J Electr Eng Technol.2018; 13(5): 1778-1790
http://doi.org/10.5370/JEET.2018.13.5.1778

ISSN(Print) 1975-0102
ISSN(Online) 2093-7423

Fully Distributed Economic Dispatching Methods Based on Alternating Direction Multiplier Method

Linfeng Yang†, Tingting Zhang*, Guo Chen*, Zhenrong Zhang*,**, Jiangyao Luo* and Shanshan Pan§**

Abstract – Based on the requirements and characteristics of multi-zone autonomous decision-making in modern power system, fully distributed computing methods are needed to optimize the economic dispatch (ED) problem coordination of multi-regional power system on the basis of constructing decomposition and interaction mechanism. In this paper, four fully distributed methods based on alternating direction method of multipliers (ADMM) are used for solving the ED problem in distributed manner. By duplicating variables, the 2-block classical ADMM can be directly used to solve ED problem fully distributed. The second method is employing ADMM to solve the dual problem of ED in fully distributed manner. N-block methods based on ADMM including Alternating Direction Method with Gaussian back substitution (ADM_G) and Exchange ADMM (E_ADMM) are employed also. These two methods all can solve ED problem in distributed manner. However, the former one cannot be carried out in parallel. In this paper, four fully distributed methods solve the ED problem in distributed collaborative manner. And we also discussed the difference of four algorithms from the aspects of algorithm convergence, calculation speed and parameter change. Some simulation results are reported to test the performance of these distributed algorithms in serial and parallel.

Keywords: ADMM, Economic dispatch, Fully distributed, Dual, Parallel.

1. Introduction

The economic dispatch (ED) model of power system is the basic problem of the optimal operation of the power system. It means to optimize the output of each unit to minimize the total operating cost of the generator set under the condition of satisfying the system load demand and the running condition of the unit [1]. Most of the real-time optimal scheduling procedures adopted by the current national power companies are based on the classical ED mathematical model.

To date, a number of approaches have been proposed for ED. These methods mainly include two categories, the traditional classical optimization algorithm and the intelligent optimization algorithm. The traditional optimization algorithms include quadratic programming [2], linear programming [3], dynamic programming [4], Lagrangian relaxation [5, 6], and priority list [7]. The intelligent optimization algorithm mainly includes artificial neural network algorithm [8, 9], genetic algorithm [10], particle swarm optimization (PSO) [11], and some hybrid or improved method [12-14]. A hybrid algorithm ACO–ABC–HS [12] combines the framework of Ant Colony Optimization (ACO), Artificial Bee Colony (ABC) and Harmonic Search (HS) algorithms to find the optimized solution for the problem of ED for a multi-generator system. A novel modified particle swarm optimization [13], which includes advantages of bacterial foraging and PSO for constrained dynamic ED problem. The modified PSO has better balance between local and global search abilities and it can avoid local minima quickly. The integrated genetic algorithm [14] is implemented in both parallel and cluster structures.

With the development of renewable energy, the structure of power grid is more and more complex. Centralized ED is clearly not suitable for modern power grid. Most of the above algorithms only focus on how to quickly solve the problem from the perspective of computing speed. And they did not take into account the following questions: First, parallel computing should not mean a simple task of multi-machine allocation [15]. But it should be combined with the characteristics of the power system to achieve the distributed processing of resources and tasks [16]. Second, in the above parallel algorithms, at least one host needs to understand the entire network of mathematical models and detailed parameters, which needs to collect all the data of the whole system [14]. For this manner, data communication is the bottleneck and the participants' private parameters exposure is the mainly concerned problem. Finally, the development of the electricity market

† Corresponding Author: School of Computer Electronics and Information, Guangxi University, China. (ylf@gxu.edu.cn)
* School of Computer Electronics and Information, Guangxi University, China.
** Guangxi Key Laboratory of Multimedia Communication and Network Technology, China.
*** School of Electrical Engineering and Telecommunications, University of New South Wales, Australia.
§ College of Electrical Engineering, Guangxi University, China.
Received: November 10, 2017; Accepted: February 2, 2018

makes the models and data within the divisions of the grid a commercial secret, and the data exchange between regions becomes impossible and unnecessary [17, 18]. Based on the above reasons, the fully distributed parallel processing technology in the power system still is a problem worth a further study. In the fully distributed method, all participators can keep their device characteristics secret, and each agent solves a subproblem with limited information communication. The distributed method ADMM is very suitable for distributed convex optimization, and in particular to large-scale problems [19]. At present, many ADMM variations have been derived. Distributed computing of power system needs a kind of parallel algorithm which is suited to large power grid layered distribution control structure, quickly applied to the real-time control, conform to the development direction of power market. The ADMM can meet this requirement. ADMM for formulating and solving a decentralized unit commitment problem is presented in [20] which provide a significant benefit for computational speed. In [21], an optimal power flow scheme based on ADMM is proposed, which proves the expansibility and convergence of the algorithm.

