Distributed Production-Sharing Optimization and Application to Power Grid Networks

Azary Abboud, Franck Iutzeler, Romain Couillet, Merouane Debbah, Houria Siguerdidjane

Distributed Production-Sharing Optimization and Application to Power Grid Networks

Azary Abboud*, *Student Member, IEEE,* Franck Iutzeler, *Member, IEEE,* Romain Couillet, *Member, IEEE,* Mérouane Debbah, *Fellow Member, IEEE,* and Houria Siguerdidjane, *Member, IEEE*

***Abstract*—Based on recent works on asynchronous versions of the distributed Alternating Direction Method of Multipliers (ADMM) algorithm, we develop and prove the convergence of a distributed asynchronous method for Production-Sharing Problems over networks. The asynchronous nature of the algorithm allows both for the relaxation of the synchronization constraint often inherent to distributed ADMM-based methods and distributed optimization methods at large, but also allows for random local failures to occur in fully centralized methods. These two considerations motivate the application of the method to the Direct-Current Optimal Power Flow (DC-OPF) problem in power transmission networks. Applied to the DC-OPF, this method leads to an overall network optimal production obtained through a sequence of local computations in subareas of the network (each area waking up randomly while the rest of the network is non-operational) and neighboring data exchanges. In another scenario, the DC-OPF is performed via iterations of a centralized network-wide ADMM method which may contain disconnected nodes (in general with low probability and for a short duration). In both cases, this method still converges and thus provides additional flexibility to classical DC-OPF algorithms. The proposed algorithm, inherently designed for networks of overlapping subareas, is then extended to networks of non-overlapping areas. Simulations are carried out on the IEEE-30 and IEEE-118 bus test systems which illustrate the convergence, scalability and effectiveness of the proposed algorithms.**

***Index Terms*—Convex optimization of large scale problems, ADMM, randomized methods, smart grids, flow calculations, distributed control.**

I. Introduction

One of the salient features of power transmission systems is their ability to ensure an optimal transmission throughout the power grid of generated powers at a low economical cost. This naturally leads to the so-called Optimal Power Flow [2] problem, widely studied up until very recently with the emergence of the smart-grid paradigm, see e.g., [3]–[6] for recent works. This problem can be generally cast as that of finding an operational point leading to the least global generation cost subject to power flow equations (such as Kirchhoff and Ohm's laws) and other operational constraints [7]. The physical constraints make the problem nonlinear and non-convex [8], but as far as power transmission is concerned linear approximations are considered accurate and significantly simplify the problem [6].

Our focus is here on the Direct-Current Optimal Power Flow (DC-OPF) problem, which is a linear DC approximation of the original OPF problem [2]. In the DC-OPF formulation, voltage angles differences are considered close to zero and voltage magnitudes are fixed.[1] Embracing the smart-grid paradigm, we seek here to minimize the power generation cost of the grid while taking advantage of its advanced communication characteristics [2], [11]. A classical method to solve problems of the OPF class relies on a central fusion center which collects data from all the network agents and centrally evaluates the optimal system parameters before communicating these values to the corresponding agents [12]. However, on the one hand, this centralized optimization method is rather prohibitive in large-scale and sparse networks due to the communication burden and the delay induced by the required computations and communication [12], [13]. On the other hand, the privacy concerns of the end users or the grid owners in case of network with multiple authorities, and the recent tendency toward multinational electricity markets, make it impossible to efficiently implement the centralized optimization scheme. Besides, several security issues such as single point of failure, bottleneck and agreement on decisions, may arise in case of centralized optimization [12], [14].

To reduce the computational burden and the inherent weakness of a single centralized computation unit, one may leverage on the presence of independent processors in each component of the power grid [13] to increase the available computational capacity. This implies having a distributed computing platform, which naturally calls for the intervention of distributed optimization approaches in order to solve the management problems in the power grid. This distributed scheme is specifically preferred when power sources are located at different parts of a large network while being under the control of multiple companies or authorities. We thus aim here to solve the DC-OPF problem by using several processors distributed all over the network. Distributed DC-OPF approaches where multi-utility transmission systems are divided into autonomously managed area were proposed in [15], [16] and in references therein. The distributed computation is a preferable approach that helps in reducing the data exchange overhead and the processing time, leading to a smart optimization of the power system. The pioneering work of

Azary Abboud and Mérouane Debbah are with Laneas Group, Supélec, Gif-sur-Yvette, France. Franck Iutzeler is with the ICTEAM Institute, Université Catholique de Louvain, Louvain-la-Neuve, Belgium. Romain Couillet is with the Telecommunication Department, Supélec, Gif-sur-Yvette, France. Houria Siguerdidjane is with the Automatic and Control Department, Supélec, Gif-sur-Yvette, France. Email: firstname.lastname@supelec.fr. Part of the material of this paper has been presented at the 39th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Florence, May 9, 2014 [1].

[1] Although the DC-OPF constitutes a simplified version of the OPF, it is used by several North American Independent System Operators (ISO) for day-to-day operation [3], [9], [10].

Tsitsiklis and Bertsekas [17], [18] in this field has led to extensive studies on distributed optimization techniques and several methods were developed to solve various optimization problems in a distributed manner [19], [20].

Several methods can be used to perform distributed optimization, either in a sequential or in a parallel manner, such as Lagrangian relaxation [21], augmented Lagrangian [15], approximate Newton directions in conjunction with standard Lagrangian [16], [22] as well as the auxiliary problem principle [23]. These methods are however computationally cumbersome [24], [25] as they share the same synchronization that ties the different areas and demand strong coordination (each area optimizes its variables in every iteration and communicates with its adjacent areas before moving to another iteration). The synchronous burden tends to cause degradation in performance mainly because of the inherent communication overhead and the idle processing time [26]. Among those methods, the technique known as the Alternating Direction Method of Multipliers (ADMM), while sharing the same performance limitations, has the advantage of converging faster than the aforementioned methods [27], [28], of being easier to implement, and of requiring less time and computational capacity at each processor. The structure of the iterations in this method makes their distribution among the existing processors easier than the previously mentioned methods [28].

Recently [29] proposed a fully asynchronous distributed version of the ADMM that is applied sequentially by randomly selecting overlapping areas of the network, the union of which covers the whole network. Despite the asynchrony, by relying on a careful analysis of the proximal splitting method [27] of which the ADMM is a special instance, it is proved that such an algorithm converges. As opposed to [29], which was not designed for DC-OPF problems specifically, in the present article, we additionally consider that the problem under study (being DC-OPF compliant) is such that the optimization variables are spread across the network (instead of being shared among the nodes) and that these variables have to satisfy some additional box constraints. In power grid terms, the power generated in the network are local quantities only bounded to an indirect user satisfaction constraint, and must be such that maximal capacities of power lines are not exceeded. In [29] the distributed agents seek to find consensus on an (unconstrained) unique parameter minimizing a node-wise aggregate cost. As such, although closely related, our problem formulation differs from [29]. We formulate the problem in a more generic form than the DC-OPF, the Production-Sharing Problem. Each agent has a production ability at a certain cost and can exchange its resources with its neighbors through the links connecting them. This generic formulation makes it applicable to problems involving distributed computations other than power systems (please refer to [30] for an example). Because imposing overlapping areas in a power network structure may lead to practical management problems, inspired by the method of passing adjacent variables [31] that allows to convert networks of non-overlapping and independent areas into a network of overlapping areas, we extend the approach from [29] to cover this context.

The rest of the article is organized as follows. Section II defines the convex optimization problem with linear constraints for which we provide in Section III the distributed formulation and solution using ADMM. Then, in Section IV, relying on the distributed Production-Sharing optimization w/ ADMM algorithm, we develop the asynchronous distributed Production-Sharing w/ ADMM algorithm. In Section V, we extend our algorithm to the non-overlapping area scenario. Section VI is devoted to the application of the algorithms on the linear DC-OPF problem. The results are presented on the power grid network given by the IEEE-30 bus test system [32], then on the IEEE-118 bus test system [32] in order to verify the scalability of the proposed algorithms. Lastly, Section VII contains our concluding remarks.

Notations

Lowercase (resp. uppercase) boldface characters will denote vectors (resp. matrices). Furthermore, x_i denotes the i^{th} entry of vector $\boldsymbol{x}$ and A_{ij} stands for the i^{th} row j^{th} column entry of matrix $\mathbf{A}$. $[\mathbf{A}]_i$ corresponds to the i^{th} row of the matrix $\mathbf{A}$. $\boldsymbol{x}^T$ and $\boldsymbol{A}^T$ denote respectively the transpose of vector $\boldsymbol{x}$ and matrix $\boldsymbol{A}$. In addition, we will note $\boldsymbol{x} = [\boldsymbol{x}_1; \ldots; \boldsymbol{x}_N]$ the vector formed by the vertical concatenation of vectors $(\boldsymbol{x}_i)_{i=1,\ldots,N}$.

II. Problem formulation

In this section, we will formulate a generic sharing optimization problem which encompasses as a particular case the DC-OPF problem.

A. A Production-Sharing Problem on a Graph

Consider a group of N agents connected by an undirected connected graph of N nodes, $G = (V, E)$, where $V = \{1, ..., N\}$ is the set of nodes[2] and E is the set of bidirectional edges. We will say that i and j are neighbors, or $i \sim j$, if $\{i, j\} \in E$, and we will note $\mathcal{N}_i = \{j \in V : i \sim j\}$ the set of neighbors of i. We introduce symmetric edge weights $w_{ij} = w_{ji}$ such that $w_{ij} > 0$ if and only if $\{i, j\} \in E$ and 0 elsewhere, the matrix $\mathbf{W}$ defined by the entries $\{w_{ij}\}$ is called the weighted adjacency matrix, the diagonal matrix $\mathbf{D}$ such that $D_{ii} = \sum_{j \in V} w_{ij}$ is called the degree matrix, and finally $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is called the Laplacian matrix of the graph.

