ASYNCHRONOUS ALTERNATING DIRECTION METHOD OF MULTIPLIERS APPLIED TO THE DIRECT-CURRET OPTIMAL POWER FLOW PROBLEM

Azary Abboud[⋆†] *Romain Couillet*[⋆] *Mérouane Debbah*[⋆] *Houria Siguerdidjane*[†]

[⋆] Alcatel-Lucent Chair on Flexible Radio - SUPÉLEC, Gif-sur-Yvette, France
[†]Automatic Control Department - SUPÉLEC, Gif-sur-Yvette, France.

ABSTRACT

We consider a distributed convex optimization problem where each agent has its private convex cost function and controls a set of local variables. We provide an algorithm to carry out a multi-area decentralized optimization in an asynchronous fashion, obtained by applying random Gauss-Seidel iterations on the Douglas-Rachford splitting operator. As an application, a Direct-current linear optimal power flow model is implemented and simulations confirm the convergence of the proposed algorithm.

Index Terms— convex optimization, operator splitting, distributed control, optimal power flow

I. INTRODUCTION

In this paper, we focus on power grid networks and specifically on the direct current optimal power flow (DC-OPF) problem [1]. The network contains a set of N agents that control their own generated powers. These agents aim at minimizing the global generation cost in a distributed manner. However, this minimization is constrained by the grid infrastructure and the physical limits on the available power. We formulate this problem generically as a convex optimization problem with linear constraints as follows

$$\begin{aligned} \underset{\{\boldsymbol{x}_v,\, v\in\{1,\dots,N\}\}}{\text{minimize}} \quad & \sum_{v=1}^{N} f_v(\boldsymbol{x}_v) \\ \text{subject to} \quad & \boldsymbol{a}_i^T(\boldsymbol{x}_1^T,\dots,\boldsymbol{x}_N^T)^T = b_i \quad i=1,\dots,m, \\ & \boldsymbol{x}_{v,min} \le \boldsymbol{x}_v \le \boldsymbol{x}_{v,max} \quad v=1,\dots,N. \end{aligned} \tag{1}$$

where each node $v\in\{1,\dots,N\}$ controls a vector $\boldsymbol{x}_v=(x_{(\sum_{\vartheta=1}^{v-1} n_\vartheta+1)},\dots,x_{(\sum_{\vartheta=1}^{v} n_\vartheta)})^T\in\mathbb{R}^{n_v}$, bounded by $\boldsymbol{x}_{v,min}\in\mathbb{R}^{n_v}$ and $\boldsymbol{x}_{v,max}\in\mathbb{R}^{n_v}$. The local operating cost of a node v is given by the closed proper[1] convex function $f_v:\mathbb{R}^{n_v}\longmapsto\mathbb{R}$. Each equality constraint i, $i\in\{1,\dots,m\}$, is characterized by the vector of coefficients $\boldsymbol{a}_i=(a_{i1},\dots,a_{ij},\dots,a_{in})^T\in\mathbb{R}^n$, with $n=\sum_{v=1}^{N} n_v$, and $b_i\in\mathbb{R}$. Unless stated otherwise, all the vectors are supposed to be column vectors.

To solve the problem distibutively, we decompose the overall system into L smaller areas/micro grids. Each area has its own subproblem and it seeks to update its variables while having limited coordination with the other areas. Several mathematical methods can be used to perform the distributed optimization, such as the Lagrangian relaxation technique [2], the augmented Lagrangian technique [3], the auxiliary problem principle [4] and the approximate Newton directions [5], [6]. These iterative methods usually require the computation of the Hessian of the global objective function, the inversion of large matrices, synchronization and coordination between the areas and may have convergence issues [7]. The Alternating Direction Method of Multipliers (ADMM) [8], [9], in which the augmented Lagrangian of the problem is recursively minimized first with respect to the primal variables and then with respect to the dual variables, converges faster than the aforementioned methods and overcomes all their problems, except the synchronization and the coordination issues [9].

Synchronization induces latency in the computation of the solution because all the areas must wait for the slowest area to solve its subproblem before carrying out another iteration. Moreover, neighboring areas need to coordinate and communicate the variables coupling them. For these reasons, relying on the recent results of [10], we derive an asynchronous distributed algorithm with guaranteed convergence. The work presented in [10] aims at finding the global state of the network by solving an unconstrained optimization problem. All the agents share the same state, which is the same as the global network state. In contrast, in this paper, every node has its own set of primal variables that it seeks to determine, and the global cost of the network is minimized subject to a set of equality and inequality constraints. The update steps of this algorithm inherits the principle of ADMM (i.e., alternating between the resolution of a primal and dual convex optimization problems), but at each iteration, only one area is randomly chosen to solve its subproblem. Since the areas are assumed overlapping, no coordination takes place between them and no inter-area communication is required. We give our algorithm in a generic form, which makes it applicable to problems involving distributed computations other than power systems.

The rest of the paper is organized as follows. We formulate the problem in Section II. In Section III we apply the Douglas-Rachford (DR) splitting method in order to obtain our distributed algorithm. Then, in Section IV we derive the asynchronous distributed algorithm that we prove to converge. We provide an implementation of the DC-OPF problem and simulations in Section V. Section VI concludes the article.

II. PROBLEM FORMULATION

The network is represented by an undirected graph $G=(V,E)$ consisting of a set of N nodes, V, and a set of edges E. We divide G into L overlapping areas A_l, $l\in\{1,\dots,L\}$. To each area A_l we assign a subset of vertices $V_l\subset V$ and a subset of edges $E_l=G(V_l)\subset E$ such that $\bigcup_{l=1}^{L} V_l=V$ and $\bigcup_{l=1}^{L} E_l$ is connected.[2] Let $\boldsymbol{x}=(\boldsymbol{x}_1^T,\dots,\boldsymbol{x}_v^T,\dots,\boldsymbol{x}_N^T)^T\in\mathbb{R}^n$, $\boldsymbol{A}=[\boldsymbol{a}_1,\dots,\boldsymbol{a}_m]^T\in\mathbb{R}^{mn}$, $\boldsymbol{b}=(b_1,\dots,b_m)^T\in\mathbb{R}^m$, $\boldsymbol{x}_{min}=(\boldsymbol{x}_{1,min}^T,\dots,\boldsymbol{x}_{N,min}^T)^T\in\mathbb{R}^n$ and $\boldsymbol{x}_{max}=(\boldsymbol{x}_{1,max}^T,\dots,\boldsymbol{x}_{N,max}^T)^T\in\mathbb{R}^n$. We convert problem (1) into

[1]i.e., $\exists\, x_v\in;\ f_v(x_v)<+\infty\ \&\ \forall x_v\in\mathbb{R}^{n_v},\ f_v(x_v)>-\infty$.