Aiming at the characteristics of multi-zone autonomous decision-making in modern power system, based on the construction of decomposition and interaction mechanism, four kinds of distributed methods based on ADMM are employed to realize distributed cooperative ED of power system. These algorithms have the following three advantages: 1) Distributed ED methods have better confidentiality. 2) Distributed ED methods are more scalable and flexible. 3) Distributed ED methods are more robust than centralized ED when loss the certain local controllers. The major contributions are summarized as follows: 1) Four algorithms all decouple sub-problems for each unit. This means that the parameters of each unit can be kept in only one computing node. They are fully distributed to meet the needs of modern industrial privacy protection. 2) We use four fully distributed algorithms to solve ED problem. Among these methods, which are ADMM, E_ADMM, D_ADMM, and ADM_G, the first three methods can be implemented in master-slave distributed and parallel schema. And the master-node can be deployed in regional center; other computing nodes can be deployed (or installed) on each unit. Most computations for each unit can be done in each node in parallel. 3) The simulation results show that E_ADMM and D_ADMM can obtain high-quality solutions in reasonable times and D_ADMM algorithm has better parallel performance, ADMM has the best convergence for finding a normal quality solution, and this method is suited to solve large-scale ED problems when real-time solutions are needed.

2. Formulation for ED

The objective of the ED problem is to minimize the total operation cost:

$$min\ F = \sum_{i=1}^{N}\sum_{t=1}^{T}\left(a_i P_{i,t}^2 + b_i P_{i,t} + c_i\right) \tag{1}$$

where F is the total fuel cost of the units, N is the total number of units in the system, $P_{i,t}$ is the power output of unit i in period t, a_i, b_i, and c_i are the cost coefficients of unit i, respectively.

The constraints of the ED model are:

1) Power balance constraints: the sum of all units must be equal to the system requirements.

$$\sum_{i=1}^{N} P_{i,t} = P_{D,t} \tag{2}$$

where $P_{D,t}$ is the system load demand in period t.

2) Unit generation limits: the power output of each unit must be limited to one upper and lower.

$$P_i^{min} \le P_{i,t} \le P_i^{max} \tag{3}$$

where P_i^{min}, P_i^{max} are the maximum power output and minimum power output of unit i, respectively.

3) Ramp rate limits: the power output of a unit can't change by more than a certain value over a period of time.

$$P_{i,t} - P_{i,t-1} \le UR_i \tag{4}$$

$$P_{i,t-1} - P_{i,t} \le DR_i \tag{5}$$

where $P_{i,t}$ and $P_{i,t-1}$ are the power output of unit i in period t and $t-1$. UR_i, DR_i are the ramp up and down limits of unit i, respectively.

Then, the ED problem model is:

$$\min F = \sum_{i=1}^{N}\sum_{t=1}^{T}(a_i P_{i,t}^2 + b_i P_{i,t} + c_i)$$
$$\text{s.t.}\begin{cases} \sum_{i=1}^{N} P_{i,t} = P_{\mathrm{D},t} \\ P_i^{\min} \le P_{i,t} \le P_i^{\max} \\ P_{i,t} - P_{i,t-1} \le UR_i \\ P_{i,t-1} - P_{i,t} \le DR_i \end{cases} \tag{6}$$

3. Theoretical Basis of ADMM

ADMM can be traced back to the 1950s, and it was developed rapidly in the 1970s [19], which was first proposed by Gabay and Mercier. In recent years, due to fast processing performance and good convergence performance, the algorithm has attracted much attention in the field of large-scale data analysis and processing, such as statistical learning, speech recognition, image processing, etc.

ADMM is an important method to solve the convex optimization problem with separable structure. It is generally used to solve the convex optimization problem with equality constraints as follows:

$$\min f(x)+g(z) \quad s.t.\ Ax+Bz=c \tag{7}$$

Where $x \in R^n$, $z \in R^m$ are variables, $A \in R^{p\times n}$, $B \in R^{p\times m}$, $P_1^{k+1} := \tilde{P}_1^k$. The $f(\bullet)$ and $g(\bullet)$ are convex functions.

To solve the problem (7) by ADMM, we form the augmented Lagrangian

$$L_\rho(x,z,y)=f(x)+g(z)+y^T(Ax+Bz-c) +(\rho/2)\|Ax+Bz-c\|^2 \tag{8}$$

where y is the Lagrange multiplier, $\rho>0$ is called the penalty parameter, "$\|\bullet\|$" denotes "$\|\bullet\|_2$".