We look at the case where each agent $j \in V$ produces and/or consumes some resource (typically electric power) represented by a real value, and the agents can exchange this resource using the links of the aforementioned graph. More precisely, each agent has a production ability in the form of an interval (reduced to $\{0\}$ for non-producers) and a cost function depending on the quantity produced (which will be assumed convex in the following). The natural problem ensuing is thus to minimize the total production cost under the constraints that i) the demands are satisfied and ii) the network is able to dispatch correctly the resource. This last point can take multiple forms depending on the resource type and will be made explicit when considering the DC-OPF problem next;

[2]We will use the terms nodes and agents interchangeably.

however, we will write it generically as *"the vector of the differences between production and consumption at each agent belongs to the span of the Laplacian matrix"*.

We write this problem formally as a convex optimization problem[3] with linear constraints:

Production-Sharing Problem

$$\begin{array}{ll} \min\limits_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}\in\mathbb{R}^N} & \sum_{j\in V} f_j(x_j) \\ \text{subject to} & \forall j \in V, \underline{r_j} \leq x_j \leq \overline{r_j} \\ & \begin{cases} \boldsymbol{y} = \boldsymbol{x} - \boldsymbol{d} \\ \boldsymbol{y} = \mathbf{L}\boldsymbol{z} \end{cases} \end{array} \tag{1}$$

where $\boldsymbol{x}$ is the resource production vector, $\boldsymbol{d}$ is the demand vector, f_j is the production cost function at agent j and $[\underline{r_j}, \overline{r_j}]$ is its production range, $\mathbf{L}$ is the Laplacian of graph G, and finally $\boldsymbol{y}$ and $\boldsymbol{z}$ are intermediate vectors.

In order to be able to perform meaningful derivations, we will make the following standard assumption.

Assumption 1. *The functions $\{f_j\}_{j\in V}$ are convex, proper, lower semi-continuous and the graph G is undirected and connected. Furthermore, the set of minimizers of Problem* (1) *is non-empty.*

A simple example: *Consider a complete graph with unit edge weights and say that each of the N agents has a unit demand ($\boldsymbol{d} = [1, ..., 1]^T$). Because the graph is complete $\mathbf{L} = N\mathbf{I} - \mathbf{J}$ where $\mathbf{I}$ is the identity matrix and $\mathbf{J}$ is filled with ones. This means that the second condition of the above problem simply translates to* "the mean of the resource production vector $\boldsymbol{x}$ is one".
Now, assume that half of the agents cannot produce any resource (for them, $\underline{r_j} = \overline{r_j} = 0$ thus $x_j = 0$) while the other half can produce any quantity x ($\underline{r_j} = 0, \overline{r_j} = +\infty$) with cost ax (for every producer j, $f_j(x_j) = ax_j$ and $a > 0$); it is immediate to see that each producer will produce the same quantity (as they are indistinguishable in the problem with a complete unweighted graph) and this quantity must be 2 (as only half of the agents produce and the mean of the production vector has to be one).

Now, let us see how this problem can encompass problems encountered in power networks such as the DC-OPF.

B. DC-OPF as a Production-Sharing Problem on a Graph

A power network consists of a set of agents/buses V. Each agent $j \in V$ can *generate* a power p_j^g with a *cost* $f_j(p_j^g)$ in its generation ability range $[\underline{r_j}, \overline{r_j}]$ (possibly reduced to $\{0\}$ if the agent is not a generator) as well as a power demand p_j^d.

The agents are linked by transmission lines E, $G = (V, E)$ forming a connected undirected graph. Let θ_j be the *phase* of the current produced by node $j \in V$. In the DC model, the power flow on the transmission line from agent i to j is proportional to their phase difference $\theta_i - \theta_j$ through a constant B_{ij} representing the imaginary part of the element in the i^{th} row and j^{th} column of the bus admittance matrix $\boldsymbol{Y}$ [2], [33], [34]. We also introduce the *excess power* of node j, $p_j^e = p_j^g - p_j^d$.

Kirchoff's Voltage law and the DC model conditions impose the excess power (which is possibly negative) to be transmitted through the adjacent transmission lines. Formally, for every agent j, this sums up to $p_j^e = \sum_{i\sim j} B_{ji}(\theta_j - \theta_i) = (\sum_{i\sim j} B_{ji})\theta_j - \sum_{i\sim j} B_{ji}\theta_i$. This condition can be rewritten in a vectorial manner as $\boldsymbol{p}^e = \mathbf{L}^B\boldsymbol{\theta}$ where $\mathbf{L}^B$ is the Laplacian matrix defined from the weighted adjacency matrix $\mathbf{B}$ as defined previously and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_N)$ is the phase vector. Let $\boldsymbol{p}^g = (p_1^g, \ldots, p_N^g)$ be the power generation vector and $\boldsymbol{p}^d = (p_1^d, \ldots, p_N^d)$ be the power demand vector. Then, the DC-OPF Production-Sharing Problem can be formulated as follows,

DC-OPF Production-Sharing Problem

$$\begin{array}{ll} \min\limits_{\boldsymbol{p}^g,\boldsymbol{p}^e,\boldsymbol{\theta}\in\mathbb{R}^N} & \sum_{j\in V} f_j(p_j^g) \\ \text{subject to} & \forall j \in V, \underline{r_j} \leq p_j^g \leq \overline{r_j} \\ & \begin{cases} \boldsymbol{p}^e = \boldsymbol{p}^g - \boldsymbol{p}^d \\ \boldsymbol{p}^e = \mathbf{L}^B\boldsymbol{\theta} \end{cases} \end{array} \tag{2}$$

It is straightforward to see that the above problem has the same form as Problem (1). It is important to notice that the matrix $\mathbf{B}$ (through its Laplacian) suffices to structure the generated power vector to comply with electrical equalities across the power network.

III. Distributed Production-Sharing

The Production-Sharing Problem introduced in the previous section is essentially centralized. Indeed, while the underlying graph is present through the Laplacian matrix, the condition $\boldsymbol{y} = \mathbf{L}\boldsymbol{z}$ on the transmission of the excess production implies a coordinated action of the whole network. In this section, we first design an equivalent problem where the network condition is split into overlapping subgraphs and add an additional indicator function in order to ensure the equivalence with the former problem. Then, after noticing that this new problem is well suited for the Alternating Direction Method of Multipliers, we derive a distributed Production-Sharing algorithm solving our problem.

A. Distributed Formulation of the Problem

Starting from Problem (1), as every agent can control its resource generation (and thus its excess) as well as its network intermediate vector (its phase in the DC-OPF case), it is convenient to stack these variables into a vector of size $3N$, $\boldsymbol{u} = [\boldsymbol{x}; \boldsymbol{y}; \boldsymbol{z}]$.

With this new variable, it is useful to introduce a function F corresponding to the function to minimize in (1) applied to $\boldsymbol{u}$ and encompassing the range condition:

$$\begin{array}{rccl} F: & \mathbb{R}^{3N} & \longrightarrow & (-\infty, +\infty] \\ & \boldsymbol{u} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix} & \longmapsto & \begin{cases} \sum\limits_{j\in V} f_j(x_j) & \text{if } \forall j, x_j \in [\underline{r_j}, \overline{r_j}] \\ +\infty & \text{elsewhere} \end{cases} \end{array} \tag{3}$$

[3] the introduction of the excess variable $\boldsymbol{y}$ is not mandatory but it is kept for clarity purposes.

One can remark that, provided that the functions $\{f_j\}_{j\in V}$ are convex, F is also convex. Now, the rest of the conditions can be written linearly as:

$$\underbrace{\begin{bmatrix} \mathbf{I} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{L} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \end{bmatrix}}_{\boldsymbol{u}} = \underbrace{\begin{bmatrix} \boldsymbol{d} \\ \mathbf{0} \end{bmatrix}}_{\tilde{\boldsymbol{d}}} \tag{4}$$

Hence, Problem (1) rewrites as

$$\begin{aligned} \min_{\boldsymbol{u}\in\mathbb{R}^{3N}} \quad & F(\boldsymbol{u}) \\ \text{subject to} \quad & \mathbf{A}\boldsymbol{u} = \tilde{\boldsymbol{d}} \end{aligned} \tag{5}$$

Let us recall that in our scheme, each agent j has the knowledge of its cost function f_j, its production-related variables x_j, y_j, z_j, and the j^{th} row/column of the Laplacian matrix $\mathbf{L}$. It is then straightforward to see that, whereas the upper part of the equality in (4) is local to each agent, the rows of the lower part corresponds to the network connections.

In order to derive a distributed algorithm, a common solution is to introduce one variable per condition line per agent and an indicator function ensuring that for each condition line, the sum of the agents related variables is equal to the corresponding value in $\tilde{\boldsymbol{d}}$. Formally, define the $6N^2 \times 3N$ matrix

$$\tilde{\mathbf{A}} = \begin{bmatrix} \text{diag}([\mathbf{A}]_1) \\ \vdots \\ \text{diag}([\mathbf{A}]_{2N}) \end{bmatrix} \tag{6}$$

and the indicator function[4]

$$\begin{aligned} G: \quad \mathbb{R}^{6N^2} &\longrightarrow (-\infty, +\infty] \\ \boldsymbol{v} &\longmapsto \begin{cases} 0 \quad \text{if } \forall i = 1\dots 2N, \sum_{j=3(i-1)N+1}^{3iN} v_j = \tilde{d}_i \\ +\infty \quad \text{elsewhere,} \end{cases} \end{aligned} \tag{7}$$

where, diag($[\mathbf{A}]_i$) is the diagonal matrix whose diagonal entries are the elements of the row $[\mathbf{A}]_i$.

