

Asynchronous alternating direction method of multipliers applied to the direct-current optimal power flow problem

Azary Abboud, Romain Couillet, Merouane Debbah, Houria Siguerdidjane

► **To cite this version:**

Azary Abboud, Romain Couillet, Merouane Debbah, Houria Siguerdidjane. Asynchronous alternating direction method of multipliers applied to the direct-current optimal power flow problem. 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) , May 2014, Florence, Italy. 10.1109/ICASSP.2014.6855111 . hal-01094819

HAL Id: hal-01094819

<https://hal.archives-ouvertes.fr/hal-01094819>

Submitted on 28 Jan 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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

Azary Abboud*[†] Romain Couillet* Mérouane Debbah* Houria Siguerdjane[†]

* Alcatel-Lucent Chair on Flexible Radio - SUPÉLEC, Gif-sur-Yvette, France

[†]Automatic Control Department - SUPÉLEC, Gif-sur-Yvette, France.

ABSTRACT

In a large network of agents, we consider a distributed convex optimization problem where each agent has a private convex cost function and 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 results 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

$$\underset{\{\mathbf{x}_v, v \in \{1, \dots, N\}\}}{\text{minimize}} \quad \sum_{v=1}^N f_v(\mathbf{x}_v)$$

subject to

$$\begin{aligned} \mathbf{a}_i^T (\mathbf{x}_1^T, \dots, \mathbf{x}_N^T)^T &= b_i \quad i = 1, \dots, m, \\ \mathbf{x}_{v, \min} &\leq \mathbf{x}_v \leq \mathbf{x}_{v, \max} \quad v = 1, \dots, N. \end{aligned} \quad (1)$$

where each node $v \in \{1, \dots, N\}$ controls a vector of local variables $\mathbf{x}_v = (x_{(\sum_{\vartheta=1}^{v-1} n_{\vartheta}+1)}, \dots, x_{(\sum_{\vartheta=1}^v n_{\vartheta})})^T \in \mathbb{R}^{n_v}$, bounded by $\mathbf{x}_{v, \min} \in \mathbb{R}^{n_v}$ and $\mathbf{x}_{v, \max} \in \mathbb{R}^{n_v}$. The local operating cost of a node v is given by the closed proper convex function $f_v : \mathbb{R}^{n_v} \mapsto \mathbb{R}$. Note that f_v is proper if $\exists x_v \in \mathbb{R}^{n_v}$ such that $f_v(x_v) < +\infty$, and $\forall x_v \in \mathbb{R}^{n_v}$, $f_v(x_v) > -\infty$. Each equality constraint i , $i \in \{1, \dots, m\}$, is characterized by the vector of coefficients $\mathbf{a}_i = (a_{i1}, \dots, a_{ij}, \dots, a_{in})^T \in \mathbb{R}^n$, 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 distributively, 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 augmented Lagrangian technique [2], the auxiliary problem principle [3] and the approximate Newton directions [4], [5]. These 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 [6]. The alternating direction method of multipliers (ADMM) [7], [8], 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 [8].

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 values of the variables coupling them. For these reasons, relying on the recent results of [9], we derive an asynchronous distributed algorithm with guaranteed convergence. The work presented in [9] aims at finding the global state of the network by solving an unconstrained optimization problem. The agents share the same state which represents the global network state and they try to find a consensus on the value of this individual variable. 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, we derive in Section IV 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 , $l \in \{1, \dots, L\}$. For each area A_l , we assign a subset of vertices $V_l \subset V$ and a subset of edges $E_l = \{\{u, v\}; (u, v) \in V_l^2\} \cap E$ such that $\bigcup_{l=1}^L V_l = V$ and $\bigcup_{l=1}^L G(V_l)$ is connected, where $G(V_l)$ is the sub-graph (V_l, E_l) . Let $\mathbf{x} = (\mathbf{x}_1^T, \dots, \mathbf{x}_v^T, \dots, \mathbf{x}_N^T)^T \in \mathbb{R}^n$, $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_m]^T \in \mathbb{R}^{mn}$,