[2]$G(V_l)$ is the graph (V_l,E_l) with $E_l=\{\{u,v\};(u,v)\in V_l^2\}\cap E$.

the following canonical form

$$\begin{aligned} &\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \quad f(\boldsymbol{x})+g(\boldsymbol{z}) \\ &\text{subject to} \quad \boldsymbol{M}\boldsymbol{x}=\boldsymbol{z}, \end{aligned} \tag{2}$$

where f and g are two closed proper convex functions given by

$$f(\boldsymbol{x})=\begin{cases} \sum\limits_{v\in V} f_v(\boldsymbol{x}_v) & \text{if } \boldsymbol{x}_{v,min}\le\boldsymbol{x}_v\le\boldsymbol{x}_{v,max},\ \forall v\in V \\ +\infty & \text{otherwise,} \end{cases}$$

$$g(\boldsymbol{z})=\begin{cases} 0 & \text{if } \sum\limits_{j=1}^{n} z_{ij}=b_i,\quad \forall i=1,\dots,m \\ +\infty & \text{otherwise,} \end{cases}$$

where $\boldsymbol{z}=(\boldsymbol{z}_1^T,\dots,\boldsymbol{z}_l^T,\dots,\boldsymbol{z}_L^T)^T$, and $\boldsymbol{z}_l=\prod_{V_l}\boldsymbol{z}'$ is the projection of $\boldsymbol{z}'=(z_{11},z_{12},\dots,z_{ij},\dots,z_{mn})^T$ on V_l given by

$$\begin{aligned} \prod_{V_l}:\ &\mathbb{R}^{mn}\to\mathbb{R}^{m_l} \\ &\boldsymbol{z}'\mapsto\boldsymbol{z}=(z_{ij})_{(i,j)\in I_l}. \end{aligned} \tag{3}$$

$I_l=\{(i,j);a_{ij}\neq 0, j\in\bigcup_{v\in V_l}\{\sum_{\vartheta=1}^{v-1}n_\vartheta+1,\dots,\sum_{\vartheta=1}^{v}n_\vartheta\}$ and m_l=$|I_l|$. Thus, $\boldsymbol{z}_l$ is the vector composed by the elements $\{z_{ij}\}$ of $\boldsymbol{z}'$, corresponding to the j^{th} component of $\boldsymbol{x}_v$ having a nonzero coefficient a_{ij} in the i^{th} constraint assigned to A_l. Let $\boldsymbol{M}'=[\mathrm{diag}(\boldsymbol{a}_1),\dots,\mathrm{diag}(\boldsymbol{a}_m)]^T$, where $\mathrm{diag}(\boldsymbol{a}_i)$ is the diagonal matrix constituted by $\boldsymbol{a}_i$, and $\boldsymbol{M}\boldsymbol{x}=((\prod_{V_l}\boldsymbol{M}'\boldsymbol{x})^T,\dots,(\prod_{V_l}\boldsymbol{M}'\boldsymbol{x})^T)^T$.

Problem (2) is equivalent to Problem (1). This is proved by letting $a_{ij}x_j=z_{ij}$ and summing upon j for every constraint i.

The dual of the minimization problem (2) is given by [11]

$$\underset{\boldsymbol{\lambda}\in\mathbb{R}^m}{\text{minimize}} \quad \left\{f^*(-\boldsymbol{M}^*\boldsymbol{\lambda})+g^*(\boldsymbol{\lambda})\right\}, \tag{4}$$

where $\boldsymbol{\lambda}=(\boldsymbol{\lambda}_1^T,\dots,\boldsymbol{\lambda}_l^T,\dots,\boldsymbol{\lambda}_L^T)^T$ with $\boldsymbol{\lambda}_l=\prod_{V_l}\boldsymbol{\lambda}'$ and $\boldsymbol{\lambda}'=(\lambda_{11},\dots,\lambda_{mn})^T\in\mathbb{R}^{mn}$ being the Lagrangian multipliers vector associated to the set of constraints $a_{ij}x_j=z_{ij}$. The functions f^* and g^* are the Fenchel's conjugate of f and g respectively, i.e., $f^*(\boldsymbol{u})=\sup_{\boldsymbol{x}}\{\langle\boldsymbol{x},\boldsymbol{u}\rangle-f(\boldsymbol{x})\}$, where $\langle\boldsymbol{x},\boldsymbol{u}\rangle=\boldsymbol{x}^T\boldsymbol{u}$ is the inner product of $\boldsymbol{x}$ and $\boldsymbol{u}$.

By strong duality, the optimization problem reduces to finding the minimum of the dual (4).

III. MONOTONE OPERATOR THEORY AND DISTRIBUTED OPTIMIZATION

Using monotone operator theory basics, we will prove that the dual problem (4) can be solved distributively by applying the proximal point algorithm (PPA) to the Douglas-Rachford (DR) splitting operator.

III-A. Monotone operator theory

Take an Euclidean space set $\mathcal{Y}$. We define its power set, denoted $\mathcal{P}(\mathcal{Y})=2^{\mathcal{Y}}$, as the family of all subsets of $\mathcal{Y}$ including the empty set $\emptyset$ and $\mathcal{Y}$ itself. An operator $D:\mathcal{X}\to\mathcal{Y}$ maps every point $\boldsymbol{x}\in\mathcal{X}$ to a point $D\boldsymbol{x}\in\mathcal{Y}$, while a set valued operator $D:\mathcal{X}\to 2^{\mathcal{Y}}$ maps every point $\boldsymbol{x}\in\mathcal{X}$ to a set $D\boldsymbol{x}\subset\mathcal{Y}$.