Then the unscaled form ADMM consists of the iterations as follows:

$$x^{k+1} := \underset{x}{argmin}\, L_\rho\left(x,z^k,y^k\right) \tag{9}$$

$$z^{k+1} := \underset{z}{argmin}\, L_\rho\left(x^{k+1},z,y^k\right) \tag{10}$$

$$y^{k+1} := y^k+\rho\left(Ax^{k+1}+Bz^{k+1}-c\right) \tag{11}$$

Let $u=(1/\rho)y$ which called as the scaled dual variable, then (9)-(11) can be equivalently expressed as a scaled form:

$$x^{k+1} := \underset{x}{argmin}\left(f(x)+(\rho/2)\left\|Ax+Bz^k-c+u^k\right\|^2\right) \tag{12}$$

$$z^{k+1} := \underset{z}{argmin}\left(g(z)+(\rho/2)\left\|Ax^{k+1}+Bz-c+u^k\right\|^2\right) \tag{13}$$

$$u^{k+1} := u^k+Ax^{k+1}+Bz^{k+1}-c \tag{14}$$

Under two assumptions, $L_{\rho=0}(x,z,y)$ has a saddle point, and $f(\bullet)$ and $g(\bullet)$ are closed, proper, and convex, the objective function of the iterates approaches the optimal value, i.e., $f(x^k)+g(z^k)\to p^*$, where p^* is the optimal value of (7).

4. Methodology for 4 Distributed ED Methods based on ADMM

For the sake of convenience, throughout the paper the following notations are used for the ED problem descripted in Section 2. $P_i=\left[P_{i,1};P_{i,2};\cdots;P_{i,T}\right]$, $P=[P_1;P_2;\cdots;P_N]$, $f(P)=\sum_{i=1}^N f_i(P_i)$, e_{ADMM}, $P_D=\left[P_{D,1},P_{D,2},\cdots,P_{D,T}\right]$. The constraints (3)(4)(5) can be fully decoupled according to the unit, and then define the set ρ_3, $i=1,\ldots,N$; $\chi_1=\{P\,|\,(3)(4)(5)\}$; Constraint (2) can be fully decoupled according to the time period, then define the set $\chi_2=\{P\,|\,(2)\}$.

We note that, with the above notations, all private parameters of each unit i are included in $f_i(P_i)$ and $\chi_{1,i}$. In order to solve the ED problem in distributed manner, our paper aims to decouple the ED to be each sub-problem corresponding to each unit i, and each sub-problem i only needs the data of $f_i(P_i)$ and $\chi_{1,i}$, i.e., each unit can keeping its information secret.

4.1 ADMM for fully distributed ED

To use the classic ADMM, we introduce the auxiliary variable z and the indicator function of set χ, defined as,

$$I_\chi(x)=\begin{cases}0 & if\ \ x\in\chi\\ +\infty & if\ \ x\notin\chi\end{cases} \tag{15}$$

Then, the objective function can be decomposed into two blocks with no overlapping variables by introducing indicator function, i.e., the model (6) can be rewritten in ADMM form as

$$\min f(P)+I_{\chi_1}(P)+I_{\chi_2}(z) \quad s.t.\ P-\mathrm{z}=0 \tag{16}$$

The iterations of the scaled form of ADMM for model (16) are the following:

$$P^{k+1} := \underset{P}{argmin}\left(f(P)+I_{\chi_1}(P)+(\rho_1/2)\left\|P-z^k+u^k\right\|^2\right) \tag{17}$$

$$z^{k+1} := \underset{z}{argmin}\left(I_{\chi_2}(z)+(\rho_1/2)\left\|P^{k+1}-z+u^k\right\|^2\right) \tag{18}$$

$$u^{k+1} := u^k+\left(P^{k+1}-z^{k+1}\right) \tag{19}$$

P-update (17) can be completely decoupled according to each unit i, which is equivalent to solving the following sub-problems:

$$\min f_i(P_i)+(\rho_1/2)\left\|P_i-z_i^k+u_i^k\right\|^2 \quad s.t.\ P_i\in\chi_{1,i} \tag{20}$$

It can be seen that the sub-problems (20) can be performed in parallel. Each sub-problem only requires the relevant parameters of the unit i. Each sub-problem is small in size and contains only *T* variables, i.e., P_i.

The *z*-update step (18) is equivalent to solving the following simple quadratic programming problem:

$$\min \frac{\rho_1}{2}\left\| z - P^{k+1} - u^k \right\|^2 \quad s.t. z \in \chi_2 \tag{21}$$

We are now in a position to give the classical ADMM for solving the problem of ED in full details.

Algorithm 1 ADMM for Distributed ED

Initialization: $z^0 = 0$, $u^0 = 0$, $M > 0$, $\rho_1 > 0$, $\varepsilon_1^{\text{pri}} > 0$, $\varepsilon_1^{\text{dual}} > 0$.
for $k = 0, \ldots, M$
  P-update:
    for each unit $i = 1, \ldots, N$: (in parallel)
      get P_i^{k+1} by solving problem (20).
    end
  z-update: get z^{k+1} by solving (21).
  u-update: (19)
  $r_1^k = P^k - z^k$,
  $s_1^k = \rho_1\left(z^k - z^{k+1}\right)$,
  if $r_1^k \le \varepsilon_1^{\text{pri}}$ and $s_1^k \le \varepsilon_1^{\text{dual}}$ **break**.
end
return $f(P)$ and P.

4.2 ADMM solve the ED dual problem

The constraint (2) in model (6) can be fully decoupled according to the time period. But in order to implement totally distributed ED solving, we must decoupled all constraints to be subproblems according to each unit.