One can note that vector $\boldsymbol{v}$ has a lot more components than actually needed. Due to the sparsity of the matrix $\mathbf{A}$, several components in $\boldsymbol{v}$ are equal to zero and can be omitted. But, for notation simplicity, the full vector will be kept until we provide the actual algorithm derivation, where we will eliminate unnecessary components and precise the actual number of variables to be updated per agent.

Finally, we obtain the following distributed problem.

Distributed Production-Sharing Problem

$$\min_{\boldsymbol{u}\in\mathbb{R}^{3N}} \quad F(\boldsymbol{u}) + G(\tilde{\mathbf{A}}\boldsymbol{u}) \tag{8}$$

By construction, the solutions of this problem allow to derive the solutions of the original problem as mentioned in the following lemma.

[4]throughout this paper, we call indicator function, a function which returns 0 when the argument is in some set C and $+\infty$ elsewhere. It is immediate to check that is C is closed and convex, then the indicator of set C is convex and lower semi-continuous.

Lemma 1. *Let Assumption 1 hold. Then, the solutions of Problem* (8) *are of the form* $\boldsymbol{u}^\star = [\boldsymbol{x}^\star; \boldsymbol{y}^\star; \boldsymbol{z}^\star]$ *where* $(\boldsymbol{x}^\star, \boldsymbol{y}^\star, \boldsymbol{z}^\star)$ *is a solution of Problem* (1).

B. Application of the ADMM to the Distributed Production-Sharing Problem

Writing an iteration of the ADMM algorithm on Problem (8) leads to the following set of equations

$$\boldsymbol{u}^{k+1} = \underset{\boldsymbol{u}}{\text{argmin}} \left\{ F(\boldsymbol{u}) + \frac{\rho}{2} \left\| \tilde{\mathbf{A}}\boldsymbol{u} - \boldsymbol{v}^k + \frac{\boldsymbol{\lambda}^k}{\rho} \right\| \right\}^2 \tag{9a}$$

$$\boldsymbol{v}^{k+1} = \underset{\boldsymbol{v}}{\text{argmin}} \left\{ G(\boldsymbol{v}) + \frac{\rho}{2} \left\| \tilde{\mathbf{A}}\boldsymbol{u}^{k+1} - \boldsymbol{v} + \frac{\boldsymbol{\lambda}^k}{\rho} \right\| \right\}^2 \tag{9b}$$

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho \left(\tilde{\mathbf{A}}\boldsymbol{u}^{k+1} - \boldsymbol{v}^{k+1} \right) \tag{9c}$$

where $\rho > 0$ is a free hyper-parameter and $\boldsymbol{\lambda}$ is the vector of Lagrangian multipliers.

Using the definitions of function F and G, and matrix $\tilde{\mathbf{A}}$, we obtain the following algorithm after some derivations reported in Appendix A.

Distributed Production-Sharing optimization w/ ADMM

1) Initialize $\boldsymbol{u}$ and $\boldsymbol{\pi}$ to the initial values $\boldsymbol{u}^0$ and $\boldsymbol{\pi}^0$.
2) At iteration k every agent $j = 1, \dots, N$ performs the following steps :
   a) Update x_j, y_j and z_j

$$\begin{aligned} x_j^{k+1} = \underset{x_j}{\text{argmin}} \; & f_j(x_j) + \frac{\rho}{2} \| x_j + \frac{\pi_{1,j}^k}{\rho} - x_j^k + \\ & \frac{1}{d_1(j)} \left(x_j^k - y_j^k - d_j \right) \|^2 \end{aligned} \tag{10a}$$

$$y_j^{k+1} = y_j^k + \frac{\pi_{1,j}^k + \pi_{2,j}^k}{2\rho} + \frac{r_{1,j}^k}{2d_1(j)} + \frac{r_{2,j}^k}{2d_2(j)} \tag{10b}$$

$$z_j^{k+1} = z_j^k - \frac{1}{\sum_{i=1}^N L_{ij}^2} \sum_{i=1}^N L_{ij} \left(\frac{\pi_{2,i}^k}{\rho} + \frac{r_{2,i}^k}{d_2(i)} \right) \tag{10c}$$

   with $\forall i = 1, ..., N$, $d_1(i) = 2$ and $d_2(i) = 2 + |\mathcal{N}_i|$
   b) Communicate z_j^{k+1} and y_j^{k+1} to neighboring nodes.
   c) For each constraint $i \in I_j = \{i; \exists \boldsymbol{A}_{ip} \neq 0, p = \{j, j+N, j+2N\}\}$, compute

$$r_{1,i}^{k+1} = x_i^{k+1} - y_i^{k+1} - d_i \tag{11a}$$

$$r_{2,i}^{k+1} = -y_i^{k+1} + \sum_{j=1}^N L_{ij} z_j^{k+1} \tag{11b}$$

$$\pi_{1,i}^{k+1} = \frac{\rho}{d_1(i)} r_{1,i}^{k+1} + \pi_{1,i}^k \tag{12a}$$

$$\pi_{2,i}^{k+1} = \frac{\rho}{d_2(i)} r_{2,i}^{k+1} + \pi_{2,i}^k \tag{12b}$$

3) If the stopping criterion is not satisfied, increase k and go to 2). Otherwise, retain x_j^{k+1}, y_j^{k+1} and z_j^{k+1}.

This algorithm is effectively distributed; agent j computes and maintains its personal variables using local information $(f_j, L_{j,\cdot})$ and exchanges with its neighbors. We can also remark that, contrary to the consensus-based distributed ADMM [35], the relations between the agents are directly linked to the original constraints and not due to a reformulation of an originally centralized problem.

Theorem 1. *Let Assumption 1 hold. Then, the sequence* $(\boldsymbol{u}^k)_{k>0} = ([\boldsymbol{x}^k, \boldsymbol{y}^k, \boldsymbol{z}^k])_{k>0}$ *generated by* Distributed Production-Sharing optimization w/ ADMM *converges to* $\boldsymbol{u}^\star = [\boldsymbol{x}^\star; \boldsymbol{y}^\star; \boldsymbol{z}^\star]$ *where* $(\boldsymbol{x}^\star, \boldsymbol{y}^\star, \boldsymbol{z}^\star)$ *is a solution of Problem* (1).

The proof of Theorem 1 is straightforward as Distributed Production-Sharing optimization w/ ADMM relies on ADMM applied to a sum of two functions that are proper lower semi-continuous and convex by construction (F is constructed from the (f_i) with Assumption 1; G is a projection onto a linear space). These properties are the sole conditions for the convergence of ADMM (see e.g. [36]) and thus for our algorithm.

The convergence speed is expected to be at least $O(1/k)$ as the reasoning from [37] can be adapted; nevertheless, linear convergence speed have also been proved for other ADMM-based algorithms with additional assumptions on F [38], [39].

IV. Randomized Distributed Production-Sharing

A. Theoretical foundations

In the previous section, we presented an algorithm for solving Problem (1) in a distributed manner over the graph associated with $\mathbf{L}$. However, it is not always possible nor appropriate to compute a full iteration of this algorithm. For example, some agents of the network can randomly fail to exchange or compute their update. Also, it may be faster to solve the problem by taking into account only a random subset of the agents/links at each iteration in the spirit of *mini-batch* algorithms.

For these reasons, it is interesting to consider a randomized version of the previous algorithm where only some parts of the network are active at a given iteration.

This can be achieved by following the ADMM randomization scheme proposed in [29]. At each iteration k, this method consists of picking a random set of coordinates ξ^k (or equivalently a random subset of agents and links), and performing the updates of Eqs. (9b) and (9c) only for the coordinates of ξ^k, the other ones being kept at their former value. As for the first update Eq. (9a), only the coordinates needed for the partial update of Eqs. (9b) and (9c) are to be computed.

In this way, provided that the random coordinate selection sequence $(\xi^k)_{k>0}$ is independent and identically distributed (i.i.d.) and such that the selection probability is positive for every coordinate, the randomized algorithm converges almost surely to a sought solution [29] (see also [40] for refinements).

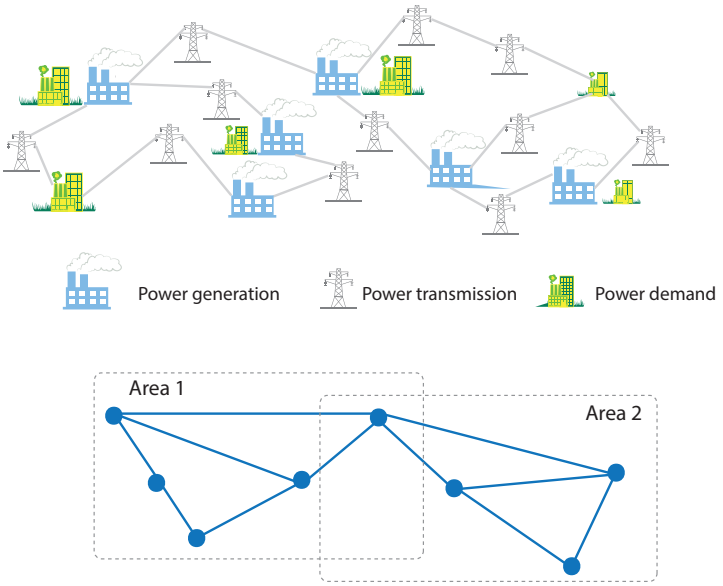


Fig. 1. Power grid and its graph presentation divided into 2 overlapping areas.

B. Randomized algorithm

In the case of a network of agents with multiple authorities or when the network is divided into multiple micro-networks, it is advised to use area-based distributed mechanisms to manage efficiently the network. Interestingly, the components of $\boldsymbol{\lambda}$ and $\boldsymbol{v}$ are linked either to a node or to an edge of the network; thus, updating only a part of these components sums up to performing computations and exchanges only on a subgraph.