An operator D (single-valued or multi-valued), is characterized by its

- *graph:* $\mathrm{gra}\,(D)=\{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{X}\times\mathcal{Y}\mid\boldsymbol{y}\in D\boldsymbol{x}\}$
- *domain:* $\mathrm{dom}\,(D)=\{\boldsymbol{x}\in\mathcal{X}\mid\exists\,\boldsymbol{y}\in\mathcal{Y}:(\boldsymbol{x},\boldsymbol{y})\in D\}$
- *inverse:* $D^{-1}=\{(\boldsymbol{y},\boldsymbol{x})\in\mathcal{Y}\times\mathcal{X}\mid(\boldsymbol{x},\boldsymbol{y})\in gra\ D\}$
- *zero's set:* $\mathrm{Zer}\,(D)=D^{-1}0=\{\boldsymbol{x}\in\mathcal{X}\mid 0\in D\boldsymbol{x}\}$
- *set of fixed points:* $\mathrm{Fix}\,(D)=\{\boldsymbol{x}\in\mathcal{X}\mid x\in D\boldsymbol{x}\}$

III-B. Proximal point algorithm

Let $\boldsymbol{\lambda}$ be the minimum of the dual problem, and $D(\boldsymbol{\lambda})=-\boldsymbol{M}\,\partial f^*(-\boldsymbol{M}^*\boldsymbol{\lambda})+\partial g^*(\boldsymbol{\lambda})$ its subgradient mapping. By Fermat's rule [11, Th 16.2], $\boldsymbol{\lambda}$ is also the zero of D. D itself is a single valued maximal monotone operator [12]. Thus by [13, Th 3.6], and for any $\rho>0$, the resolvent $\mathcal{J}_{\rho D}=(I+\rho D)^{-1}$ of D is a single valued firmly nonexpansive operator with full domain. The following Lemma is then applicable.

Lemma 1 (Proximal Point Algorithm, PPA, [14]): Given a maximal monotone operator D, such that $\mathrm{Zer}\,(D)\neq\emptyset$. Then $\mathrm{Fix}\,(\mathcal{J}_{\rho D})$ is a singleton and $\mathrm{Zer}\,(D)=\mathrm{Fix}\,(\mathcal{J}_{\rho D})$. Moreover, starting from any initial point $\boldsymbol{\zeta}^0\in\mathrm{dom}\,(D)$, $\boldsymbol{\zeta}^k\to\mathrm{Fix}\,(\mathcal{J}_{\rho D})$, where $\boldsymbol{\zeta}^{k+1}=\mathcal{J}_{\rho D}(\boldsymbol{\zeta}^k)$, $k\ge 1$.

Hence instead of searching for $\mathrm{Zer}\,(D)$, we will search for $\mathrm{Fix}\,(\mathcal{J}_{\rho D})$. That is, starting from any initial point $\boldsymbol{\zeta}^0$, we will iterate $\boldsymbol{\zeta}^{k+1}=\mathcal{J}_{\rho D}(\boldsymbol{\zeta}^k)$ until convergence.

III-C. The distributed optimization

D can be written as $D=T+U$, where $T=-M\partial f^*\circ(-M^*)$ and $U=\partial g^*$ are two maximal monotone operators [12] . To apply the PPA, we need to compute $\mathcal{J}_{\rho(T+U)}$ which is not an easy task. From the structure of D, searching for $\mathrm{Zer}\,(D)$ is equivalent to finding $\mathrm{Zer}\,(T+U)$. This naturally calls for the DR splitting method in which the operators T and U are employed in separate steps. The DR method is indeed used to find the zero of the sum of two maximal monotone operators. The DR splitting operator, on which we will apply PPA, is given by

$$\begin{aligned} \mathcal{R}=\{&(\boldsymbol{\nu}+\rho\boldsymbol{z},\boldsymbol{\lambda}-\boldsymbol{\nu})\,;(\boldsymbol{\nu},\boldsymbol{\alpha})\in T,(\boldsymbol{\lambda},\boldsymbol{z})\in U \\ &\text{and } \boldsymbol{\nu}+\rho\boldsymbol{\alpha}=\boldsymbol{\lambda}-\rho\boldsymbol{z}\}. \end{aligned} \tag{5}$$

Since T and U are maximal monotone operators, thus $\mathcal{R}$ is also a maximal monotone operator [8], its resolvent $\mathcal{S}=(\mathcal{R}+\mathcal{I})^{-1}$ is firmly nonexpansive with full domain and is given by

$$\begin{aligned} \mathcal{S}&=\mathcal{J}_{\lambda T}\circ(2\mathcal{J}_{\boldsymbol{\lambda}U}-\mathcal{I})+(\mathcal{I}-\mathcal{J}_{\boldsymbol{\lambda}U}) \\ &=\{(\boldsymbol{\lambda}+\rho\boldsymbol{z},\boldsymbol{\nu}+\rho\boldsymbol{z})\,;\ (\boldsymbol{\nu},\boldsymbol{\alpha})\in T,(\boldsymbol{\lambda},\boldsymbol{z})\in U \\ &\qquad\text{and}\ \ \boldsymbol{\nu}+\rho\boldsymbol{\alpha}=\boldsymbol{\lambda}-\rho\boldsymbol{z}\}, \end{aligned} \tag{6}$$

$$\text{with } \mathrm{Fix}\,(\mathcal{S})=\{\boldsymbol{\lambda}+\rho\boldsymbol{z};(\boldsymbol{\lambda},\boldsymbol{z})\in U,(\boldsymbol{\lambda},-\boldsymbol{z})\in T\}. \tag{7}$$

Lemma 2: If $\bar{\boldsymbol{\zeta}}=\mathrm{Zer}\,(\mathcal{R})$, then $\bar{\boldsymbol{\lambda}}=\mathcal{J}_{\rho U}(\bar{\boldsymbol{\zeta}})=\mathrm{Zer}\,(T+U)$, where $\mathcal{J}_{\rho U}=\{(\boldsymbol{\lambda}+\rho\boldsymbol{z},\boldsymbol{\lambda})\,;\ (\boldsymbol{\lambda},\boldsymbol{z})\in U\}$ is the resolvent of U.