We rewrite the model (6) as

$$\min \sum_{i=1}^{N} f_i(P_i)$$
$$s.t. \begin{cases} \sum_{i=1}^{N}(P_i - \frac{1}{N}P_D) = 0 \\ P_i \in \chi_{1,i} \qquad , \ i = 1, \cdots, N, \end{cases} \tag{22}$$

For solving the above separable convex programming problem (22), we use a decomposition algorithm presented in [22]. Here, we note that, according to the practical significance of problem (22), Slater condition holds for the problem (22). So, strong duality holds for (22).

Let $h_i(y) = \inf_{P_i \in \chi_{1,i}} \left\{ f_i(P_i) + \left\langle y, P_i - \frac{1}{N}P_D \right\rangle \right\}$, where y is Lagrangian multiplier vector associated with equality constraints, "$\langle \bullet, \bullet \rangle$" denotes the inner product. Then the Lagrangian dual function of (22) is

$$h(y) = \sum_{i=1}^{N} h_i(y) \tag{23}$$

Then, we obtain the dual problem of (22) as

$$\max h(y) \tag{24}$$

This problem can be rewritten to be a minimization problem with local variables z_i and a common global variable y:

$$min\left\{ -\sum_{i=1}^{N} h_i(z_i) \right\}$$
$$s.t.\ y - z_i = 0 \ , \quad i = 1, \cdots, N. \tag{25}$$

Partitioning the multiplier vector $p = (p_1, \cdots, p_N)$ and giving $\rho_2 > 0$, we may write the ADMM for (25) as follows:

$$y^{k+1} := \underset{y}{argmin}\left\{ < \sum_{i=1}^{N} p_i^k, y > + (\rho_2 / 2)\sum_{i=1}^{N}\left\| y - z_i^k \right\|^2 \right\} \tag{26}$$

$$z_i^{k+1} := \underset{z_i}{argmin}\left\{ -h_i(z_i) - \left\langle p_i^k, z_i \right\rangle + (\rho_2 / 2)\left\| y^{k+1} - z_i \right\|^2 \right\} \tag{27}$$

$$p_i^{k+1} := p_i^k + \rho_2\left(y^{k+1} - z_i^{k+1} \right) \tag{28}$$

The y-update problem (26) is minimizing a convex quadratic function of y, we can get the minimize y from the optimality condition, i.e.,

$$y^{k+1} = \frac{1}{N}\sum_{i=1}^{N} z_i^k - \frac{1}{N\rho_2}\sum_{i=1}^{N} p_i^k \tag{29}$$

By substituting the expression for $h_i(z_i)$ into z_i-update problem (27), we obtain that

$$\inf_{z_i} \{ -\inf_{P_i \in \chi_{1,i}} \left\{ f_i(P_i) + \left\langle z_i, P_i - \frac{1}{N}P_D \right\rangle \right\} - \left\langle p_i^k, z_i \right\rangle + (\rho_2 / 2)\left\| y^{k+1} - z_i \right\|^2 \}$$
$$\inf_{z_i} \sup_{P_i \in \chi_{1,i}} \left\{ -f_i(P_i) - \left\langle z_i, P_i - \frac{1}{N}P_D \right\rangle \right\} - \left\langle p_i^k, z_i \right\rangle + (\rho_2 / 2)\left\| y^{k+1} - z_i \right\|^2 \}$$
$$\sup_{P_i \in \chi_{1,i}} \inf_{z_i} \{ -f_i(P_i) - \left\langle z_i, p_i^k + P_i - \frac{1}{N}P_D \right\rangle + (\rho_2 / 2)\left\| y^{k+1} - z_i \right\|^2 \} \tag{30}$$

For any fixed P_i, the minimum on the right-hand side of (30) is uniquely attained by

$$z_i = y^{k+1} + \frac{1}{\rho_2}\left(p_i^k + P_i - \frac{1}{N}P_D \right) \tag{31}$$

We may thus substitute (31) into the function on the right-hand side of (30) to eliminate the variables z_i. As a result, we obtain

$$P_i^* := \arg\max_{P_i\in\chi_{1,i}} \{-f_i(P_i) - < y^{k+1} + \frac{1}{\rho_2}\left(p_i^k + P_i - \frac{1}{N}P_D\right),$$
$$p_i^k + P_i - \frac{1}{N}P_D > + (\rho_2/2)\left\|\frac{1}{\rho_2}\left(p_i^k + P_i - \frac{1}{N}P_D\right)\right\|^2\}$$
$$= argmin_{P_i\in\chi_{1,i}}\left\{f_i(P_i) + \frac{\rho_2}{2}\left(y^{k+1} + \frac{1}{\rho_2}\left(p_i^k + P_i - \frac{1}{N}P_D\right)\right)^2\right\} \tag{32}$$

Therefore, if a solution P_i^* of problem (32) is found, we can determine z_i by (31).