Let us decompose the graph G into L connex areas A_ℓ, $l \in \{1, \ldots, L\}$ (see Fig. 1); area A_ℓ is a graph with vertices $V_\ell \subset V$ and edges $E_\ell = \{\{i,j\}; (i,j) \in V_\ell^2\} \cap E$. Each area A_ℓ will act as a local processor exchanging data with its closest neighbors when selected by the random sequence $(\xi^k)_{k>0}$; *i.e.* at times k such that $\ell \in \xi^{k+1}$. In order to ensure convergence to the sought solution [29], we will assume that i) the areas are connected and encompass the whole graph; and ii) $(\xi^k)_{k>0}$ is i.i.d. and for all $l \in \{1, \ldots, L\}$, the selection probability for area A_ℓ is positive: $\mathbb{P}[\ell \in \xi^1] > 0$.

Assumption 2. *For any* $l \in \{1, ..., L\}$, *let* $G(V_\ell) \triangleq (V_\ell, E_\ell)$ *be the sub-graph of area* A_ℓ, *the following properties are assumed:*

1) $\bigcup_{l=1}^{L} V_\ell = V$,
2) $\bigcup_{l=1}^{L} G(V_\ell)$ *is connected.*

Furthermore, the area selection sequence $(\xi^k)_{k>0}$, *valued in the set of the subsets of* $\{1, ..., L\}$, *is independent and identically distributed and such that* $\forall \ell,\ \mathbb{P}[\ell \in \xi^1] > 0$.

This means that in order to ensure convergence, the areas must overlap (*i.e.*, have at least an agent in common, see Fig. 1), which can be restrictive. Let us first state the asynchronous algorithm with overlapping areas and extend it to the non-overlapping case in next section.

At each iteration k, applying Eqs. (9a)-(9c) but keeping only

the components related to the areas[5] of ξ^{k+1} in the two latter equations leads to the following algorithm (the calculations are quite similar as the ones in Appendix A).

Asynchronous Distributed Production-Sharing optimization w/ ADMM

1) Initialize $\boldsymbol{u}$ and $\boldsymbol{\pi}$ to the initial values $\boldsymbol{u}^0$ and $\boldsymbol{\pi}^0$.
2) At iteration k, a random area $A_{\xi^{k+1}}$ becomes operational, and each agent $j \in V_{\xi^{k+1}}$ performs the following:
   a) Update x_j, y_j and z_j using equations (10a), (10b) and (10c)
   b) Communicate z_j^{k+1} and y_j^{k+1} to the neighboring nodes.
   c) For each constraint $i \in I_j$, compute $r_{1,i}^{k+1}$, $r_{2,i}^{k+1}$ $\pi_{1,i}^{k+1}$ and $\pi_{2,i}^{k+1}$ using equations (11a) to (12b) respectively.
3) Every non-selected agent $j \notin V_{\xi^{k+1}}$ keeps its former values $x_j^{k+1} = x_j^k$, $z_j^{k+1} = z_j^k$, ...
4) If the stopping criterion is not satisfied, increase k and go to 2). Otherwise, retain x_j^{k+1}, y_j^{k+1} and z_j^{k+1}.

We remark that this algorithm is effectively randomized in the sense that only the agents of the chosen areas perform computations and exchanges, the others simply keeping their former variables. The performance of this algorithm as well as details about the choice of the area selection sequence will be provided in Section VI.

Theorem 2. *Let Assumptions 1 and 2 hold. Then, the sequence* $(\boldsymbol{u}^k)_{k>0} = ([\boldsymbol{x}^k, \boldsymbol{y}^k, \boldsymbol{z}^k])_{k>0}$ *generated by* Asynchronous Distributed Production-Sharing optimization w/ ADMM *converges almost surely to* $\boldsymbol{u}^\star = [\boldsymbol{x}^\star; \boldsymbol{y}^\star; \boldsymbol{z}^\star]$ *where* $(\boldsymbol{x}^\star, \boldsymbol{y}^\star, \boldsymbol{z}^\star)$ *is a solution of Problem* (1).

The proof of Theorem 2 is derived from [29, Th. 2] as Asynchronous Distributed Production-Sharing optimization w/ ADMM relies on i) ADMM applied to a sum of two functions that are proper lower semi-continuous and convex by construction (see Theorem 1); and ii) an i.i.d randomization veryfying Assumption 2 in order to match the assumptions of [29, Th. 2]. Again, the convergence speed in the mean square sense is expected to be $O(1/k)$ from [41].

As mentioned before, the convergence result is limited to the case where these areas overlap. This is equivalent to having some agents falling under the authority of multiple areas. In power grid networks, this may result in conflicts in decision making as a coordination is thus required between these authorities. In the following section, we extend our algorithms to the case of non-overlapping areas by introducing dummy nodes between the areas. Subsequently, the areas become independent as every agent refers to only one area.

[5]It may seem counter-intuitive that the active set of nodes at times k is denoted by ξ^{k+1}, this index *mismatch* is justified so that every variable at time $k+1$ (denoted $\cdot^{k+1}$) is $\mathcal{F}^{k+1}$-measurable where $\mathcal{F}^{k+1}$ is the sigma-field induced by $(\boldsymbol{u}^0, \boldsymbol{\pi}^0, \xi^1, ..., \xi^{k+1})$. Indeed, the only randomness in the algorithm comes from the area selection process and variables at time $k+1$ are derived from the last selection variable (ξ^{k+1}) in a deterministic manner.

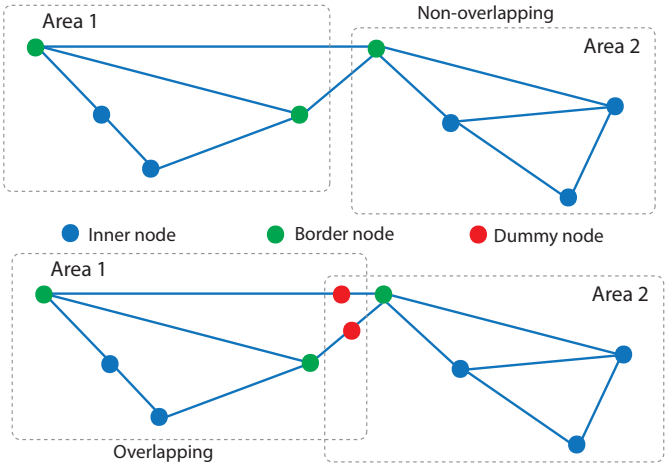


Fig. 2. Network of 2 non-overlapping areas converted to overlapping network.

V. Extension to the non-overlapping areas case

In this section, we extend our previous result to the case where non-overlapping areas are activated at each iteration. We focus here on the aforementioned Distributed Production -Sharing algorithm but the reasoning below can be easily adapted to a large class of distributed ADMM-based algorithms.

As opposed to Assumption 2, we divide here the graph G (linked to Laplacian $\mathbf{L}$ as described in the previous section) into L strictly separated areas A_ℓ, $\ell \in \{1, \dots, L\}$ where every node belongs to exactly one area. A node will be called a *border node* if and only if it is connected to a node in a different area, and an *inner node* otherwise.

Starting from the initial Production-Sharing Problem (1), we aim at decomposing it into L sub-problems, each associated to an area A_ℓ. Obviously, some constraints couple multiple areas together and thus cannot be assigned to one area. These coupling constraints are related to the connections between border nodes. Inspired by the method of passing adjacent variables [31], we propose an approach that transforms our problem so that only the agents of a given area are active at each iteration. For this, we add a *dummy node* between each pair of connected border nodes of different areas as shown in Fig. 2. These dummy nodes are not associated with any cost function but only serve to rewrite the constraints coupling adjacent areas using a *shared variable*, that will have to be exchanged as we will see later.

Thus, the changes to be applied to the initial optimization problem are:

- Add chosen dummy nodes;
- Rewrite the coupling constraints accordingly.

Practically, the problem becomes

Non-overlapping Distributed Production-Sharing Problem

$$\begin{array}{ll} \min\limits_{\mathring{\boldsymbol{x}},\mathring{\boldsymbol{y}},\mathring{\boldsymbol{z}}\in\mathbb{R}^{\mathring{N}}} & \sum_{j\in\mathring{V}} \mathring{f}_j(\mathring{x}_j) \\ \text{subject to} & \forall j\in\mathring{V}, \underline{\mathring{r}_j} \le \mathring{x}_j \le \overline{\mathring{r}_j} \\ & \begin{cases} \mathring{\boldsymbol{y}} = \mathring{\boldsymbol{x}} - \mathring{\boldsymbol{d}} \\ \mathring{\boldsymbol{y}} = \mathring{\mathbf{L}}\mathring{\boldsymbol{z}} \end{cases} \end{array} \tag{13}$$

where

- $\mathring{V} = V \bigcup \overset{\bullet}{V}$ is the new set of $\mathring{N}$ agents composed of the set of the N original agents V plus the set of the $\overset{\bullet}{N}$ dummy agents $\overset{\bullet}{V}$.
- for all $j \in V$, $\mathring{f}_j$, $\underline{\mathring{r}_j}$, $\overline{\mathring{r}_j}$, and $\mathring{\boldsymbol{d}}_j$ are respectively equal to their original values f_j, $\underline{r_j}$, $\overline{r_j}$, and $\boldsymbol{d}_j$;
- for all $d \in \overset{\bullet}{V}$, $\mathring{f}_d \equiv 0$, $\underline{\mathring{r}_d} = \overline{\mathring{r}_d} = 0$, and $\mathring{\boldsymbol{d}}_d = 0$;
- the design of $\mathring{\mathbf{L}}$ from $\mathbf{L}$ is a bit more tedious and explained in Appendix B:

$$\mathring{L}_{jj} = L_{jj} + \frac{1}{2}\sum_{d\sim j} \mathring{L}_{jd}, \quad \mathring{L}_{jd} = -2L_{ij}, \quad \mathring{L}_{dd} = 4L_{ij},$$

$$\mathring{L}_{ij} = L_{ij}, \text{ if } i \text{ and } j \text{ belong to the same area.}$$

With this formulation we can see that Problems (1) and (13) are equivalent. Following the reasoning of the previous sections while considering the modified problem, one can derive a new distributed algorithm using ADMM that accounts for the modified functions and matrices. A solution of the original problem of non-overlapping areas can then extracted from a solution of the modified overlapping problem by omitting the entries related to the dummy nodes. One can remark that in the new algorithm, when an area is updated, it only needs local information plus the value of the variables corresponding to the dummy nodes. An iteration is thus now composed of two parts: i) local computations and exchanges in the selected area; ii) communication of border-related values to the adjacent areas.

