{k+1}(m) = 0$ for $e = \{n,m\} \in E$ are not valid anymore. By looking at the derivations of Section III-B in the same order, we still have for $n \in \mathcal{V}^{k+1}$ and $m \sim n$

$$z_{\{n,m\}}^{k+1}(n) = \frac{x_n^k + x_m^k}{2} + \rho \frac{\lambda_{\{n,m\}}^k(n) + \lambda_{\{n,m\}}^k(m)}{2}$$

but not $\lambda_{\{n,m\}}^{k+1}(n) + \lambda_{\{n,m\}}^{k+1}(m) = 0$ as this rely on the update of $\lambda_{\{n,m\}}^{k+1}(m)$ which does not happen as m is not necessarily in $\mathcal{V}^{k+1}$. Instead, we have

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

Finally, concerning the update of x, the same steps are still valid except the last one where u^{k+1} and λ^{k+1} are replaced, as their simplifications do not hold anymore. We now have $\forall n \in \mathcal{V}^{k+1}, m \sim n$

$$\begin{aligned}x_n^{k+1} &= \text{prox}_{\tau g_n/d_n}\left(-\frac{\tau}{d_n}\nabla f_n(x_n^k) + \sum_{m:n\sim m} \frac{u_{\{n,m\}}^{k+1}(n) - \tau\lambda_{\{n,m\}}^{k+1}(n)}{d_n}\right) \\ &= \text{prox}_{\tau g_n/d_n}\left((1-\tau\rho^{-1})x_n^k - \frac{\tau}{d_n}\nabla f_n(x_n^k)\right. \\ &\qquad \left. + \frac{\tau}{d_n}\sum_{m:n\sim m}\left(\rho^{-1}x_m^k - \lambda_{\{n,m\}}^k(m)\right)\right)\end{aligned}$$

which completes the derivation of the algorithm. One can remark a major difference due to the asynchronism is that now in addition to x_n^{k+1}, an agent $n \in \mathcal{V}^{k+1}$ also has to transmit $\lambda_{\{n,m\}}^{k+1}(m)$ to its neighbors $m \sim n$ at the end of its update.

We are now able to state our *Distributed Asynchronous Primal Dual* algorithm (DAPD), which convergence is a straightforward consequence of Theorem 1 and 2.

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

- Select a random set of agents $\mathcal{V}^{k+1}$, each agent $n \in \mathcal{V}^{k+1}$ performs the following operations:
  - for all $m \sim n$,

$$\lambda_{\{n,m\}}^{k+1}(n) = \frac{x_n^k - x_m^k}{2\rho} + \frac{\lambda_{\{n,m\}}^k(n) - \lambda_{\{n,m\}}^k(m)}{2}$$

$$x_n^{k+1} = \text{prox}_{\tau g_n/d_n}\left((1-\tau\rho^{-1})x_n^k - \frac{\tau}{d_n}\nabla f_n(x_n^k) + \frac{\tau}{d_n}\sum_{m:n\sim m}\left(\rho^{-1}x_m^k - \lambda_{\{n,m\}}^k(m)\right)\right)$$

  - for all $m \sim n$, send $\{x_n^{k+1}, \lambda_{\{n,m\}}^{k+1}(n)\}$ to neighbor m.
- for any agent $n \notin \mathcal{V}^{k+1}$, $x_n^{k+1} = x_n^k$, and $\lambda_{\{n,m\}}^{k+1}(n) = \lambda_{\{n,m\}}^k(n)$ for all $m \sim n$.
- Increment k.

Theorem 3. *Suppose that the nodes selection sequence $(\mathcal{V}^k)_k$ is i.i.d. and verifies for all $n \in V$, $\mathbb{P}[n \in \mathcal{V}^1] > 0$. Let Assumptions 1–3 hold true. Assume that $\tau^{-1} - \rho^{-1} > L/2$. Let $(x_n^{k+1})_{n\in V}$ be the output of the DAPD algorithm. For any initial value (x^0, λ^0) there exists a minimizer $x^\star$ of Problem (1) such that for all $n \in V$, $(x_n^k)_k$ converges almost surely to $x^\star$.*

VI. Numerical illustrations

We now illustrate the performances of the DAPD and compare it to other asynchronous distributed algorithms. As our focus is on composite optimization, we will consider the following distributed group lasso problem:

$$\min_{x\in\mathbb{R}^K} \|Ax-b\|_2^2 + \mu\|x\|_1 = \sum_{n=1}^N \left(\|A_n x - b_n\|_2^2 + \frac{\mu}{N}\|x\|_1\right)$$

where the dimension of the problem is $K = 50$, A is a 250×50 real random matrix and b a size 250 vector equal to $Ax_0 + n$ where x_0 is a random sparse vector and n is an additional Gaussian noise. Matrix A and vector b are split equally line-wise between the $N = 5$ sensors thus $(A_n)_n$ and $(b_n)_n$ are collections of 50×50 matrices and size-50 vectors. Parameter μ is set so that the solution and x_0 have similar sparsity. Finally, the underlying communication network is a connected Random Geometric Graph.