Proof: Let $\bar{\boldsymbol{\zeta}}=\mathrm{Zer}\,(\mathcal{R})$ then we also have $\bar{\boldsymbol{\zeta}}=\mathrm{Fix}\,(\mathcal{S})$. From (7), there is a unique $(\bar{\boldsymbol{\lambda}},\bar{\boldsymbol{z}})\in U$ verifying $\bar{\boldsymbol{\zeta}}=\bar{\boldsymbol{\lambda}}+\rho\,\bar{\boldsymbol{z}}$ and $T(\bar{\boldsymbol{\lambda}})=-\bar{\boldsymbol{z}}$. Then, $\mathcal{J}_{\rho U}(\bar{\boldsymbol{\zeta}})=\mathcal{J}_{\rho U}(\bar{\boldsymbol{\lambda}}+\rho\,\bar{\boldsymbol{z}})=\bar{\boldsymbol{\lambda}}$ and $(T+U)(\bar{\boldsymbol{\lambda}})=T(\bar{\boldsymbol{\lambda}})+U(\bar{\boldsymbol{\lambda}})=0$. Thus, $\bar{\boldsymbol{\lambda}}=\mathrm{Zer}\,(T+U)$. ∎

From Lemma 2, we conclude that searching for $\mathrm{Zer}\,(T+U)$ is equivalent to searching for $\mathrm{Zer}\,(\mathcal{R})$. Hence, we will apply PPA in order to find $\mathrm{Zer}\,(\mathcal{R})$, i.e., we will recursively search for $\bar{\boldsymbol{\zeta}}=\mathrm{Fix}\,(\mathcal{S})$.

Lemma 3: For any $\boldsymbol{\zeta}=\boldsymbol{\lambda}+\rho\boldsymbol{z}$, such that $(\boldsymbol{\lambda},\boldsymbol{z})\in U$ and $\boldsymbol{\lambda}=\mathcal{J}_{\rho U}(\boldsymbol{\zeta})$, there is a unique $\boldsymbol{x}$ such that the following is valid
(i) $\mathcal{S}(\boldsymbol{\zeta})=\mathcal{J}_{\mathcal{R}}(\boldsymbol{\zeta})=\boldsymbol{\lambda}+\rho\boldsymbol{M}\boldsymbol{x}$,
(ii) $\boldsymbol{x}=\underset{\boldsymbol{x}}{\mathrm{argmin}}\ \mathcal{L}_\rho(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\lambda})$,
where $\rho>0$ and $\mathcal{L}_\rho(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\lambda})$ is the augmented Lagrangian of the general problem (2) given by
$\mathcal{L}_\rho(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\lambda})\triangleq f(\boldsymbol{x})+g(\boldsymbol{z})+\langle\boldsymbol{\lambda},\boldsymbol{M}\boldsymbol{x}-\boldsymbol{z}\rangle+\frac{\rho}{2}\|\boldsymbol{M}\boldsymbol{x}-\boldsymbol{z}\|^2$.

Proof: (i) $\mathcal{R}(\boldsymbol{\zeta})=\boldsymbol{\lambda}-\boldsymbol{\nu}$ where $(\boldsymbol{\nu},\boldsymbol{\alpha})\in T$ and $\boldsymbol{\nu}+\rho\boldsymbol{\alpha}=\boldsymbol{\lambda}-\rho\boldsymbol{z}$. T is the maximal monotone operator given by $T=-M\partial f^*\circ(-M^*)$. Therefore, $\boldsymbol{\alpha}\in-\boldsymbol{M}\partial f^*(-\boldsymbol{M}^*\boldsymbol{\nu})$ and $\exists\,\boldsymbol{x}\in\partial f^*(-\boldsymbol{M}^*\boldsymbol{\nu})$ such that $\boldsymbol{\alpha}=-\boldsymbol{M}\boldsymbol{x}$. From (6) we have

$\mathcal{S}(\boldsymbol{\zeta}) = \boldsymbol{\nu} + \rho \boldsymbol{z} = \boldsymbol{\lambda} - \rho\boldsymbol{\alpha}$. Or $\boldsymbol{\alpha} = -\boldsymbol{M}\boldsymbol{x}$, we conclude that $\mathcal{S}(\boldsymbol{\zeta}) = \boldsymbol{\lambda} + \rho\boldsymbol{M}\boldsymbol{x}$.

(ii) Since f is a closed proper convex function, then by the Fenchel-Young inequality [11, Prop 16.9] the expression $\boldsymbol{x} \in \partial f^*(-\boldsymbol{M}^*\boldsymbol{\nu})$ is equivalent to $-\boldsymbol{M}^*\boldsymbol{\nu} \in \partial f(\boldsymbol{x})$, it follows that $0 \in \partial f(\boldsymbol{x}) + \boldsymbol{M}^*\boldsymbol{\nu}$. From the output of (5) we have $\boldsymbol{\nu} = \boldsymbol{\lambda} - \rho(\boldsymbol{z} + \boldsymbol{\alpha})$. We already proved that $\boldsymbol{\alpha} = -\boldsymbol{M}\boldsymbol{x}$, thus $\boldsymbol{\nu} = \boldsymbol{\lambda} + \rho(\boldsymbol{M}\boldsymbol{x} - \boldsymbol{z})$. It follows that $0 \in \partial f(\boldsymbol{x}) + \boldsymbol{M}^*\boldsymbol{\nu}$ translates to $0 \in \partial f(\boldsymbol{x}) + \boldsymbol{M}^*\boldsymbol{\lambda} + \rho\boldsymbol{M}^*(\boldsymbol{M}\boldsymbol{x} - \boldsymbol{z})$. We conclude that $\boldsymbol{x} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\ \mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\lambda})$. ■

Using Lemma 3 we can write explicitly the k^{th} recursion of the PPA applied to $\mathcal{R}$, $\boldsymbol{\zeta}^{k+1} = \mathcal{S}(\boldsymbol{\zeta}^k)$ as proved next.