$\mathbf{b} = (b_1, \dots, b_m)^T \in \mathbb{R}^m$, $\mathbf{x}_{min} = (\mathbf{x}_{1,min}^T, \dots, \mathbf{x}_{N,min}^T)^T \in \mathbb{R}^n$ and $\mathbf{x}_{max} = (\mathbf{x}_{1,max}^T, \dots, \mathbf{x}_{N,max}^T)^T \in \mathbb{R}^n$. We convert problem (1) into the following canonical form

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

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

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

$$g(\mathbf{z}) = \begin{cases} 0 & \text{if } \sum_{j=1}^n z_{ij} = b_i; a_{ij} = 0 \Rightarrow z_{ij} = 0, \forall i = 1, \dots, m \\ +\infty & \text{otherwise,} \end{cases}$$

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

$$\prod_{V_l} : \mathbb{R}^{mn} \rightarrow \mathbb{R}^{m_l}$$

$$\mathbf{z}' \mapsto \mathbf{z} = (z_{ij})_{(i,j) \in I_l}. \quad (3)$$

$I_l = \{(i, j); a_{ij} \neq 0, j \in \bigcup_{v \in V_l} \{\sum_{\vartheta=1}^{v-1} n_{\vartheta} + 1, \dots, \sum_{\vartheta=1}^v n_{\vartheta}\}\}$ and $m_l = |I_l|$. Thus, \mathbf{z}_l is the vector composed by the elements $\{z_{ij}\}$ of \mathbf{z}' corresponding to the j^{th} component of \mathbf{x}_v having a nonzero coefficient a_{ij} in the i^{th} constraint assigned to A_l . $\mathbf{M}' = [\text{diag}(\mathbf{a}_1), \dots, \text{diag}(\mathbf{a}_m)]^T$, where $\text{diag}(\mathbf{a}_i)$ is the diagonal matrix constituted by \mathbf{a}_i , and $\mathbf{M}\mathbf{x} = ((\prod_{V_1} \mathbf{M}'\mathbf{x})^T, \dots, (\prod_{V_l} \mathbf{M}'\mathbf{x})^T)^T$.

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

The dual of the minimization problem (2) is given by [10]

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

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., $f^*(\boldsymbol{\lambda}) = \sup_{\mathbf{x}} \{\langle \mathbf{x}, \boldsymbol{\lambda} \rangle - f(\mathbf{x})\}$, where $\langle \mathbf{x}, \boldsymbol{\lambda} \rangle = \mathbf{x}^T \boldsymbol{\lambda}$ is the inner product of \mathbf{x} and $\boldsymbol{\lambda}$.

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 prove that the dual problem (4) can be solved distributively by applying the proximal point algorithm (PPA) on 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 $D : \mathcal{X} \rightarrow \mathcal{Y}$ maps every point $\mathbf{x} \in \mathcal{X}$ to a point $D\mathbf{x} \in \mathcal{Y}$, while a set valued operator $D : \mathcal{X} \rightarrow 2^{\mathcal{Y}}$ maps every point $\mathbf{x} \in \mathcal{X}$ to a set $D\mathbf{x} \subset \mathcal{Y}$.

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

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

III-B. Proximal point algorithm

Let $\boldsymbol{\lambda}$ be the minimum of the dual problem, and $D(\boldsymbol{\lambda}) = -M\partial f^*(-M^*\boldsymbol{\lambda}) + \partial g^*(\boldsymbol{\lambda})$ its subgradient mapping. By Fermat's rule [10, Th 16.2], $\boldsymbol{\lambda}$ is also the zero of D . D itself is a single valued maximal monotone operator [11]. Thus, by [12, 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 (PPA, [13]): Given a maximal monotone operator D , such that $\text{Zer}(D) \neq \emptyset$. Then $\text{Fix}(\mathcal{J}_{\rho D})$ is a singleton and $\text{Zer}(D) = \text{Fix}(\mathcal{J}_{\rho D})$. Moreover, starting from any initial point $\zeta^0 \in \text{dom}(D)$, $\zeta^k \rightarrow \text{Fix}(\mathcal{J}_{\rho D})$, where $\zeta^{k+1} = \mathcal{J}_{\rho D}(\zeta^k)$, $k \geq 1$.