Note that, by the separability of problem (25), the updates (27) and (28) of variables $z = (z_1, \cdots, z_N)$ and $p = (p_1, \cdots, p_N)$ can be performed in parallel for $i = 1, \cdots, N$.

Algorithm 2 D_ADMM for Distributed ED

Initialization: $z_i^0 = 0$, $p_i^0 = 0$, $M > 0$, $\rho_2 > 0$, $\varepsilon_2^{\text{feasible}} > 0$.
for $k = 0, \ldots, M$
  (29)
  for each unit $i = 1, \ldots, N$:(in parallel)
    (32)
    $z_i^{k+1} = y^{k+1} + \frac{1}{\rho_2}\left(p_i^k + P_i^* - \frac{1}{N}P_D\right)$
    $p_i^{k+1} := p_i^k + \rho_2(y^{k+1} - z_i^{k+1})$,
  end
  $\varepsilon = \left\|y^{k+1} - y^k\right\|_\infty$
  if $\varepsilon < \varepsilon_2^{\text{feasible}}$ **break**.
end
return $f(P)$ and P.

4.3 Alternating direction method with gaussian back substitution

It can be seen that the model (6) is the linearly constrained separable convex minimization problem. And the objective function is decomposed into N blocks, then, the model (6) can be rewritten as:

$$\min \sum_{i=1}^N f_i(P_i)$$
$$s.t. \begin{cases} \sum_{i=1}^N P_i = P_D \\ P_i \in \chi_{1,i}, \qquad i = 1, \ldots, N \end{cases} \tag{33}$$

Inspired by the efficiency of ADMM, a natural idea for solving (33) is to directly extend the two blocks ADMM scheme descripted in section 3 for (33) with N blocks:

$$P_i^{k+1} := \arg\min_{p_i} \{f_i(P_i) + \frac{\rho_3}{2}\|(\sum_{j=1}^{i-1}P_j^k + P_i + \sum_{j=i+1}^N P_j^k - P_D) - \frac{1}{\rho_3}\lambda^k\|^2 \,|P_i \in \chi_{1,i}\}, i = 1, \cdots, N \tag{34}$$

$$\lambda^{k+1} = \lambda^k - \rho_3\left(\sum_{j=1}^N P_j^k - P_D\right) \tag{35}$$

However, [23] points out that the scheme (34)-(35) is not necessarily convergent if no further conditions are posed on the model (33), even though its efficiency has been verified empirically by some recent applications (see [24, 25]). Here, we introduce the Gaussian back substitution proposed in [26] to guarantee the convergence of ADMM with N blocks.

After each round of ADMM iteration, i.e. (34)-(35), let

$$\tilde{w}^k := \left(\tilde{P}_2^k; \ldots; \tilde{P}_N^k; \tilde{\lambda}^k\right) = \left(P_2^{k+1}; \ldots; P_N^{k+1}; \lambda^{k+1}\right) := w^{k+1}, \tag{36}$$

then the Gaussian back substitution step obtains new and corrected w^{k+1} by solving following equation,

$$H^{-1}M^T\left(w^{k+1} - w^k\right) = \sigma\left(\tilde{w}^k - w^k\right), \tag{37}$$

where $\sigma \in (0,1)$, and [26] recommend $\sigma \in (0.5,1)$ (or even more aggressively, σ=1);

$$M = \begin{bmatrix} \rho_3 D_2^T D_2 & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots \\ \rho_3 D_N^T D_2 & \ldots & \rho_3 D_N^T D_N & 0 \\ 0 & \ldots & 0 & \frac{1}{\rho_3}E^\lambda \end{bmatrix},$$

E^λ is identify matrix with dimension matching vector λ.

$$H = diag\left(\rho_3 D_2^T D_2, \ldots, \rho_3 D_N^T D_N, \frac{1}{\rho_3}E^\lambda\right),$$

D_i is the coefficient matrix of P_i in constraint (2), for our ED problem, D_i actually is an identity matrix, and then it is easy to verify that $H^{-1}M^T = \begin{bmatrix} E & \ldots & E & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \ldots & E & 0 \\ 0 & \ldots & 0 & E^\lambda \end{bmatrix}$ which is nearly a upper triangle block matrix. So the Gaussian back substitution step (37) can be solved by using back-substitution algorithm [27] just as shown in the following Algorithm 3.

Now, we have our ADM with Gaussian back substitution (ADM-G) for solving ED distributed. But, a disadvantage of ADM_G is that the blocks are updated one after another, which is not amenable for parallelization.