VI. Implementations and Simulations

In this section, we implement our algorithms on the DC-OPF problem [2]. Simulations are first carried out on the IEEE−30 bus test system [32] then, on the IEEE−118 bus test system.

In the DC-OPF context defined in II-B, Problem (5) writes

$$\begin{array}{ll} \min\limits_{\boldsymbol{u}\in\mathbb{R}^{3N}} & F(\boldsymbol{u}) \\ \text{subject to} & \mathbf{A}\boldsymbol{u} = \tilde{\boldsymbol{d}} \end{array} \tag{14}$$

where,

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{I} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{L}^B \end{bmatrix}$$

and

$$\tilde{\boldsymbol{d}} = (p_1^D, \ldots, p_N^D, 0, \ldots, 0)^T \in \mathbb{R}^{2N}.$$

Note that the cost only depends on the generated power. We consider here quadratic cost functions, more precisely for each *generator* agent j, we take $f_j(x_j) = c'_j x_j^2 + c_j x_j$ for $p^g_{j,\min} \le x_j \le p^g_{j,\max}$ and $+\infty$ otherwise, with the coefficients given in Table I.

TABLE I
Generators data, IEEE−30 Bus Test System

Node	$p^g_{j,\min}$	$p^g_{j,\max}$	c'_j	c_j
1	0	100	0.037	20
2	0	30	0.01	20
6	0	80	0.0175	10
10	0	35	0.0083	10
13	0	20	0.01	15
15	0	50	0.0625	10
19	0	20	0.01	15
24	0	30	0.0250	20
27	0	40	0.0250	20

TABLE II
IEEE−30 bus test system division into 3 overlapping areas

Area	Nodes	Number of nodes
A_1	1-11,17,20,28	14
A_2	3,4,12-20,23	12
A_3	10,21-30	11

In the following, we apply our asynchronous distributed algorithm to the DC-OPF problem using the network of the IEEE−30 bus test system.

A. Overlapping areas

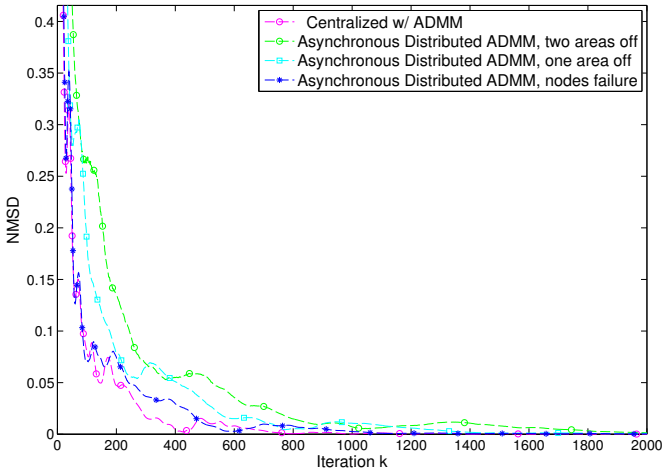
The cost functions (f_j) are defined as above and ρ is set to 1. Eq. (10a) then simplifies to,

$$x_j^{k+1} = \underset{[p^g_{j,\min}, p^g_{j,\max}]}{\Pi} \left[\frac{\frac{1}{2}x_j^k - c_j - \pi_{1,j}^k + \frac{1}{2}(y_j^k + d_j)}{2c'_j + 1} \right]$$

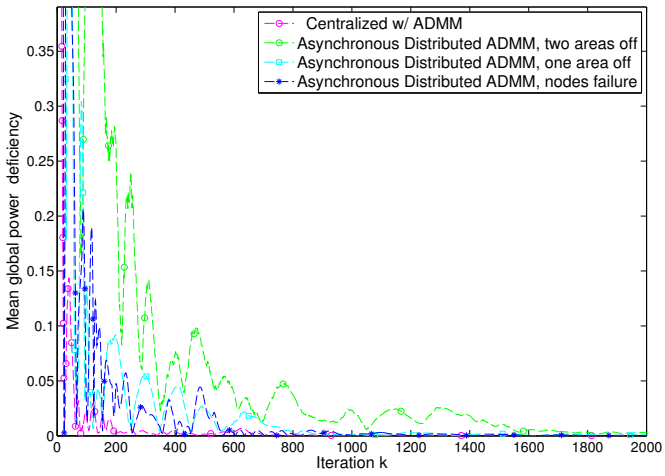
where $\Pi_{[p^g_{j,\min}, p^g_{j,\max}]}$ is the projection onto the interval $[p^g_{j,\min}, p^g_{j,\max}]$.

Firstly, we divide the network representing the IEEE−30 bus test system into $L = 3$ overlapping areas as per Table II. We compare the synchronous Distributed Production-Sharing algorithm with the asynchronous version in two different settings: i) one randomly chosen area is active at each iteration; and ii) areas A_1 and A_3 are activated together while area A_2 is activated randomly 50% of the time. In the first scenario only one area is activated per iteration. On total, each area is only activated for 33% of the total time. In the second scenario, A_1 and A_3 are activated together, while A_2 is activated 50% of the time.

Secondly, we test the application of the algorithm in the case of nodes failure. That is, the whole network except for random nodes is being updated every iteration. In this scenario, nodes $1-4, 27$ are randomly switched off for 45% of the time and nodes $10-14, 19, 22, 28-30$ are randomly switched off



(a) Normalized mean squared deviation



(b) Mean global power deficiency

Fig. 3. Performance of the asynchronous distributed Production-Sharing w/ ADMM algorithm when the IEEE−30 bus system is divided to 3 overlapping areas.

for 15% of the time. As an example, in a certain iteration when nodes $1-4$ are deactivated, the rest of the network is considered as one area that is currently activated. In another iteration, when nodes $28-30$ are deactivated, the rest of the network, represented by another area, is then activated.

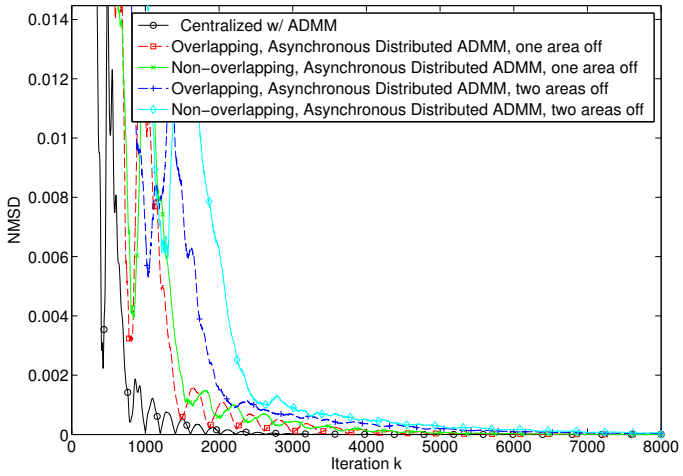
In Fig. 3a, we plot the normalized mean squared deviation (NMSD) to the optimal solution versus the number of iterations. Remark here that while every agent is active in the centralized scheme, only a subset of the agents may be active in the asynchronous version leading to a lower computations per iteration ratio. In Fig. 3b, we check the sufficiency of the power demand in this network by plotting the residual of the power flow constraints.

B. Non-overlapping areas

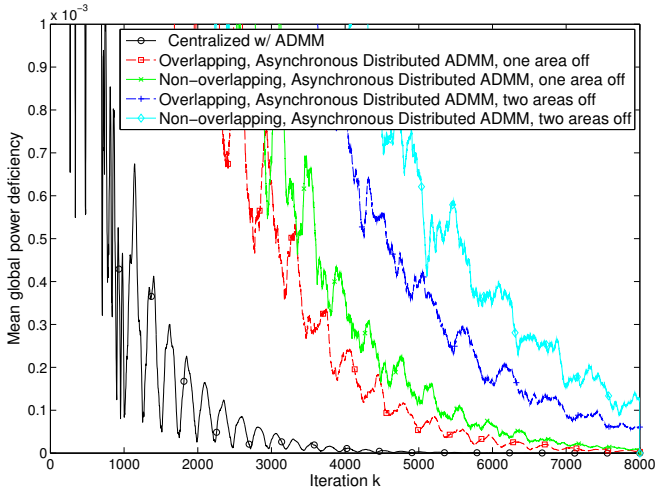
We take the same IEEE−30 bus system and we divide it into $L = 3$ *non-overlapping* areas as given by Table III. We modify this network by introducing 9 dummy nodes on the tie-lines linking two different areas. Using the two scenarios explained in Section VI-A, we plot in Fig. 4a and Fig. 4b the results of applying the Distributed Production-Sharing algorithm w/ADMM on this modified network. As observed from the plots, the convergence is slightly slower when the

