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.science/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 Siguerdidjane†

* 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

$$
\begin{align*}
\text{minimize} & \quad \sum_{v=1}^{N} f_v(x_v) \\
\text{subject to} & \quad \alpha_i^T(x_1^T, \ldots, x_N^T) = b_i, \quad i = 1, \ldots, m, \\
& \quad x_v, \min \leq x_v \leq x_v, \max, \quad v = 1, \ldots, N,
\end{align*}
$$

where each node $v \in \{1, \ldots, N\}$ controls a vector of local variables $x_v = (x_v^1, x_v^2, \ldots, x_v^n)^T \in \mathbb{R}^n_v$, bounded by $x_v, \min \in \mathbb{R}^n_v$ and $x_v, \max \in \mathbb{R}^n_v$. The local operating cost of a node $v$ is given by the closed proper convex function $f_v : \mathbb{R}^n_v \rightarrow \mathbb{R}$. 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 $i$, $i = 1, \ldots, m$, is characterized by the vector of coefficients $\alpha_i = (\alpha_{i1}, \ldots, \alpha_{im})^T \in \mathbb{R}^m$, 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 $A_l$, $l \in \{1, \ldots, L\}$. For each area $A_l$, we assign a subset of vertices $V_l \subset V$ and a subset of edges $E_l = \{(u, v) \in V^2 \} \cap E$ 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 $x = (x_1^1, x_1^2, \ldots, x_L^N)^T \in \mathbb{R}^n$, $A = [a_1, \ldots, a_m]^T \in \mathbb{R}^mn$, etc.
\[ b = (b_1, \ldots, b_m)^T \in \mathbb{R}^m, \quad x_{\text{min}} = (x_{1,\text{min}}, \ldots, x_{n,\text{min}})^T \in \mathbb{R}^n \] and
\[ x_{\text{max}} = (x_{1,\text{max}}, \ldots, x_{N,\text{max}})^T \in \mathbb{R}^n, \]
We convert problem (1) into the following canonical form
\[
\begin{align*}
\text{minimize}_{x \in \mathbb{R}^m} & \quad f(x) + g(z) \\
\text{subject to} & \quad Mx = z,
\end{align*}
\]
where \( f \) and \( g \) are two closed proper convex functions given by
\[
\begin{align*}
f(x) &= \begin{cases} 
\sum_{x \in V} f_x(x) & \text{if } x_{\text{min}} \leq x \leq x_{\text{max}}, \forall x \in V \\
+\infty & \text{otherwise,}
\end{cases} \\
g(z) &= \begin{cases} 
0 & \text{if } \sum_{j=1}^m z_{ij} = b_i, a_{ij} = 0 \Rightarrow z_{ij} = 0, \quad \forall i = 1, \ldots, m \\
+\infty & \text{otherwise,}
\end{cases}
\end{align*}
\]
where \( z = (z_1^T, \ldots, z_m^T) \), and \( z_i = \prod_{V_i} z' \) is the projection of \( z' = (z_{i1}, z_{i1}, \ldots, z_{im})^T \) on \( V_i \) given by
\[
\prod_{V_i} z_i = (\prod_{V_i} z_{i1}^T, \ldots, \prod_{V_i} z_{im}^T)^T.
\]
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 \( i \).

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

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

- **graph:** graf \( D \) = \{ (x, y) \in \mathcal{X} \times \mathcal{Y} | y = Dx \};
- **domain:** dom \( D \) = \{ x \in \mathcal{X} | \exists y \in \mathcal{Y} : (x, y) \in D \};
- **inverse:** \( D^{-1} \) = \{ (y, x) \in \mathcal{Y} \times \mathcal{X} | (x, y) \in D \};
- **zero's set:** zer \( D \) = \{ x \in \mathcal{X} | D^{-1} = \emptyset \};
- **set of fixed points:** Fix \( D \) = \{ x \in \mathcal{X} | x \in D \mathcal{X} \}.

#### III-B. Proximal point algorithm

Let \( \lambda \) be the minimum of the dual problem, and \( D(\lambda) = -M \partial f^*(-M^{*} \lambda) + g^* \lambda \) its subgradient mapping. By Fermat’s rule [10, Th 16.2], \( \lambda \) 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 \( J_{\rho D}(x) = (I + \rho D)^{-1} \) 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 \( \text{Fix}(J_{\rho D}) \) is a singleton and \( \text{zer}(D) = \text{Fix}(J_{\rho D}) \). Moreover, starting from any initial point \( \xi^0 \in \text{dom}(D) \), \( \xi^0 \rightarrow \text{Fix}(J_{\rho D}) \), where \( \xi^{k+1} = J_{\rho D}(\xi^k) \). \( k \geq 1 \).