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

III-C. The distributed optimization

D can be written as $D = T + U$, where $T = -M\partial f^*(-M^*)$ and $U = \partial g^*$ are two maximal monotone operators [11]. 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 $\text{Zer}(D)$ is equivalent to searching for $\text{Zer}(T+U)$. 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 apply the PPA, is given by

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

Since T and U are maximal monotone operators, \mathcal{R} is also a maximal monotone operator [7]. The resolvent $\mathcal{S} = (\mathcal{R} + \mathcal{I})^{-1}$ of \mathcal{R} is firmly nonexpansive with full domain, it is given by

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

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

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

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

From Lemma 2, we conclude that trying to find $\text{Zer}(\mathcal{R})$ is equivalent to finding $\text{Zer}(T + U)$. Hence, we apply PPA on \mathcal{R} , i.e., we recursively search for $\bar{\zeta} = \text{Fix}(\mathcal{S})$.

Lemma 3: For any $\zeta = \boldsymbol{\lambda} + \rho\mathbf{z}$, such that $(\boldsymbol{\lambda}, \mathbf{z}) \in U$ and $\boldsymbol{\lambda} = \mathcal{J}_{\rho U}(\zeta)$, there is a unique \mathbf{x} such that the following is valid

$$(i) \mathcal{S}(\zeta) = \mathcal{J}_{\mathcal{R}}(\zeta) = \boldsymbol{\lambda} + \rho\mathbf{M}\mathbf{x},$$

$$(ii) \mathbf{x} = \underset{\mathbf{x}}{\text{argmin}} \mathcal{L}_{\rho}(\mathbf{x}, \mathbf{z}; \boldsymbol{\lambda}),$$

where $\rho > 0$ and $\mathcal{L}_{\rho}(\mathbf{x}, \mathbf{z}; \boldsymbol{\lambda})$ is the augmented Lagrangian of the general problem (2) given by

$$\mathcal{L}_{\rho}(\mathbf{x}, \mathbf{z}; \boldsymbol{\lambda}) \triangleq f(\mathbf{x}) + g(\mathbf{z}) + \langle \boldsymbol{\lambda}, \mathbf{M}\mathbf{x} - \mathbf{z} \rangle + \frac{\rho}{2} \|\mathbf{M}\mathbf{x} - \mathbf{z}\|^2.$$

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

(ii) Since f is a closed proper convex function, then by the Fenchel-Young inequality [10, Prop 16.9] the expression $\mathbf{x} \in \partial f^*(-\mathbf{M}^*\boldsymbol{\nu})$ is equivalent to $-\mathbf{M}^*\boldsymbol{\nu} \in \partial f(\mathbf{x})$, it follows that $0 \in \partial f(\mathbf{x}) + \mathbf{M}^*\boldsymbol{\nu}$. From the output of (5), we have $\boldsymbol{\nu} = \boldsymbol{\lambda} - \rho(\mathbf{z} + \boldsymbol{\alpha})$. We also have $\boldsymbol{\alpha} = -\mathbf{M}\mathbf{x}$. Then, $\boldsymbol{\nu} = \boldsymbol{\lambda} + \rho(\mathbf{M}\mathbf{x} - \mathbf{z})$. It follows that $0 \in \partial f(\mathbf{x}) + \mathbf{M}^*\boldsymbol{\nu}$ which translates to $0 \in \partial f(\mathbf{x}) + \mathbf{M}^*\boldsymbol{\lambda} + \rho\mathbf{M}^*(\mathbf{M}\mathbf{x} - \mathbf{z})$. We conclude that $\mathbf{x} = \operatorname{argmin}_{\mathbf{x}} \mathcal{L}_\rho(\mathbf{x}, \mathbf{z}; \boldsymbol{\lambda})$. ■

Next, we write explicitly the k^{th} recursion of the PPA applied to \mathcal{R} , i.e., the recursion $\zeta^{k+1} = \mathcal{S}(\zeta^k)$.