TABLE III
IEEE−30 BUS TEST SYSTEM DIVISION INTO 3 NON-OVERLAPPING AREAS

Area	Nodes	Number of nodes
A_1	1,2,5-11,17,20,28	12
A_2	3,4,12-19	9
A_3	21-27,29,30	9



(a) Normalized mean squared deviation

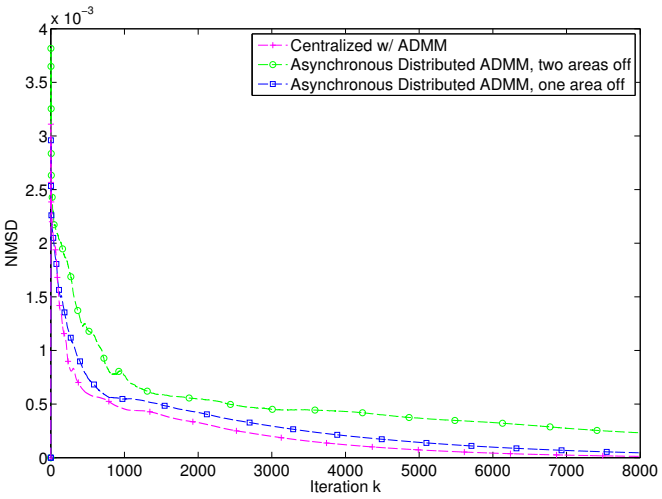


(b) Mean global power deficiency

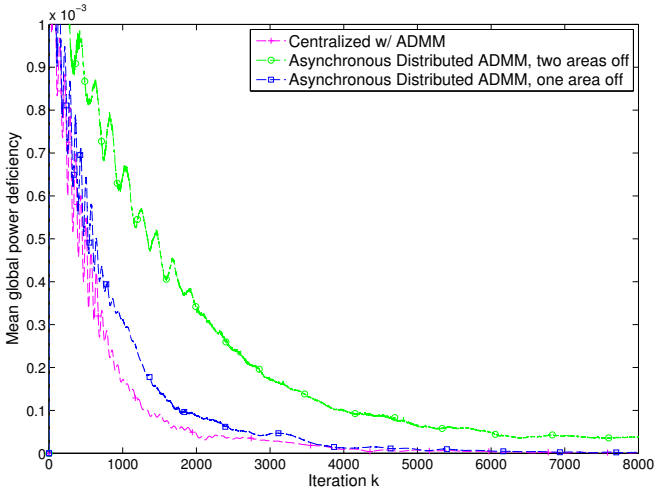
Fig. 4. Performance of the asynchronous distributed Production-Sharing w/ ADMM algorithm when the IEEE−30 bus system is divided to 3 overlapping and non-overlapping areas.

TABLE IV
IEEE−118 BUS TEST SYSTEM DIVISION INTO 3 OVERLAPPING AREAS

Area	Nodes	Number of nodes
A_1	1-34,38,113-115,117	39
A_2	24,33-75,116	45
A_3	68,69,75-112,118	41



(a) Normalized mean squared deviation



(b) Mean global power deficiency

Fig. 5. Performance of the asynchronous distributed ADMM algorithm when the network of 118 nodes is divided to 3 overlapping areas.

areas do not overlap. This is mainly because the variables of the shared nodes are being updated more frequently in the overlapping case. Thus, even in the case of a network with multiple authorities, an agreement on the minimum of the global cost can be met using the distributed Production-Sharing w/ ADMM algorithm.

C. IEEE−118 bus test system

We test our algorithms with larger areas using the IEEE−118 bus test system [32] as our network. This network consists of $N = 118$ buses, 53 generators and 186 branches. We divide it into 3 overlapping areas as in Table IV and we implement the two scenarios described in Section VI-A.

We obtain the plots in Fig. 5a and 5b representing the evolution of the NMSD and the global power demand deficiency (*i.e.*, the residual of the power flow constraints).

These plots illustrate the capability of our algorithms to solve the DC-OPF problem even when the network becomes larger. This feature makes them adequate for large power grid problems.

VII. Conclusion

In this paper, we provided distributed algorithms for solving a Production-Sharing Problem. First, a synchronous version based on the ADMM is developed. Then, an asynchronous version is derived; this scheme reduces the number of active agents to a random subset of the original network, allowing for random failures or for deactivation of parts of the network while preserving convergence.

We applied these algorithms to the DC-OPF problem, which can be cast as a Production-Sharing Problem. Simulations were carried on the IEEE−30 and IEEE−118 bus test systems. Multiple scenarios were provided, illustrating the convergence to the optimal network global state despite the presence of random failures and/or inactivated areas.

An interesting topic for future work is to study the asynchronous behavior of the algorithm when the network under study is dynamic. This is the main case when the nodes are equipped with renewable energy sources and storage devices. As for the asynchronous behavior of the algorithm, it would be interesting to study the case where the random coordinate selection sequence is Markovian and not i.i.d. so to model more realistic long-term failures.

Appendix A
Derivation of the synchronous algorithm

For every node $j \in V$, let $I_j = \{i : \boldsymbol{A}_{ip} \neq 0, p = \{j, j+N, j+2N\}\}$ be the set of constraints in which node j is involved.

The minimization step (9a) can be written as

$$\begin{aligned}\boldsymbol{u}^{k+1} = \underset{\substack{x_j,y_j,z_j,\\ j=1,\dots,N}}{\operatorname{argmin}} \sum_{j=1}^{N}\Bigg\{ f_j(x_j) + \sum_{i=1}^{2N}\Bigg\{ \frac{\rho}{2}\left\| A_{ij}x_j + \frac{\lambda_\alpha^k}{\rho} - v_\alpha^k \right\|^2 \\ + \frac{\rho}{2}\left\| A_{ij_{+N}}y_j + \frac{\lambda_{\alpha+N}^k}{\rho} - v_{\alpha+N}^k \right\|^2 \\ + \frac{\rho}{2}\left\| A_{ij_{+2N}}z_j + \frac{\lambda_{\alpha+2N}^k}{\rho} - v_{\alpha+2N}^k \right\|^2 \Bigg\}\Bigg\} \end{aligned} \tag{15}$$

where $\alpha = \alpha(i,j) = 3N(i-1) + j$ is used as a compact notation to index the values of $\boldsymbol{v}$ and $\boldsymbol{\lambda}$ corresponding to the element A_{ij} (similarly $\alpha+N$ and $\alpha+2N$ correspond to $A_{ij_{+N}}$ and $A_{ij_{+2N}}$).

This expression is separable into node basis because the component corresponding to the update of x_j (respectively, y_j and z_j), depends only on the previous values calculated for node j. Thus the iteration on $\boldsymbol{u}$ can be implemented in a distributed manner where each node $j \in V$ update its three components x_j, y_j and z_j by solving the following update steps.

$$x_j^{k+1} = \underset{x_j}{\operatorname{argmin}}\ f_j(x_j) + \frac{\rho}{2}\sum_{i=1}^{2N}\| \boldsymbol{A}_{ij}x_j + \frac{\lambda_\alpha^k}{\rho} - v_\alpha^k\|^2 \tag{16a}$$

$$y_j^{k+1} = \underset{y_j}{\operatorname{argmin}}\ \frac{\rho}{2}\sum_{i=1}^{2N}\| \boldsymbol{A}_{ij_{+N}}y_j + \frac{\lambda_{\alpha+N}^k}{\rho} - v_{\alpha_{+N}}^k\|^2 \tag{16b}$$

$$z_j^{k+1} = \underset{z_j}{\operatorname{argmin}}\ \frac{\rho}{2}\sum_{i=1}^{2N}\| \boldsymbol{A}_{ij_{+2N}}z_j + \frac{\lambda_{\alpha+2N}^k}{\rho} - v_{\alpha_{+2N}}^k\|^2. \tag{16c}$$

These update steps can be further simplified as follows using the specific structure of matrix $\boldsymbol{A}$ and using compact notations $\beta = \beta(i,j) = 3N(j-1)+j$, $\gamma = \gamma(i,j) = 3N(j+N-1)+j$ and $\omega = \omega(i,j) = 3N(i+N-1)+j$.

$$\begin{aligned} x_j^{k+1} &= \underset{x_j}{\text{argmin}}\ f_j(x_j) + \frac{\rho}{2}\sum_{i=1}^{N}\|\boldsymbol{A}_{ij}x_j + \frac{\lambda_\alpha^k}{\rho} - v_\alpha^k\|^2 \\ &\qquad + \frac{\rho}{2}\sum_{i=N+1}^{2N}\|\frac{\lambda_\alpha^k}{\rho} - v_\alpha^k\|^2 \\ &= \underset{x_j}{\text{argmin}}\ f_j(x_j) + \frac{\rho}{2}\|x_j + \frac{\lambda_\beta^k}{\rho} - v_\beta^k\|^2 \qquad (17) \\ y_j^{k+1} &= \underset{y_j}{\text{argmin}}\ \frac{\rho}{2}\sum_{i=1}^{N}\|\boldsymbol{A}_{ij_{+N}}y_j + \frac{\lambda_{\alpha_{+N}}^k}{\rho} - v_{\alpha_{+N}}^k\|^2 + \\ &\qquad \frac{\rho}{2}\sum_{i=N+1}^{2N}\|\boldsymbol{A}_{ij_{+N}}y_j + \frac{\lambda_{\alpha_{+N}}^k}{\rho} - v_{\alpha_{+N}}^k\|^2 \\ &= \underset{y_j}{\text{argmin}}\ \frac{\rho}{2}\Big\{\|-y_j + \frac{\lambda_{\beta_{+N}}^k}{\rho} - v_{\beta_{+N}}^k\|^2 + \\ &\qquad \|-y_j + \frac{\lambda_{\gamma_{+N}}^k}{\rho} - v_{\gamma_{+N}}^k\|^2\Big\} \qquad (18) \\ z_j^{k+1} &= \underset{z_j}{\text{argmin}}\ \frac{\rho}{2}\sum_{i=1}^{2N}\|\boldsymbol{A}_{ij_{+2N}}z_j + \frac{\lambda_{\alpha_{+2N}}^k}{\rho} - v_{\alpha_{+2N}}^k\|^2 \\ &= \underset{z_j}{\text{argmin}}\ \frac{\rho}{2}\sum_{i=1}^{N}\|L_{ij}z_j + \frac{\lambda_{\omega_{+2N}}^k}{\rho} - v_{\omega_{+2N}}^k\|^2 \qquad (19) \end{aligned}$$