Lemma 4: Let $\boldsymbol{\zeta}^0 = \boldsymbol{\lambda}^0 + \rho\boldsymbol{z}^0$ such that $\boldsymbol{\lambda}^0 = \mathcal{J}_{\rho U}(\boldsymbol{\zeta}^0)$ and $(\boldsymbol{z}^0, \boldsymbol{\lambda}^0) \in U$. Define $\forall\, k \geq 0$, $\boldsymbol{\zeta}^{k+1} = \mathcal{S}(\boldsymbol{\zeta}^k)$. Let $\boldsymbol{\lambda}^k = \mathcal{J}_{\rho U}(\boldsymbol{\zeta}^k)$, $(\boldsymbol{z}^k, \boldsymbol{\lambda}^k) \in U$ and $\boldsymbol{x}^{k+1}$ the unique $\boldsymbol{x}$ defined in Lemma 2 such that $\mathcal{S}(\boldsymbol{\zeta}^k) = \boldsymbol{\lambda}^k + \rho\boldsymbol{M}\boldsymbol{x}$. Then the following holds

$$\begin{aligned}
\boldsymbol{x}^{k+1} &= \underset{\boldsymbol{x}}{\operatorname{argmin}}\ \mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{z}^k; \boldsymbol{\lambda}^k),\\
\boldsymbol{z}^{k+1} &= \underset{\boldsymbol{z}}{\operatorname{argmin}}\ \mathcal{L}_\rho(\boldsymbol{x}^{k+1}, \boldsymbol{z}; \boldsymbol{\lambda}^k),\\
\boldsymbol{\lambda}^{k+1} &= \boldsymbol{\lambda}^k + \rho(\boldsymbol{M}\boldsymbol{x}^{k+1} - \boldsymbol{z}^{k+1}),
\end{aligned}$$

Proof: $\boldsymbol{\zeta}^k = \boldsymbol{\lambda}^k + \rho\boldsymbol{z}^k$, by Lemma 3 there is a unique $\boldsymbol{x}^{k+1}$ such that $\mathcal{S}(\boldsymbol{\zeta}^k) = \boldsymbol{\nu}^k + \rho\boldsymbol{z}^k = \boldsymbol{\lambda}^k + \rho\boldsymbol{M}\boldsymbol{x}^{k+1}$ and $\boldsymbol{x}^{k+1} = \operatorname{argmin}_{\boldsymbol{x}}\ \mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{z}^k; \boldsymbol{\lambda}^k)$.

To demonstrate the expression of $\boldsymbol{z}^{k+1}$ we use the hypothesis $\boldsymbol{\zeta}^{k+1} = \mathcal{S}(\boldsymbol{\zeta}^k)$. On the one hand, let $\boldsymbol{\zeta}^{k+1} = \boldsymbol{\lambda}^{k+1} + \rho\boldsymbol{z}^{k+1}$ where $(\boldsymbol{\lambda}^{k+1}, \boldsymbol{z}^{k+1}) \in U$ and $\boldsymbol{\lambda}^{k+1} = \mathcal{J}_{\rho U}(\boldsymbol{\zeta}^{k+1})$, then $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\zeta}^{k+1} - \rho\boldsymbol{z}^{k+1}$. On the other hand, $\mathcal{S}(\boldsymbol{\zeta}^k) = \boldsymbol{\lambda}^k + \rho\boldsymbol{M}\boldsymbol{x}^{k+1} = \boldsymbol{\zeta}^{k+1}$. Thus $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho\boldsymbol{M}\boldsymbol{x}^{k+1} - \rho\boldsymbol{z}^{k+1}$. Moreover, $\boldsymbol{z}^{k+1} \in U(\boldsymbol{\lambda}^{k+1})$, then by the Fenchel-Young inequality [11, Prop 16.9] this is equivalent to $\boldsymbol{\lambda}^{k+1} \in \partial g(\boldsymbol{z}^{k+1})$. Thus $0 \in \partial g(\boldsymbol{z}^{k+1}) - \boldsymbol{\lambda}^{k+1}$, it follows that $0 \in \partial g(\boldsymbol{z}^{k+1}) - \boldsymbol{\lambda}^k + \rho\boldsymbol{z}^{k+1} - \rho\boldsymbol{M}\boldsymbol{x}^{k+1}$, which is equivalent to $\boldsymbol{z}^{k+1} = \operatorname{argmin}_{\boldsymbol{z}}\ \mathcal{L}_\rho(\boldsymbol{x}^{k+1}, \boldsymbol{z}; \boldsymbol{\lambda}^k)$.

As for $\boldsymbol{\lambda}$ we have $\mathcal{S}(\boldsymbol{\zeta}^k) = \boldsymbol{\lambda}^k + \rho\boldsymbol{M}\boldsymbol{x}^{k+1}$. This expression is the same as $\boldsymbol{\zeta}^{k+1} = \boldsymbol{\lambda}^{k+1} + \rho\boldsymbol{z}^{k+1}$ because $\boldsymbol{\zeta}^{k+1} = \mathcal{S}(\boldsymbol{\zeta}^k)$. Using these two expressions of $\boldsymbol{\zeta}^{k+1}$, we obtain $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho(\boldsymbol{M}\boldsymbol{x}^{k+1} - \boldsymbol{z}^{k+1})$. ■

These three update steps of $\boldsymbol{x}^{k+1}$, $\boldsymbol{z}^{k+1}$ and $\boldsymbol{\lambda}^{k+1}$ can be distributed among the L subsystems as demonstrated next.