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

$$\mathbf{x}^{k+1} = \operatorname{argmin}_{\mathbf{x}} \mathcal{L}_\rho(\mathbf{x}, \mathbf{z}^k; \boldsymbol{\lambda}^k),$$

$$\mathbf{z}^{k+1} = \operatorname{argmin}_{\mathbf{z}} \mathcal{L}_\rho(\mathbf{x}^{k+1}, \mathbf{z}; \boldsymbol{\lambda}^k),$$

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^{k+1}),$$

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

To demonstrate the expressions of $\boldsymbol{\lambda}^{k+1}$ and \mathbf{z}^{k+1} we use the hypothesis $\zeta^{k+1} = \mathcal{S}(\zeta^k)$. On the one hand, let $\zeta^{k+1} = \boldsymbol{\lambda}^{k+1} + \rho\mathbf{z}^{k+1}$ where $(\boldsymbol{\lambda}^{k+1}, \mathbf{z}^{k+1}) \in U$ and $\boldsymbol{\lambda}^{k+1} = \mathcal{J}_{\rho U}(\zeta^{k+1})$, then $\boldsymbol{\lambda}^{k+1} = \zeta^{k+1} - \rho\mathbf{z}^{k+1}$. On the other hand, $\mathcal{S}(\zeta^k) = \boldsymbol{\lambda}^k + \rho\mathbf{M}\mathbf{x}^{k+1} = \zeta^{k+1}$. Thus $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho\mathbf{M}\mathbf{x}^{k+1} - \rho\mathbf{z}^{k+1}$. Moreover, $\mathbf{z}^{k+1} \in U(\boldsymbol{\lambda}^{k+1})$, then by the Fenchel-Young inequality [10, Prop 16.9] this is equivalent to $\boldsymbol{\lambda}^{k+1} \in \partial g(\mathbf{z}^{k+1})$. Thus $0 \in \partial g(\mathbf{z}^{k+1}) - \boldsymbol{\lambda}^{k+1}$, it follows that $0 \in \partial g(\mathbf{z}^{k+1}) - \boldsymbol{\lambda}^k + \rho\mathbf{z}^{k+1} - \rho\mathbf{M}\mathbf{x}^{k+1}$, which is equivalent to $\mathbf{z}^{k+1} = \operatorname{argmin}_{\mathbf{z}} \mathcal{L}_\rho(\mathbf{x}^{k+1}, \mathbf{z}; \boldsymbol{\lambda}^k)$. ■

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

Lemma 5: Define $\mathcal{S}_l(\zeta) = (\boldsymbol{\lambda}_l + \prod_{V_l} \mathbf{M}'\mathbf{x})$, i.e., the l^{th} sub-block of $\mathcal{S}(\zeta) = ((\boldsymbol{\lambda}_1 + \prod_{V_1} \mathbf{M}'\mathbf{x})^T, \dots, (\boldsymbol{\lambda}_L + \prod_{V_L} \mathbf{M}'\mathbf{x})^T)^T$. For each area A_l we have

$$\mathbf{x}_l^{k+1} = \operatorname{argmin}_{\mathbf{x}_l} \sum_{v \in V_l} f_v(\mathbf{x}_v) + \boldsymbol{\lambda}_l^{kT} \prod_{V_l} (\mathbf{M}'\mathbf{x}) + \frac{\rho}{2} \left\| \prod_{V_l} (\mathbf{M}'\mathbf{x}) - \mathbf{z}_l^k \right\|^2 \quad (8a)$$

$$\mathbf{z}_l^{k+1} = \operatorname{argmin}_{\mathbf{z}_l} -\boldsymbol{\lambda}_l^{kT} \mathbf{z}_l + \frac{\rho}{2} \left\| \prod_{V_l} (\mathbf{M}'\mathbf{x}^{k+1}) - \mathbf{z}_l \right\|^2 \quad (8b)$$

$$\boldsymbol{\lambda}_l^{k+1} = \boldsymbol{\lambda}_l^k + \rho \left(\prod_{V_l} (\mathbf{M}'\mathbf{x}^{k+1}) - \mathbf{z}_l^{k+1} \right). \quad (8c)$$