Algorithm 3 ADM_G for Distributed ED

Initialization: $P_i^0 := 0$, $\lambda^0 = 0$, $M > 0$, $\rho_3 > 0$, $f(P) := 0$, $\sigma \in (0,1)$, $\varepsilon_1^{\text{feasible}} > 0.$

for $k = 0,\ldots,M$

  ADMM step (prediction step)**:**

   for $i = 1,\ldots,N$:

    obtain $\tilde{P}_i^k$ by solving (34)

   end

   obtain $\tilde{\lambda}^k$ by using (35)

  Gaussian back substitution step (correction step):

   $\lambda^{k+1} := \lambda^k + \sigma\left(\tilde{\lambda}^k - \lambda^k\right)$

   $P_N^k := \lambda^k + \sigma\left(\tilde{P}_N{}^k - P_N^k\right)$

   for $i = N-1$, $N-2$, …, 2

    $P_i^{k+1} := P_i^k + \sigma\left\{\left(\tilde{P}_i^k - P_i^k\right) - \left(\tilde{P}_{i+1}{}^k - P_{i+1}^k\right)\right\}$

   end

   $P_1^{k+1} := \tilde{P}_1^k$

   $$\varepsilon = \max\left\{\frac{\max_i\left\{\left\|P_i^{k+1} - P_i^k\right\|_1\right\}}{\left\|P_i^1 - P_i^0\right\|_1}, \frac{\left\|\lambda^{k+1} - \lambda^k\right\|_1}{\left\|\lambda^1 - \lambda^0\right\|_1}\right\},$$

  if $\varepsilon \le \varepsilon_1^{\text{feasible}}$ **break**.

end

return $f(P)$nd P .

4.4 Exchange ADMM algorithm

The sequential nature of ADM_G motivates us to consider using a scheme that updates all the N blocks in parallel. By substituting $x_i = P_i - \frac{1}{N}P_D$ into problem (33), the constraints (3)(4)(5) become as follows:

$$P_i^{min} - \frac{1}{N}P_{D,t} \le x_{i,t} \le P_i^{max} - \frac{1}{N}P_{D,t}, \tag{38}$$

$$x_{i,t} - x_{i,t-1} \le UR_i, \tag{39}$$

$$x_{i,t-1} - x_{i,t} \le DR_i, \tag{40}$$

Similar to χ_1 and $\chi_{1,i}$, we define the set $\chi_{3,i} = \{x_i \mid (38)(39)(40)\}$, $i = 1,\ldots,N$ and $\chi_3 = \{x := (x_1,...,x_N) \mid (38)(39)(40)\}$.

Then we have the follow exchange problem:

$$min \sum_{i=1}^{N} g^i_{x_i \in \chi_{3,i}}(x_i)$$

$$s.t. \sum_i^N x_i = 0, i = 1,\ldots N, \tag{41}$$

where

$$g^i(x_i) = a_i x_i^2 + (b_i + 2a_i\frac{1}{N}P_D)x_i + a_i\left(\frac{1}{N}P_D\right)^2 + b_i\frac{1}{N}P_D + c_i.$$

The exchange problem can be solved via ADMM, the iterations of the scaled form of ADMM [19]:

$$x_i^{k+1} := \arg\min_{x_i}\{g^i(x_i) + \tfrac{\rho_4}{2}\| x_i - x_i^k + \overline{x}^k + u^k\|^2 \,|x_i \in \chi_{3,i}\} i = 1,\cdots,N \tag{42}$$

$$u^{k+1} := u^k + \overline{x}^{k+1}, \tag{43}$$

where $\overline{x} = \frac{1}{N}\sum_{i=1}^N x_i$ is the average of $x_1,\cdots,x_N$.

Here the x-update (42) can be carried out in parallel, with the subvectors x_i updated by N separate minimizations. The u-update gathers all the x_i^{k+1}, and broadcasts u^{k+1} back to the processors handing the x_i updates.

Algorithm 4 E_ADMM for Distributed ED

Initialization: $x_i^0 := 0$, $u^0 = 0$, $M > 0$, $\rho_4 > 0$, $\varepsilon^{\text{pri}} > 0$, $\varepsilon^{\text{dual}} > 0$.

for $k = 0,\ldots,M$

  for each unit $i = 1,\ldots,N$:(parallel)

   (42)

  end

  (43)

  $r_2^{k+1} = \left(x_1^{k+1} - \overline{x}^{k+1},\cdots,x_N^{k+1} - \overline{x}^{k+1}\right),$

  $s_2^k = -\rho_4\left(\overline{x}^k - \overline{x}^{k-1},\cdots,\overline{x}^k - \overline{x}^{k-1}\right).,$

  if $r_2^k \le \varepsilon_2^{\text{pri}}$ and $s_2^k \le \varepsilon_2^{\text{dual}}$ **break**.

end

$P^{k+1} = x^{k+1} + \frac{1}{N}P_{\text{D}},$

return $f(P)$ and P.

We note that, except ADM_G, the other three ADMMs can be extended and applied to the case that DC power flow network constraints being considered.