Lemma 5 : Define $\mathcal{S}_l(\boldsymbol{\zeta}) = (\boldsymbol{\lambda}_l + \prod_{V_l} \boldsymbol{M}'\boldsymbol{x})$, i.e, the l^{th} sub-block of $\mathcal{S}(\boldsymbol{\zeta}) = ((\boldsymbol{\lambda}_1 + \prod_{V_l} \boldsymbol{M}'\boldsymbol{x})^T, \ldots, (\boldsymbol{\lambda}_L + \prod_{V_l} \boldsymbol{M}'\boldsymbol{x})^T)^T$. For each area A_l we have

$$\begin{aligned}
\boldsymbol{x}_l^{k+1} = \underset{\boldsymbol{x}_l}{\operatorname{argmin}}\ & \sum_{v \in V_l} f_v(\boldsymbol{x}_v) + {\boldsymbol{\lambda}_l^k}^T \prod_{V_l}(\boldsymbol{M}'\boldsymbol{x}) \\
& + \frac{\rho}{2} \| \prod_{V_l}(\boldsymbol{M}'\boldsymbol{x}) - \boldsymbol{z}_l{}^k \|^2 \qquad (8a)\\
\boldsymbol{z}_l^{k+1} = \underset{\boldsymbol{z}_l}{\operatorname{argmin}}\ & -{\boldsymbol{\lambda}_l^k}^T \boldsymbol{z}_l + \frac{\rho}{2} \| \prod_{V_l}(\boldsymbol{M}'\boldsymbol{x}^{k+1}) - \boldsymbol{z}_l \|^2 \qquad (8b)\\
\boldsymbol{\lambda}_l^{k+1} = \boldsymbol{\lambda}_l^k + \rho(& \prod_{V_l}(\boldsymbol{M}'\boldsymbol{x}^{k+1}) - \boldsymbol{z}_l^{k+1}). \qquad (8c)
\end{aligned}$$

Proof: We will prove that the decomposition is true for $\boldsymbol{x}$. The same argumentation can be used for $\boldsymbol{z}$ and $\boldsymbol{\lambda}$.

$$\begin{aligned}
\boldsymbol{x}^{k+1} &= \underset{\boldsymbol{x}}{\operatorname{argmin}}\ f(\boldsymbol{x}) + g(\boldsymbol{z}^k) + \langle \boldsymbol{\lambda}^k, \boldsymbol{M}\boldsymbol{x} - \boldsymbol{z}^k \rangle + \tfrac{\rho}{2} \|\boldsymbol{M}\boldsymbol{x} - \boldsymbol{z}^k\|^2\\
&= \underset{\boldsymbol{x}}{\operatorname{argmin}}\ f(\boldsymbol{x}) + \langle \boldsymbol{\lambda}^k, \boldsymbol{M}\boldsymbol{x} \rangle + \tfrac{\rho}{2} \|\boldsymbol{M}\boldsymbol{x} - \boldsymbol{z}^k\|^2\\
&= \underset{\boldsymbol{x}}{\operatorname{argmin}}\ f(\boldsymbol{x}) + \sum_{l=1}^{L} \{ {\boldsymbol{\lambda}_l^k}^T \textstyle\prod_{V_l}(\boldsymbol{M}'\boldsymbol{x}) + \tfrac{\rho}{2} \| \textstyle\prod_{V_l}(\boldsymbol{M}'\boldsymbol{x}) - \boldsymbol{z}_l^k \|^2 \}.
\end{aligned}$$

This expression is separable into L independent parts $\boldsymbol{x}_l^{k+1}$ where $\boldsymbol{x}_l = \prod_{V_l}(\boldsymbol{x}) \in \mathbb{R}^{n_l}$ and $n_l = \sum_{v \in V_l} n_v$. $\boldsymbol{x}_l^{k+1}$ is assigned to A_l and contains only the components of $\boldsymbol{x}^{k+1}$ corresponding to the nodes $v \in V_l$. ■

As a conclusion, using PPA on the DR splitting operator $\mathcal{R}$, by Lemma 1 we can iteratively find the solution $\bar{\boldsymbol{x}}$ of (1). However, if every area A_l solves the subproblem given by (8a), (8b) and (8c) at every iteration k, we obtain the synchronous distributed algorithm, ADMM [8]. In the next section, we will prove that applying these update steps in a random fashion, where the subproblem of a randomly chosen area is solved at each iteration, converges to the solution of (1).

IV. ASYNCHRONOUS DISTRIBUTED OPTIMIZATION

The Gauss-Seidel method is a method of successive displacement, used to find an approximate solution of a linear system of equations starting from any initial point, and iterating till a stopping criterion is fulfilled.

As stated in the previous Section, when we iterate $\boldsymbol{\zeta}^{k+1} = \mathcal{S}(\boldsymbol{\zeta}^k)$, we obtain the well-known synchronous ADMM algorithm. In order to obtain an asynchronous algorithm, the updating process should be endowed with a random behavior. To this end, let $\boldsymbol{\zeta} = (\boldsymbol{\zeta}_1^T, \ldots, \boldsymbol{\zeta}_L^T)^T$ and suppose $\mathcal{S}(\boldsymbol{\zeta}) = (\mathcal{S}_1^T(\boldsymbol{\zeta}), \ldots, \mathcal{S}_L^T(\boldsymbol{\zeta}))^T$. We define for each area A_l, the operator $\hat{\mathcal{S}}_l$: $\mathcal{Y} \to \mathcal{Y}$ as

$$\hat{\mathcal{S}}_l(\boldsymbol{\zeta}) = (\boldsymbol{\zeta}_1^T, \ldots, \boldsymbol{\zeta}_{l-1}^T, \mathcal{S}_l^T(\boldsymbol{\zeta}), \boldsymbol{\zeta}_{l+1}^T, \ldots, \boldsymbol{\zeta}_L^T)^T, \qquad (9)$$

and we use the following theorem.

Theorem 1 [10, Th. 2] : Take a firmly nonexpansive operator $\mathcal{S} = (\mathcal{S}_1^T, \ldots, \mathcal{S}_L^T)^T$ with full domain on $\mathcal{Y}$ and a sequence of i.i.d. random variables $(\xi^k)_{k \in \mathbb{N}}$. Starting from any initial value $\boldsymbol{\zeta}^0$, the random iterates, $\boldsymbol{\zeta}^{k+1} = \hat{\mathcal{S}}_{\xi_{k+1}}(\boldsymbol{\zeta}^k)$ converges almost surely to a random variable supported by $\operatorname{Fix}(\mathcal{S})$ (when $\operatorname{Fix}(\mathcal{S}) \neq \emptyset$).