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

$$\begin{aligned} \mathbf{x}^{k+1} &= \operatorname{argmin}_{\mathbf{x}} f(\mathbf{x}) + g(\mathbf{z}^k) + \langle \boldsymbol{\lambda}^k, \mathbf{M}\mathbf{x} - \mathbf{z}^k \rangle + \frac{\rho}{2} \|\mathbf{M}\mathbf{x} - \mathbf{z}^k\|^2 \\ &= \operatorname{argmin}_{\mathbf{x}} f(\mathbf{x}) + \langle \boldsymbol{\lambda}^k, \mathbf{M}\mathbf{x} \rangle + \frac{\rho}{2} \|\mathbf{M}\mathbf{x} - \mathbf{z}^k\|^2 \\ &= \operatorname{argmin}_{\mathbf{x}} f(\mathbf{x}) + \sum_{l=1}^L \left\{ \boldsymbol{\lambda}_l^{kT} \prod_{V_l} (\mathbf{M}'\mathbf{x}) + \frac{\rho}{2} \left\| \prod_{V_l} (\mathbf{M}'\mathbf{x}) - \mathbf{z}_l^k \right\|^2 \right\}. \end{aligned}$$

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

As a conclusion, when we use the PPA on the DR splitting operator \mathcal{R} , by Lemma 1 we can iteratively find the solution $\bar{\mathbf{x}}$ 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 [7]. In the next section, we 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 $\zeta^{k+1} = \mathcal{S}(\zeta^k)$, 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 $\zeta = (\zeta_1^T, \dots, \zeta_L^T)^T$ and suppose $\mathcal{S}(\zeta) = (\mathcal{S}_1^T(\zeta), \dots, \mathcal{S}_L^T(\zeta))^T$. We define for each area A_l , the operator $\hat{\mathcal{S}}_l: \mathcal{Y} \rightarrow \mathcal{Y}$ as

$$\hat{\mathcal{S}}_l(\zeta) = (\zeta_1^T, \dots, \zeta_{l-1}^T, \mathcal{S}_l^T(\zeta), \zeta_{l+1}^T, \dots, \zeta_L^T)^T, \quad (9)$$

and we use the following theorem.

Theorem 1 [9, Th. 2] : Take a firmly nonexpansive operator $\mathcal{S} = (\mathcal{S}_1^T, \dots, \mathcal{S}_L^T)^T$ 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 ζ^0 , the random iterates, $\zeta^{k+1} = \hat{\mathcal{S}}_{\xi_{k+1}}(\zeta^k)$ 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 $\mathcal{S}(\zeta) = ((\prod_{V_l} \boldsymbol{\lambda}' + \mathbf{M}'\mathbf{x})^T, \dots, (\prod_{V_l} \boldsymbol{\lambda}' + \mathbf{M}'\mathbf{x})^T)^T$ 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 $\zeta^{k+1} = \hat{\mathcal{S}}_{\xi_{k+1}}(\zeta^k)$ (i.e., if $\xi_{k+1} = l$, then only the nodes of A_l update their variables). Then these iterations converge almost surely to $\bar{\zeta} = \operatorname{Fix}(\mathcal{S})$.

Thus, if we use the result of Theorem 1 along with the DR splitting method, the random iterates lead to $\bar{\boldsymbol{\lambda}} = \mathcal{J}_{\rho U}(\bar{\zeta})$ 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 $\bar{\mathbf{x}}$ 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) while the other areas retain their older values of the primal and dual variables. 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 \mathbf{A} and \mathbf{b} from the linearized DC power flow equations $\mathbf{P}_D = \mathbf{P}_G - \mathbf{P}$ and $\mathbf{P} = \mathbf{B}\boldsymbol{\theta}$, where \mathbf{P}_D , \mathbf{P}_G and \mathbf{P} represent respectively the power demanded, generated and transmitted by the agents, $\boldsymbol{\theta}$ is the voltage phase vector and \mathbf{B} is the imaginary part of the admittance matrix \mathbf{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 . Each node

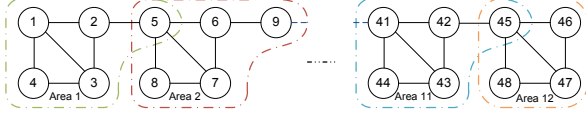


Fig. 1. Network of $N = 48$ nodes divided into $L = 12$ overlapping control areas.

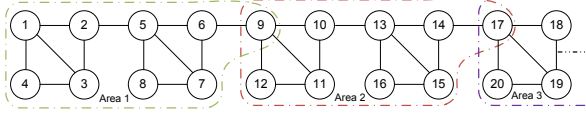


Fig. 2. Network of $N = 48$ nodes divided into $L = 6$ overlapping control areas.