5. Numerical Results and Analysis

In order to facilitate the comparison of four kinds of fully distributed algorithms for solving the ED problem, in this section, we use eight kinds of unit characteristic data

Table 1. Number of units per problem case

No.	8 types of units								Total units
	1	2	3	4	5	6	7	8	
1	12	11	0	0	1	4	0	0	28
2	13	15	2	0	4	0	0	1	35
3	15	13	2	6	3	1	1	3	44
4	15	11	0	1	4	5	6	3	45
5	15	13	3	7	5	3	2	1	49
6	10	10	2	5	7	5	6	5	50
7	17	16	1	3	1	7	2	4	51
8	17	10	6	5	2	1	3	7	51
9	12	17	4	7	5	2	0	5	52
10	13	12	5	7	2	5	4	6	54
11	46	45	8	0	5	0	12	16	132
12	40	54	14	8	3		9	13	156
13	50	41	19	11	4	4	12	15	156
14	51	58	17	19	16	1	2	1	165
15	43	46	17	15	13	15	6	12	167

as the basic unit data, and the combination is 15 cases. The data of eight units and load demands can be found in [28] and [29].The specific combination is shown in Table 1, and the total number of time periods are $T = 24$. All the algorithms in this paper run the software environment are MATLAB R2014a, the computer processor for the Intel (R) Core (TM) i7-4790 CPU 3.60GHz, memory is 8.0GB. The quadratic programming problem (QP problem) is calculated using CPLEX12.6.2. Unless otherwise specified, we used the parameter values $M = 10^3$, $\sigma = 0.5$. $\varepsilon_1^{feasible} = 0.001$, $\varepsilon_2^{feasible} = 0.01$, $\varepsilon_1^{abs} = 10^{-3}$, $\varepsilon_1^{rel} = 10^{-2}$, $\varepsilon_1^{pri} = \sqrt{NT}\varepsilon_1^{abs} + \varepsilon_1^{rel} \max\left\{\left\|\left(P_1^{k+1};\ldots;P_N^{k+1}\right)\right\|,\left\|-z^{k+1}\right\|\right\}$, $\varepsilon_1^{dual} = \sqrt{NT}\varepsilon_1^{abs} + \varepsilon_1^{rel}\left\|\left(\rho_1 u_1^{k+1};\ldots;\rho_1 u_N^{k+1}\right)\right\|$, $\varepsilon_2^{abs} = 10^{-4}$, $\varepsilon_2^{rel} = 10^{-5}$, $\varepsilon_2^{pri} = \sqrt{NT}\varepsilon_2^{abs} + \varepsilon_2^{rel} \max\left\{\left\|\left(x_1^{k+1};\ldots;x_N^{k+1}\right)\right\|\right\}$, $\varepsilon_2^{dual} = \sqrt{NT}\varepsilon_2^{abs} + \varepsilon_2^{rel}\left\|\rho_4 u^{k+1}\right\|_2$.

5.1 Convergence analysis

The convergence of the ADM-G for solving multi-block problem is proved in [26]. The convergences of the other three fully distributed ED methods can be obtained according to the convergence of the classical ADMM descripted in Section 3 of this paper. The primal residual and the dual residual, which are the features of ADMM convergence, also converge to 0 (can be seen in the following simulation results). And the convergence of the classical ADMM can be found in [19, 30]. Because that $f(P)$ is strongly convex and the constraints of problem (6) are linear, the convergences of these fully distributed ED methods can be theoretically guaranteed also.

Now, we report some numerical results about convergence speed. In our simulations, these fully distributed algorithms have the best performances in convergence speed while solving our ED cases when ρ_1, ρ_3, ρ_4 are 0.125 and ρ_2 is 10.