In our case, the expression of the resolvent $\mathcal{S}$ is simplified to $\mathcal{S}(\boldsymbol{\zeta}) = ((\prod_{V_l} \boldsymbol{\lambda}' + \boldsymbol{M}'\boldsymbol{x})^T, \ldots, (\prod_{V_l} \boldsymbol{\lambda}' + \boldsymbol{M}'\boldsymbol{x})^T)^T$ divided between the L areas , and we define $\hat{\mathcal{S}}_l$ as in (9). We choose a sequence of i.i.d. random variables (ξ^k), and we iterate $\boldsymbol{\zeta}^{k+1} = \hat{\mathcal{S}}_{\xi_{k+1}}(\boldsymbol{\zeta}^k)$ (i.e., if $\xi_{k+1} = l$, then only the nodes of A_l update their variables). Then these iterations converge almost surely to $\bar{\boldsymbol{\zeta}} = \operatorname{Fix}(\mathcal{S})$.

Thus, if we use the result of Theorem 1 along with the DR splitting method, the random iterates lead to $\bar{\boldsymbol{\lambda}} = \mathcal{J}_{\rho U}(\bar{\boldsymbol{\zeta}})$ which converges almost surely to $\operatorname{Zer}(D)$, i.e., the minimum of the dual problem (4). That is, by applying the random Gauss-Seidel iterations on the DR splitting method, we obtain an asynchronous iterative distributed process, named the asynchronous ADMM. This algorithm is given by solving (8a) to (8c) for a randomly chosen area A_l at each iteration k. This leads us to the solution $\bar{\boldsymbol{x}}$ of (1).

To summarize our asynchronous ADMM algorithm, at the k^{th} iteration, we pick a random variable ξ^{k+1} (as stated in Theorem 1), and consequently the area, A_l, whose $l = \xi^{k+1}$ is chosen to perform the update process by solving its subproblem given by (8a) to (8c). The result converges almost surely to the solution of (1).

V. IMPLEMENTATION AND SIMULATIONS

To illustrate our result, we consider the DC-OPF problem represented as (1), where we aim to control and optimize the operation of a power system so as to meet the underlying energy demand, with respect to a given objective function and subject to physical constraints on the available power and the grid infrastructure. We extract $\boldsymbol{A}$ and $\boldsymbol{b}$ from the linearized DC power flow equations $\boldsymbol{P_D} = \boldsymbol{P_G} - \boldsymbol{P}$ and $\boldsymbol{P} = \boldsymbol{B}\boldsymbol{\theta}$, where $\boldsymbol{P_D}, \boldsymbol{P_G}$ and $\boldsymbol{P}$ represent respectively the power demanded, generated and transmitted by the

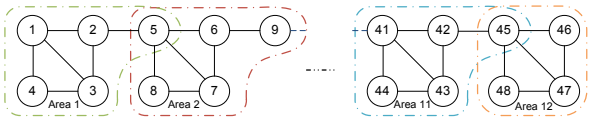


Fig. 1. Network of $N = 48$ nodes divided into $L = 12$ overlapping control areas.

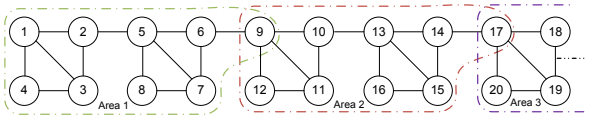


Fig. 2. Network of $N = 48$ nodes divided into $L = 6$ overlapping control areas.

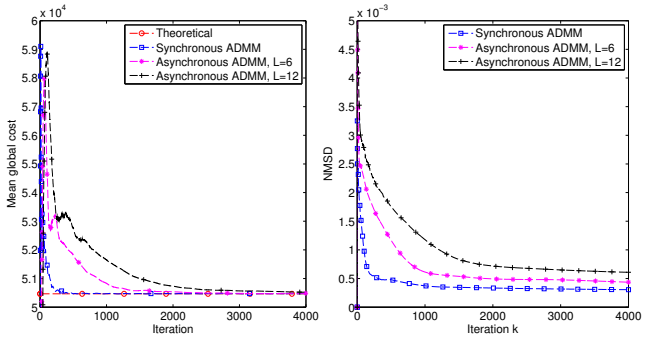
agents, $\boldsymbol{\theta}$ is the voltage phase vector and $\boldsymbol{B}$ is the imaginary part of the admittance matrix $\boldsymbol{Y}$, assumed to be purely imaginary. We divide the network into L overlapping connected areas and we solve the subproblem (8a), (8b) and (8c) for a given area A_l. Every node $v \in V_l$ has 3 primal variables indexed by $j \in \{3v-2, 3v-1, 3v\}$, the associated update expressions are given by

$$x_j^{k+1} = \min\{x_{j,min}, \max\{x_{j,max}, x_j^k - \frac{1}{\sum_{i=1} a_{ij}^2} \sum_i \frac{a_{ij} r_i(\boldsymbol{x}_l^k)}{d_l(i)} + \frac{1}{\rho \sum_{i=1} a_{ij}^2} \left.\frac{\partial f_v}{\partial x_j}\right|_{x_j = x_j^k} + \sum_i a_{ij} \lambda_{ij}^k)\}\}, \quad (10)$$

$$\lambda_{ij}^{k+1} = \lambda_{ij}^k + \frac{\rho}{d_l(i)} r_i(\boldsymbol{x}_l^{k+1}), \quad (11)$$