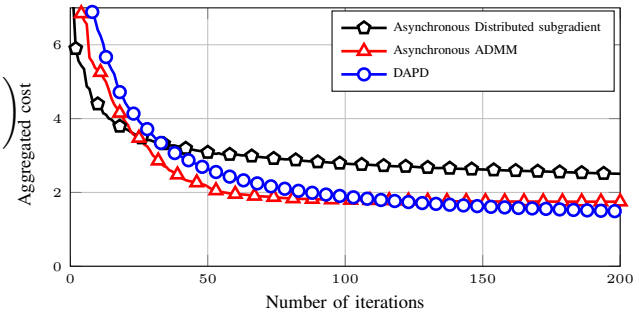

Fig. 1. Functional error of asynchronous distributed optimization algorithms

In Figure 1, we plot the total functional error with respect to the number of iterations for one run of the algorithms. We consider three algorithms: i) the Asynchronous Distributed subgradient algorithm [23], [24] with Random Gossip as an exchange protocol [9]; ii) the Asynchronous Distributed ADMM [13] with the subset taken as the edges pf the graph; and iii) our DAPD algorithm. Out of fairness for the other algorithms, we took the following random activation scheme: at each iteration, one agent activates, wakes up a neighbor, and both agents perform computations and exchanges. We remark that our algorithm offers good performances despite its use of a gradient step to reduce complexity. As mentioned above, this algorithm, contrary to others can be carried out with only one sensor active at each iteration and using asymmetric communications.

VII. Conclusion

We proposed and proved the convergence of an asynchronous distributed primal-dual algorithm that enables a network to efficiently perform minimization of an aggregate cost composed of composite functions.

REFERENCES

[1] 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.

[2] P. L. Combettes and J.-C. Pesquet, " Stochastic Quasi-Fejér Block-Coordinate Fixed Point Iterations with Random Sweeping," *arXiv preprint arXiv:1404.7536*

[3] Pesquet, J. C., and Repetti, "A class of randomized primal-dual algorithms for distributed optimization," arXiv preprint arXiv:1406.6404.

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

[5] G. Morral, P. Bianchi, G. Fort, and J. Jakubowicz, "Distributed stochastic approximation: The price of non-double stochasticity," in *Asilomar Conference on Signals, Systems and Computers*, 2012.

[6] Bianchi, P., Hachem, W., and Iutzeler, F. "A Stochastic Coordinate Descent Primal-Dual Algorithm and Applications to Large-Scale Composite Optimization," arXiv preprint arXiv:1407.0898.

[7] J. Chen and A.H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Transactions on Signal Processing* , vol. 60, no. 8, pp. 4289–4305, Aug 2012.

[8] G. Morral, P. Bianchi, and G. Fort, "Success and failure of adaptation-diffusion algorithms for consensus in multiagent networks," in *IEEE Conference on Decision and Control (CDC)*, 2014.

[9] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory* , vol. 52, no. 6, pp. 2508–2530, 2006.

[10] 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.

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

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

[13] 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)*, 2013.

[14] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, "Optimal parameter selection for the alternating direction method of multipliers (admm): Quadratic problems," *arXiv preprint arXiv:1306.2454*, 2013.

[15] P. Bianchi and W. Hachem, "A primal-dual algorithm for distributed optimization," in *IEEE Conference on Decision and Control (CDC)*, 2014.

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

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

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

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

[20] 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.

[21] 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.

[22] 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.

[23] S. Ram, A. Nedić, and V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *Journal of optimization theory and applications*, vol. 147, no. 3, pp. 516–545, 2010.

[24] 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, Nov 2013.

[25] G. Morral, P. Bianchi, and G. Fort, "Success and Failure of Diffusion-Adaptation Algorithms for Consensus in Multi-Agent Networks," *IEEE Transactions on Signal Processing* (submitted) arXiv:1410.6956