To further simplify equations (17) to (19), we first explicit the update step (9b).

$$\begin{aligned} \boldsymbol{v}^{k+1} = \underset{\boldsymbol{v}}{\text{argmin}}\ G(\boldsymbol{v}) + \frac{\rho}{2}\sum_{i=1}^{2N}\sum_{j=1}^{N}\Bigg\{ &\left\|A_{ij}x_j^{k+1} - v_\alpha + \frac{\lambda_\alpha{}^k}{\rho}\right\|^2 \\ &+ \left\|A_{ij_{+N}}y_j^{k+1} - v_{\alpha_{+N}} + \frac{\lambda_{\alpha_{+N}}{}^k}{\rho}\right\|^2 \\ &+ \left\|A_{ij_{+2N}}z_j^{k+1} - v_{\alpha_{+2N}} + \frac{\lambda_{\alpha_{+2N}}{}^k}{\rho}\right\|^2\Bigg\}. \qquad (20) \end{aligned}$$

This last equation can be divided into a set of three equations

$$\begin{aligned} v_\alpha^{k+1} &= \underset{v_\alpha}{\text{argmin}}\ \frac{\rho}{2}\left\|A_{ij}x_j^{k+1} - v_\alpha + \frac{\lambda_\alpha{}^k}{\rho}\right\|^2 \qquad (21a) \\ v_{\alpha+N}^{k+1} &= \underset{v_{\alpha+N}}{\text{argmin}}\ \frac{\rho}{2}\left\|A_{ij_{+N}}y_j^{k+1} - v_{\alpha_{+N}} + \frac{\lambda_{\alpha_{+N}}{}^k}{\rho}\right\|^2 \qquad (21b) \\ v_{\alpha+2N}^{k+1} &= \underset{v_{\alpha+2N}}{\text{argmin}}\ \frac{\rho}{2}\left\|A_{ij_{+2N}}z_j^{k+1} - v_{\alpha_{+2N}} + \frac{\lambda_{\alpha_{+2N}}{}^k}{\rho}\right\|^2 \qquad (21c) \\ &\text{subject to } \sum_{j=1}^{N} v_\alpha + v_{\alpha_{+N}} + v_{\alpha_{+2N}} = \tilde{d}_i, \forall i = 1,\dots,2N. \end{aligned}$$

In order to solve this set of minimization steps we introduce the vector of Lagrangian multipliers $\boldsymbol{\pi} = (\pi_1,\dots,\pi_{2N})^T$ to the set of constraints on the entries of $\boldsymbol{v}$ (*i.e.*, to the set of constraints $\tilde{\boldsymbol{A}}\boldsymbol{u} = \tilde{\boldsymbol{d}}$). After some algebra we obtain,

$$\begin{aligned} v_\alpha^{k+1} &= \frac{1}{\rho}(-\pi_i^{k+1} + \lambda_\alpha^k) + A_{ij}x_j^{k+1} \qquad (22a) \\ v_{\alpha+N}^{k+1} &= \frac{1}{\rho}(-\pi_i^{k+1} + \lambda_{\alpha+N}^k) + A_{ij_{+N}}y_j^{k+1} \qquad (22b) \\ v_{\alpha_{+2N}}^{k+1} &= \frac{1}{\rho}(-\pi_i^{k+1} + \lambda_{\alpha_{+2N}}^k) + A_{ij_{+2N}}z_j^{k+1}. \qquad (22c) \end{aligned}$$

We substitute these results into their corresponding constraints which gives us

$$\pi_i^{k+1} = \frac{\rho}{d(i)}r_i(\boldsymbol{u}^{k+1}) + \frac{1}{d(i)}\sum_{j=1}^{N}\lambda_\alpha^k + \lambda_{\alpha_{+N}}^k + \lambda_{\alpha_{+2N}}^k \qquad (23)$$

where $d(i) = 2 + \mathbf{1}_{i>N}\sum_{j'=1}^{N}\mathbf{1}_{j'\in\Omega_i}$ is the degree of the i^{th} constraint (*i.e.*, the count of its nonzero elements), and $r_i^k \hat{=} [\boldsymbol{A}]_i\boldsymbol{u}^k - \tilde{d}_i$ is its residual given by

$$r_i^k = \mathbf{1}_{i<N}\sum_{j=1}^{N}\mathbf{1}_{i=j}(x_j^k - y_j^k) - d_i - \mathbf{1}_{i>N}\sum_{j=1}^{N}\mathbf{1}_{i=j_{+N}}y_j^k + \sum_{j=1}^{N}L_{i_{-N}j}z_j^k$$

where $\mathbf{1}_{i<N} = 1$ when $i<j$ and 0 otherwise, vice versa for $\mathbf{1}_{i>N}$.

As for the dual variable update step (9c), we can substitute it with a set of separated updates of its components λ_α. Then, by making use of (22a) we can reduce the computational complexity of the dual update step (9c) by replacing it with a set of updates on π. This is proved as follows,

$$\begin{aligned} \boldsymbol{\lambda}^{k+1} &= \boldsymbol{\lambda}^k + \rho\left(\tilde{\mathbf{A}}\boldsymbol{u}^{k+1} - \boldsymbol{v}^{k+1}\right) \\ \Rightarrow \lambda_\alpha^{k+1} &= \lambda_\alpha^k + \rho(A_{ij}x_j^{k+1} - v_\alpha^{k+1}) \qquad (24a) \\ \lambda_{\alpha+N}^{k+1} &= \lambda_{\alpha+N}^k + \rho(A_{ij_{+N}}y_j^{k+1} - v_{\alpha+N}^{k+1}) \qquad (24b) \\ \lambda_{\alpha+2N}^{k+1} &= \lambda_{\alpha+2N}^k + \rho(A_{ij_{+2N}}z_j^{k+1} - v_{\alpha+2N}^{k+1}) \qquad (24c) \end{aligned}$$

Equation (24a) yields

$$\begin{aligned} \lambda_\alpha^{k+1} &= \lambda_\alpha^k + \rho(A_{ij}x_j^{k+1} - \frac{1}{\rho}(-\pi_i^{k+1} + \lambda_\alpha^k) - A_{ij}x_j^{k+1}) \\ &= \pi_i^{k+1}, \quad \forall j = 1,\dots,N. \qquad (25) \end{aligned}$$

In the same manner we can prove that $\lambda_{\alpha+N}^{k+1} = \pi_i^{k+1}$ and $\lambda_{\alpha+2N}^{k+1} = \pi_i^{k+1}$, $\forall j = 1,\dots,N$. We plug this last result into (22a) (22b) and (22c),

$$\begin{aligned} v_\alpha^{k+1} &= A_{ij}x_j^{k+1} - \frac{r_i^{k+1}}{d(i)} \qquad (26a) \\ v_{\alpha+N}^{k+1} &= A_{ij_{+N}}y_j^{k+1} - \frac{r_i^{k+1}}{d(i)} \qquad (26b) \\ v_{\alpha+2N}^{k+1} &= A_{ij_{+2N}}z_j^{k+1} - \frac{r_i^{k+1}}{d(i)}. \qquad (26c) \end{aligned}$$

Which can be reduced to

$$\begin{aligned} v_\alpha^{k+1} &= \mathbf{1}_{i=j}\,x_j^{k+1} - \frac{r_i^{k+1}}{d(i)} \qquad (27a) \\ v_{\alpha+N}^{k+1} &= -\mathbf{1}_{i=j||i-N=j}\,y_j^{k+1} - \frac{r_i^{k+1}}{d(i)} \qquad (27b) \\ v_{\alpha+2N}^{k+1} &= \mathbf{1}_{i>N}L_{i_{-N}j}\,z_j^{k+1} - \frac{r_i^{k+1}}{d(i)}. \qquad (27c) \end{aligned}$$

These last updates correspond to the difference between the updated primal variables and the corresponding constraints' residuals. Thus, they can be directly taken into account in the other steps of the distributed optimization algorithm.