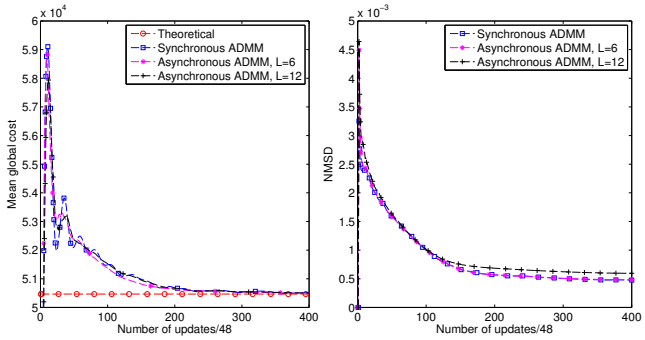
where each constraint $i \in \{i_p, \ldots, i_q\}$ is characterized by its degree $d(i)$ (i.e., the number of nonzero elements a_{ij}), by $d_l(i)$ its degree w.r.t to A_l (i.e., the number of nonzero elements a_{ij} that corresponds to $v \in V_l$) and by its residual w.r.t to A_l, $r_i(\boldsymbol{x}_l^k) = \sum_j a_{ij} x_j - \frac{d_l(i)}{d(i)} b_i$. The update step of z_{ij} reduces to $z_{ij}^{k+1} = a_{ij} x_{j,int}^{k+1} - \frac{r_i(x_l)^{k+1}}{d_l(i)}$ which is not involved in the above expressions and can be eliminated. It should be pointed out that we also have the same λ_{ij}, $\forall j \in \{1, \ldots, n\}$.

Let $V = \{1, \ldots, 24\}$ and $E = \{1, \ldots, 35\}$ be the set of nodes and edges respectively. The network is first divided into $L = 12$ then to $L = 6$ connected areas, as depicted in Fig. 1 and Fig. 2 respectively. Some nodes are shared, such as the nodes $\{5, 9, \ldots, 45\}$ in Fig. 1, and $\{9, 17, \ldots, 41\}$ in Fig. 2. We apply, using Matlab, the derived synchronous and asynchronous ADMM algorithms and we choose $\rho < 2$ because the convergence rate is degraded for larger values of ρ. We examine the convergence to the theoretical global cost and the global network state. If we compare upon the number of iterations performed, we obtain the mean global cost evolution (Fig. 3(a)) and the normalized mean squared deviation NMSD= $E\{\|\boldsymbol{x} - \boldsymbol{x}_{theoretical}\|^2 / \|\boldsymbol{x}_{theoretical}\|^2\}$ of the primal variables from the theoretical values (Fig. 3(b)). These plots prove the convergence of the derived algorithms, we also observe that having larger areas increases the convergence rate. In Fig. 4(a) and Fig. 4(b) we compare the evolution of the mean global cost and the NMSD with respect to the number of updates performed. In synchronous ADMM, at each iteration, all the nodes update and exchange their variables, while in the asynchronous algorithm, only the nodes of a randomly chosen area update their variables without communicating with the other areas (i.e., if we take a window of iterations, we may observe that one area updated more than one



(a) Mean global cost evolution. (b) Normalized mean squared deviation.

Fig. 3. Performance of ADMM algorithms w.r.t. the number of iterations performed, $N = 48, \rho = 1.8$.



(a) Mean global cost evolution. (b) Normalized mean squared deviation.

Fig. 4. Performance of ADMM algorithms w.r.t. the number of updates performed, $N = 48, \rho = 1.8$.

time its variables, while other areas were unactive). Thus, if we performed the same amount of updates as the synchronous version of ADMM, our algorithm would lead to the same result although theses updates are not equally divided between the areas.

VI. CONCLUSION

In this paper we presented an iterative decentralized solver of convex optimization problems in large networks. The derived algorithm divides the optimization problem into L subproblems, each is then solved independently of the other subproblems. We proved that it converges almost surely to the solution when the subproblems are solved synchronously (i.e., all the subproblems are solved at each iteration) or asynchronously (i.e., one subproblem is randomly chosen and solved at each iteration). We applied these distributed algorithm to the DC-OPF problem, and we showed through simulations the convergence to the optimal cost and the optimal network global state. Proving the convergence of this asynchronous distributed optimization algorithm in case of non-overlapping areas and studying the impact of noisy data exchange are interesting topics for future work.

VII. REFERENCES

[1] A. J. Wood and B. F. Wollenberg, *Power generation, operation, and control*. John Wiley & Sons, 2012.

[2] F. Zhuang and F. D. Galiana, "Towards a more rigorous and practical unit commitment by lagrangian relaxation," *Power Systems, IEEE Transactions on*, vol. 3, no. 2, pp. 763–773, 1988.

[3] B. H. Kim and R. Baldick, "Coarse-grained distributed optimal power flow," *Power Systems, IEEE Transactions on*, vol. 12, no. 2, pp. 932–939, 1997.

[4] A. Losi and M. Russo, "On the application of the auxiliary problem principle," *Journal of optimization theory and applications*, vol. 117, no. 2, pp. 377–396, 2003.

[5] A. J. Conejo, F. J. Nogales, and F. J. Prieto, "A decomposition procedure based on approximate newton directions," *Mathematical programming*, vol. 93, no. 3, pp. 495–515, 2002.

[6] D. G. Luenberger, *Linear and nonlinear programming*. Springer, 2003.

[7] D. Kalyanmoy, *Optimization for engineering design: Algorithms and examples*. PHI Learning Pvt. Ltd., 2004.

[8] J. Eckstein and D. P. Bertsekas, "On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators," *Mathematical Programming*, vol. 55, no. 1-3, pp. 293–318, 1992.

[9] 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.

[10] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, "Asynchronous Distributed Optimization using a Randomized Alternating Direction Method of Multipliers," *arXiv preprint arXiv:1303.2837*, 2013.

[11] H. H. Bauschke and P. L. Combettes, *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2011.

[12] R. Rockafellar, "On the maximal monotonicity of subdifferential mappings," *Pacific J. Math*, vol. 33, no. 1, pp. 209–216, 1970.

[13] J. Eckstein, *Splitting methods for monotone operators with applications to parallel optimization*. PhD thesis, Massachusetts Institute of Technology, 1989.

[14] R. T. Rockafellar, "Monotone operators and the proximal point algorithm," *SIAM Journal on Control and Optimization*, vol. 14, no. 5, pp. 877–898, 1976.