Hence, instead of searching for \( \text{zer}(D) \), we search for \( \text{Fix}(J_{\rho D}) \). That is, starting from any initial point \( \xi^0 \), we update \( \xi^{k+1} = J_{\rho D}(\xi^k) \) until convergence.

#### III-C. The distributed optimization

Let \( \mathcal{D} \) be written as \( \mathcal{D} = T + U \), where \( T = -M \partial f^* \circ (-M^{*}) \) and \( U = g^* \) are two maximal monotone operators [11]. To apply the PPA, we need to compute \( J_{\rho (T + U)} \) which is not an easy task. From the structure of \( \mathcal{D} \), 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
\[
\mathcal{R} = \left\{ (\nu + \rho z, \lambda - \nu) ; (\nu, \alpha) \in T, (\lambda, z) \in U \right\}
\]

(5)

Since \( T \) and \( U \) are maximal monotone operators, \( \mathcal{R} \) is also a maximal monotone operator [7]. The resolvent \( \mathcal{S} = (\mathcal{R} + I)^{-1} \) of \( \mathcal{R} \) is firmly nonexpansive with full domain, it is given by
\[
\mathcal{S} = J_{\mathcal{X} \mathcal{T}} \circ (2J_{\mathcal{X} \mathcal{U}} - I) + (I - J_{\mathcal{X} \mathcal{U}})
\]

(6)

with \( \text{Fix}(S) = \{ (\lambda + \rho z, \lambda, z) \in U, (\lambda, z) \in T \} \).

**Lemma 2:** If \( \mathcal{Z} = \text{zer}(\mathcal{R}) \), then \( \mathcal{Z} = J_{\mathcal{X} \mathcal{T}}(\mathcal{Z} + T + U) \), where \( \mathcal{J}_U = \{ (\lambda + \rho z, \lambda) ; (\lambda, z) \in U \} \) is the resolvent of \( \mathcal{U} \).

**Proof:** Let \( \mathcal{Z} = \text{zer}(\mathcal{R}) \). Since \( \mathcal{Z} \) is unique \( (\lambda, z) \in U \) verifying \( \mathcal{Z} = \lambda + \rho z \) and \( T(\lambda) = \emptyset \) then,\( \mathcal{J}_U(\mathcal{Z}) = \lambda + \rho z = \lambda \) and \( (T + U)(\lambda) = T(\lambda) + U(\lambda) = 0 \). Thus, \( \lambda = \text{zer}(T + U) \).

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 \( \mathcal{Z} = \text{Fix}(\mathcal{S}) \\

**Lemma 3:** For any \( \lambda = \lambda + \rho z \), such that \( (\lambda, z) \in U \) and \( \mathcal{S}(\mu) = J_{\mathcal{X} \mathcal{T}}(\mu) \), there is a unique \( \mu \) such that the following is valid
\[
(\mu = \lambda + \rho M \mathcal{X}, \mu = \arg \min_{\mathcal{X}} L_{\rho}(x, z; \lambda))
\]

(7)

with \( \rho > 0 \) and \( L_{\rho}(x, z; \lambda) \) is the augmented Lagrangian of the general problem (2) given by
\[
L_{\rho}(x, z; \lambda) = \frac{1}{2} \| f(x) + g(z) + \langle \lambda, Mx - z \rangle + \frac{\rho}{2} \| Mx - z \|^2.
\]

**Proof:** (i) \( \mathcal{R}(\mu) = \lambda - \nu \) where \( \nu = T(\nu) \) and \( \alpha = \lambda - \rho z \). \( T \) is the maximal monotone operator given by \( T \in -M \partial f^*(-M^{*} \mu) \). Therefore, \( \alpha = -M \partial f^*(-M^{*} \nu) \) and there is a unique \( \mu \in \partial f^*(-M^{*} \nu) \) such that \( \alpha = -Mz \). From (6) we have \( \mathcal{S}(\mu) = \nu + \rho z = \lambda - \rho \alpha \). But \( \alpha = -Mz \), we conclude that
\[
\mathcal{S}(\mu) = \lambda + \rho Mz.
\]
(ii) Since $f$ is a closed proper convex function, then by the Fenchel-Young inequality [10, Prop 16.9] the expression $x \in \partial f^*(-M^\top \nu)$ is equivalent to $-M^\top \nu \in \partial f(x)$, it follows that $0 \in \partial f(x) + M^\top \nu$. From the output of (5), we have $\nu = \lambda - \rho(z + \alpha)$. We also have $\alpha = Mx$. Then, $\nu = \lambda + \rho(Mx - z)$. It follows that $0 \in \partial f(x) + M^\top \nu$ which translates to $0 \in \partial f(x) + M^\top \lambda + \rho M^\top(Mx - z)$. We conclude that $x = \operatorname{argmin}_x L_\rho(x; z; \lambda)$.