As for (23), after some algebra we obtain

$$\pi_i^{k+1} = \frac{\rho}{d(i)} r_i^{k+1} + \pi_i^k. \tag{28}$$

We adopt a new formulation for the residual $\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{r}_2)$ where, for $i = 1, \dots, N$

$$r_{1,i}^k \hat{=} r_i^k = \sum_{j=1}^N \mathbf{1}_{i=j}(x_j^k - y_j^k) - d_i \tag{29}$$

$$r_{2,i}^k \hat{=} r_{i+N}^k = -\sum_{j=1}^N \mathbf{1}_{i=j} y_j^k + \sum_{j=1}^N L_{ij} z_j^k. \tag{30}$$

In the same manner, we divide $\boldsymbol{\pi}$ into $\boldsymbol{\pi}_1 \in \mathbb{R}^N$ and $\boldsymbol{\pi}_2 \in \mathbb{R}^N$, after some algebra we obtain

$$\pi_{1,i}^{k+1} \hat{=} \pi_i^{k+1} = \frac{\rho}{d_1(i)} r_{1,i}^{k+1} + \pi_{1,i}^k \tag{31}$$

$$\pi_{2,i}^{k+1} \hat{=} \pi_{i+N}^{k+1} = \frac{\rho}{d_2(i)} r_{2,i}^{k+1} + \pi_{2,i}^k \tag{32}$$

Finally, we can rewrite equations (17), (18) and (19) as

$$x_j^{k+1} = \underset{x_j}{\operatorname{argmin}} \left\{ f_j(x_j) + \frac{\rho}{2} \| x_j + \frac{\pi_{1,j}^k}{\rho} - x_j^k + \frac{1}{d_1(j)} (x_j^k - y_j^k - d_j) \|^2 \right\}$$

$$y_j^{k+1} = \underset{y_j}{\operatorname{argmin}} \Big\{ \| - y_j + \frac{\pi_{1,j}^k}{\rho} + y_j^k + \frac{1}{d_1(j)} r_{1,j}^k \|^2 + \| - y_j + \frac{\pi_{2,j}^k}{\rho} + y_j^k + \frac{1}{d_2(j)} r_{2,j}^k \|^2 \Big\}$$

$$z_j^{k+1} = \underset{z_j}{\operatorname{argmin}} \sum_{i=1}^N \| L_{ij} z_j - L_{ij} z_j^k + \frac{\pi_{2,i}^k}{\rho} + \frac{1}{d_2(i)} r_{2,i}^k \|^2.$$

Solving the update steps on y_j and z_j leads to the Distributed Production-Sharing optimization with ADMM listed in Section III.

Appendix B
Non-overlapping to overlapping architecture

In the original problem, the constraint related to $\mathbf{L}$ writes for all $j \in V$, $y_j = \sum_{i \in V} L_{ji} z_i$. Furthermore, if j belongs to area ℓ, this can be rewritten $y_j = \sum_{i \in A_\ell} L_{ji} z_i + \sum_{k \in V \setminus A_\ell} L_{jk} z_k$ where the second term represents the coupling relations we want to eliminate thanks to the addition of dummy nodes.

In the new problem, the constraint related to $\mathring{\mathbf{L}}$ is written for all nodes $j \in A_\ell$ as $\mathring{y}_j = \sum_{i \in A_\ell \setminus j} \mathring{L}_{ji} \mathring{z}_i + \sum_{d \sim j} \mathring{L}_{jd} \mathring{z}_d + \mathring{L}_{jj} \mathring{z}_j$. For all $i, j \in V$ belonging to the same area, we have $\mathring{L}_{ji} = L_{ji}$.

Noticing that $\mathring{y}_d = 0$ for all $d \in \mathring{V}$, we get that $0 = \mathring{L}_{dd} \mathring{z}_d + \mathring{L}_{di} \mathring{z}_i + \mathring{L}_{dj} \mathring{z}_j = \mathring{L}_{dd} \mathring{z}_d + \mathring{L}_{id} \mathring{z}_i + \mathring{L}_{jd} \mathring{z}_j$, where i and j are the two original nodes between which the dummy node was inserted (and the second equality is due to the fact that $\mathbf{L}$ needs to remain symmetric). Node d serves as a dummy node which does not generate or consume, thus $\mathring{L}_{dj}(\mathring{z}_j - \mathring{z}_d) + \mathring{L}_{di}(\mathring{z}_i - \mathring{z}_d) = 0$. We assume the symmetry on the edges linking d to i and j, thus $\mathring{L}_{id} = \mathring{L}_{jd} = \mathring{L}_{di} = \mathring{L}_{dj}$. We obtain $\mathring{L}_{dd} = -2\mathring{L}_{dj}$. We can rewrite $\mathring{y}_j$ as $\mathring{y}_j = \sum_{k \in A_\ell \setminus j} \mathring{L}_{jk} \mathring{z}_k + (\mathring{L}_{jj} - \frac{1}{2} \sum_{d \sim j} \mathring{L}_{jd}) \mathring{z}_j - \frac{1}{2} \sum_{i \sim j, d \sim j} \mathring{L}_{dj} \mathring{z}_i$.

Comparing this expression of $\mathring{y}_j$ with the previous one, we obtain,

$$\mathring{L}_{jj} = L_{jj} + \frac{1}{2} \sum_{d \sim j} \mathring{L}_{jd}, \quad \mathring{L}_{jd} = -2L_{ij}, \quad \mathring{L}_{dd} = 4L_{ij}.$$

Acknowledgment

This work has been supported by the ERC Starting Grant 305123 MORE (Advanced Mathematical Tools for Complex Network Engineering). The work of F. Iutzeler is also supported by the IAP DYSCO Network (Dynamical Systems, Control, and Optimization).

References

[1] A. Abboud, R. Couillet, M. Debbah, and H. Siguerdidjane, "Asynchronous alternating direction method of multipliers applied to the direct-current optimal power flow problem," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[2] J. D. Glover, M. S. Sarma, and T. J. Overbye, *Power system analysis and design*. CengageBrain. com, 2011.

[3] G. Giannakis, V. Kekatos, N. Gatsis, S.-J. Kim, H. Zhu, and B. Wollenberg, "Monitoring and optimization for power grids: A signal processing perspective," *IEEE Signal Processing Magazine*, vol. 30, no. 5, pp. 107–128, 2013.

[4] S. Bolognani and S. Zampieri, "A distributed control strategy for reactive power compensation in smart microgrids," *IEEE Transactions on Automatic Control*, vol. 58, no. 11, pp. 2818–2833, 2013.

[5] A. Lam, A. Dominguez-Garcia, B. Zhang, and D. Tse, "Optimal distributed voltage regulation in power distribution networks," Tech. Rep., 2012.

[6] L. Gan and S. H. Low, "Optimal power flow in direct current networks," in *IEEE Conference on Decision and Control (CDC)*, 2013.

[7] M. B. Cain, R. P. Oneill, and A. Castillo, "History of optimal power flow and formulations," 2012.

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

[9] J. Sun and L. Tesfatsion, "DC optimal power flow formulation and solution using QuadProgJ," in *IEEE Power and Energy Society General Meeting*, 2007.

[10] B. Stott, J. Jardim, and O. Alsac, "DC power flow revisited," *IEEE Transactions on Power Systems*, vol. 24, no. 3, pp. 1290–1300, 2009.

[11] B. Stott, "Review of load-flow calculation methods," *Proceedings of the IEEE*, vol. 62, no. 7, pp. 916–929, 1974.

[12] J. M. Guerrero, P. C. Loh, T.-L. Lee, and M. Chandorkar, "Advanced control architectures for intelligent microgrids? Part II: Power quality, energy storage, and AC/DC microgrids," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 4, pp. 1263–1270, 2013.

[13] S. Massoud Amin and B. F. Wollenberg, "Toward a smart grid: power delivery for the 21st century," *IEEE Power and Energy Magazine*, vol. 3, no. 5, pp. 34–41, 2005.

[14] M. Kraning, E. Chu, J. Lavaei, and S. Boyd, "Dynamic network energy management via proximal message passing," *Foundations and Trends in Optimization*, vol. 1, no. 2, pp. 73–126, 2014.

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

[16] 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.

[17] J. N. Tsitsiklis, "Problems in decentralized decision making and computation." DTIC Document, Tech. Rep., 1984.

[18] J. N. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.

[19] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: convergence analysis and network scaling," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 592–606, 2012.

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

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

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

[23] 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.

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

[25] S. Paudyal, C. A. Canizares, and K. Bhattacharya, "Three-phase distribution opf in smart grids: Optimality versus computational burden," in *IEEE International Conference and Exhibition on Innovative Smart Grid Technologies (ISGT Europe)*, 2011.

[26] H. Kung, "Synchronized and asynchronous parallel algorithms for multiprocessors," DTIC Document, Tech. Rep., 1976.

[27] 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.

[28] 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.

[29] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, "Asynchronous Distributed Optimization using a Randomized Alternating Direction Method of Multipliers," *IEEE Conference on Decision and Control (CDC)*, 2013.

[30] A. Abboud, E. Bastug, K. Hamidouche, and M. Debbah, "Distributed caching in 5g networks: An alternating direction method of multipliers approach," *Online http://goo. gl/vBdhV7*, 2015.

[31] A. J. Conejo and J. A. Aguado, "Multi-area coordinated decentralized DC optimal power flow," *IEEE Transactions on Power Systems*, vol. 13, no. 4, pp. 1272–1278, 1998.

[32] IEEE power systems: http://www.ee.washington.edu/research/pstca.

[33] A. Gómez-Expósito, A. J. Conejo, and C. Cañizares, *Electric energy systems: analysis and operation*. CRC Press, 2008.

[34] L. Powell, *Power System Load Flow Analysis (Professional Engineering)*. McGraw-Hill Professional, 2004.

[35] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc wsns with noisy linkspart i: Distributed estimation of deterministic signals," *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 350–364, 2008.

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

[37] W. Deng, M.-J. Lai, and W. Yin, "On the o (1/k) convergence and parallelization of the alternating direction method of multipliers," *arXiv preprint arXiv:1312.3040*, 2013.

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

[39] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, "Optimal parameter selection for the alternating direction method of multipliers (admm): quadratic problems," *Automatic Control, IEEE Transactions on*, vol. 60, no. 3, pp. 644–658, 2015.

[40] P. Bianchi, W. Hachem, and F. Iutzeler, "A stochastic coordinate descent primal-dual algorithm and applications to large-scale composite optimization," *arXiv preprint arXiv:1407.0898*, 2014.

[41] E. Wei and A. Ozdaglar, "On the o (1= k) convergence of asynchronous distributed alternating direction method of multipliers," in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*. IEEE, 2013, pp. 551–554.