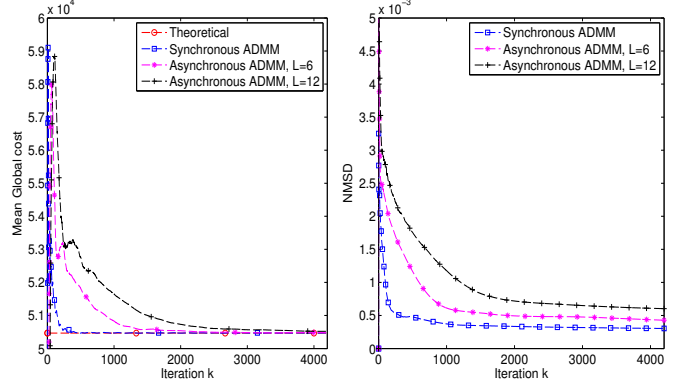
$v \in V_l$ has 3 primal variables indexed by $j \in \{3v - 2, 3v - 1, 3v\}$, the associated update expressions are given by

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

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

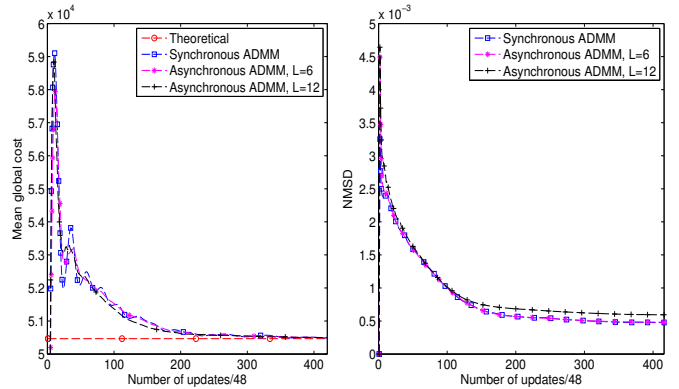
where each constraint $i \in \{i_p, \dots, i_q\}$ is characterized by its degree $d(i)$ (i.e., the number of nonzero elements $\{a_{ij}\}$), by $d_l(i)$ its degree with respect to A_l (i.e., the number of nonzero elements $\{a_{ij}\}$ that corresponds to $v \in V_l$) and by its residual with respect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

Let $V = \{1, \dots, 24\}$ and $E = \{1, \dots, 35\}$ 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 $\{5, 9, \dots, 45\}$ in Fig. 1, and $\{9, 17, \dots, 41\}$ 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 $NMSD = \mathbb{E}\{\|\mathbf{x} - \mathbf{x}^{theoretical}\|^2 / \|\mathbf{x}^{theoretical}\|^2\}$ 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 time its variables, while other areas were inactive). Thus, if we performed the same amount of updates as the synchronous version



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

Fig. 3. Performance of the proposed ADMM algorithms with respect to the number of iterations performed, $N = 48$, $\rho = 1.8$.



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

Fig. 4. Performance of the proposed ADMM algorithms with respect to the number of updates performed, $N = 48$, $\rho = 1.8$.

of ADMM, our algorithm would lead to the same result although these updates are not equally divided between the areas.

VI. CONCLUSION

In this paper, we have 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 have proved that the algorithm 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 have applied these distributed algorithms to the DC-OPF problem. We have 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] 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.
- [3] 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.
- [4] 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.
- [5] D. G. Luenberger, *Linear and nonlinear programming*. Springer, 2003.
- [6] D. Kalyanmoy, *Optimization for engineering design: Algorithms and examples*. PHI Learning Pvt. Ltd., 2004.
- [7] 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.
- [8] 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.
- [9] 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.
- [10] H. H. Bauschke and P. L. Combettes, *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2011.
- [11] R. Rockafellar, "On the maximal monotonicity of subdifferential mappings," *Pacific J. Math*, vol. 33, no. 1, pp. 209–216, 1970.
- [12] J. Eckstein, *Splitting methods for monotone operators with applications to parallel optimization*. PhD thesis, Massachusetts Institute of Technology, 1989.
- [13] R. T. Rockafellar, "Monotone operators and the proximal point algorithm," *SIAM Journal on Control and Optimization*, vol. 14, no. 5, pp. 877–898, 1976.