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

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

$$x^{k+1} = \operatorname{argmin}_x L_\lambda(x; z^k, \lambda^k),$$

$$z^{k+1} = \operatorname{argmin}_z L_\rho(x^{k+1}; z, \lambda^k),$$

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

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

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

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

Lemma 5: Define $S_l(\zeta) = (\lambda_l + \prod V_l M_x)\theta$, i.e., the $l^{th}$ sub-block of $S(\zeta) = ((\lambda_l + \prod V_l M_x)\theta, \ldots, (\lambda_l + \prod V_l M_x)\theta)^T$. For each area $A_l$ we have

$$x_l^{k+1} = \operatorname{argmin}_{x_l} \sum_{v \in V_l} f_v(x_v) + \lambda^k T \prod V_l (M_x^v)^T + \frac{\rho}{2} \sum_{v \in V_l} \|\prod V_l (M_x^v)^T - z_l^{k+1}\|^2.$$  

$$z_l^{k+1} = \operatorname{argmin}_{z_l} \lambda^k T z_l + \frac{\rho}{2} \sum_{v \in V_l} \|\prod V_l (M_x^{k+1}) - z_l - z_l^{k+1}\|^2.$$  

$$\lambda_l^{k+1} = \lambda_l^k + \rho \prod V_l (M_x^{k+1}) - z_l^{k+1}.$$  

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

$$x^{k+1} = \operatorname{argmin}_x f(x) + g(z^k) + \langle \lambda^k, Mx - z^k \rangle + \frac{\rho}{2} \|Mx - z^k\|^2$$

$$= \operatorname{argmin}_x f(x) + \langle \lambda^k, Mx \rangle + \frac{\rho}{2} \|Mx - z^k\|^2.$$

This expression is separable into $L$ independent parts $x_l^{k+1}$ where $x_l \in \prod V_l x_l$ and $n_l = \sum_{v \in V_l} n_v$. The sub-block $x_l^{k+1}$ is assigned to $A_l$ and contains only the components of $x^{k+1}$ 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 $x$ 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 $\zeta^{k+1} = S(\zeta^k)$, 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 $\zeta = (\zeta^1, \ldots, \zeta^k)^T$ and suppose $S(\zeta) = (S_1^T(\zeta), \ldots, S_L^T(\zeta))^T$. We define for each area $A_l$, the operator $S_l: \mathcal{Y} \rightarrow \mathcal{Y}$ as

$$S_l(\zeta) = (\zeta_{l+1}^1, \ldots, \zeta_{l+1}^i, \zeta_{l+1}^{i+1}, \ldots, \zeta_L^k)^T,$$

and we use the following theorem.

Theorem 1 [9, Th. 2]: Take a firmly nonexpansive operator $S = (S_1^T, \ldots, S_L^T)^T$ with full domain on $\mathcal{Y}$ and a sequence of i.i.d. random variables $\xi^k$.

Starting from any initial value $\zeta^0$, the random iterates $\zeta^k = S_{\xi^k} \zeta^0$ converges almost surely to a random variable supported by $\operatorname{Fix}(S)$, and we define $\bar{\zeta}_S^k$ as in (9).

We choose a sequence of i.i.d. random variables $\xi^k$, and we iterate $\zeta^{k+1} = S_{\xi^k} \zeta^k$ (i.e., if $\xi^k = l$, then only the nodes of $A_l$ update their variables). Then these iterations converge almost surely to $\zeta = \operatorname{Fix}(S)$.

Thus, if we use the result of Theorem 1 along with the DR splitting method, the random iterates lead to $\bar{\zeta} = J_{\mathcal{R}u}(\zeta)$ which converges almost surely to $\bar{\zeta}(D)$, 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 at each iteration $k$. This leads us to the solution $\bar{\zeta}$ of (1).

To summarize our asynchronous ADMM algorithm, at the $k^{th}$ iteration, we pick a random variable $\xi^{k+1}$ (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 $A$ and $b$ from the linearized DC power flow equations $P_D = P_C - P$ and $P = Bb$, where $P_D$, $P_C$ and $P$ represent respectively the power demanded, generated and transmitted by the agents, $\theta$ is the voltage phase vector and $B$ is the imaginary part of the admittance matrix $Y$, 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
respectively.

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.

VI. CONCLUSION

![Fig. 1. Network of $N = 48$ nodes divided into $L = 12$ overlapping control areas.](image1)

![Fig. 2. Network of $N = 48$ nodes divided into $L = 6$ overlapping control areas.](image2)

![Fig. 3. Performance of the proposed ADMM algorithms with respect to the number of iterations performed, $N = 48, \rho = 1.8$.](image3)

![Fig. 4. Performance of the proposed ADMM algorithms with respect to the number of updates performed, $N = 48, \rho = 1.8$.](image4)
VII. REFERENCES