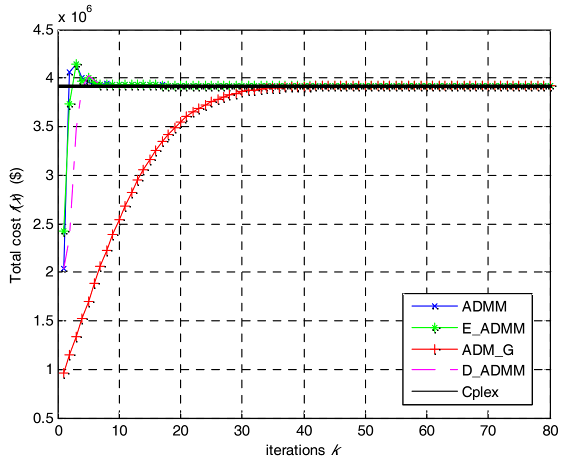


Fig. 1. Comparison of convergence speed for four methods

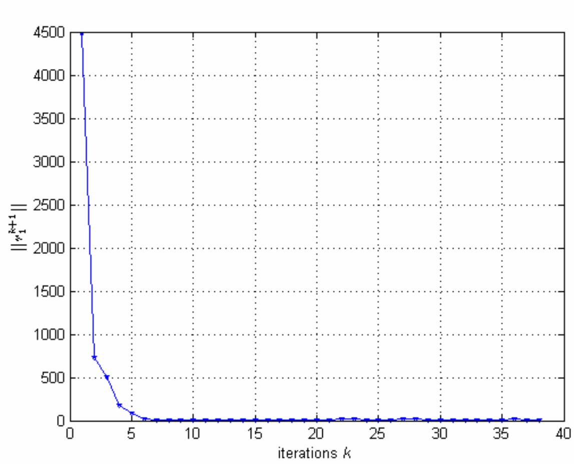


Fig. 2. ADMM primal residual values

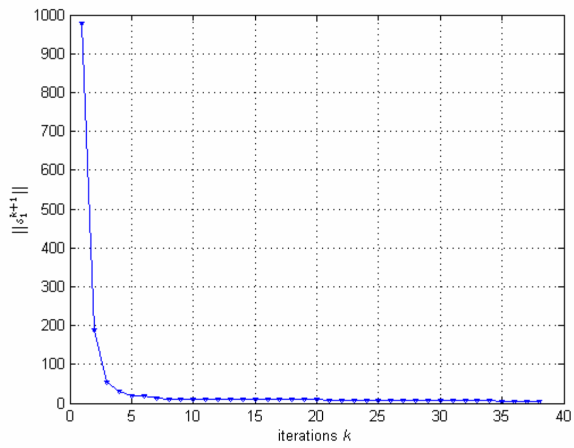


Fig. 3. ADMM dual residual values

Fig. 1 shows the results of four algorithms for solving the Case 1 in Table 1 while iteration number is $M = 80$. The black line in this Figure 1 labeled "Cplex" represents the final optimal value which obtained by using CPLEX to directly solve model (6) with default setting. As can be seen in this figure, after about 40 iterations, four methods obtain nearly equal optimal values. All these values are very close to the optimal values reported by CPLEX. And all these methods obtain nearly same solutions. However, their performances in convergence speed are different. The ADMM, E_ADMM, and D_ADMM have the similar performances and can reach the real optimal value after about 10 iterations. ADM_G is the slowest one, which need about 40 iterations.

Fig. 2 and Fig. 3 respectively report the primal residual $r_1^k = P^k - z^k$ and dual residual $s_1^k = \rho_1(z^k - z^{k+1})$ of the classic ADMM in the algorithm 1 while solving Case 1. It can be seen that these two values converge to 0 as the number of iterations increases. Fig. 1, 2, and 3 illustrate the convergence procedure of classical ADMM for solving Case 1.

5.2 Stability analysis

The stabilities of four fully distributed algorithms, which are based on ADMM, are influenced, more or less, by the setting of penalty parameter. In order to facilitate the comparison of the influence of penalty parameter variation on the stability of the algorithms, we solve the same test case with different methods by setting different penalty parameters.

Fig. 4 gives the comparison of different penalty parameters ρ_1 for algorithm 1 (ADMM) while solving Case 1. In this figure, the line labeled "Cplex" represents the optimal value obtained by CPLEX, which can be denoted as $f(P)_{Cplex}$. Furthermore, we denote the final objective value obtained by Algorithm 1 as $f(P)_{ADMM}$. Then we can denote $e_{ADMM} = \frac{f(P)_{ADMM} - f(P)_{Cplex}}{f(P)_{Cplex}}$ which represents the relative error of obtained final objective value for ADMM. Fig. 5 gives the comparison of different penalty parameters ρ_1 for algorithm 1 in the final objective values.

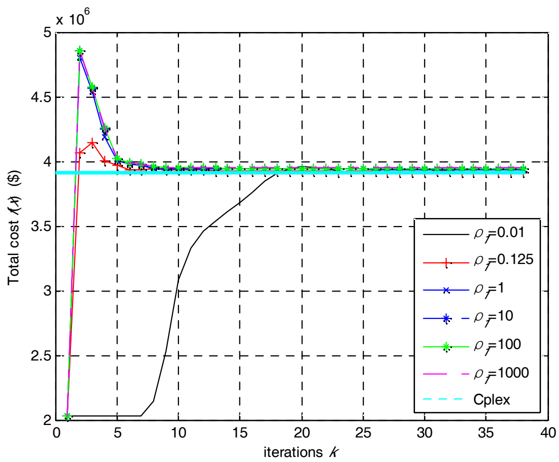


Fig. 4. Convergence comparison of ADMM with different ρ_1

As can be seen in Fig. 4, ADMM can converge with the same preset tolerances for all different parameters in our simulations, but the algorithm stability is different. See Fig. 4 and Fig. 5 simultaneously. Fig. 5 shows that when $\rho_1 = 0.01$, ADMM obtains the best final objective value which is the nearest to $f(P)_{Cplex}$, however, Fig. 4 shows that ADMM convergence very slowly for this $\rho_1 = 0.01$. When the penalty parameter is 0.125, the convergence rate is fastest and the percentage difference between the calculated value and the CPLEX calculation is about 0.02% which is a little more than the e_{ADMM} for $\rho_1 =$ 0.01 but is significantly less than the e_{ADMM} for $\rho_1 > 1$. At the same time, we know the sooner the constraint is satisfied as the penalty parameter increases. On the contrary, the more easily meet the objective function requirements. In addition, we do not give the detailed results for the other two methods, D_ADMM and E_ADMM because that, in our simulations, they have almost the similar numerical performances while setting different penalty parameters.

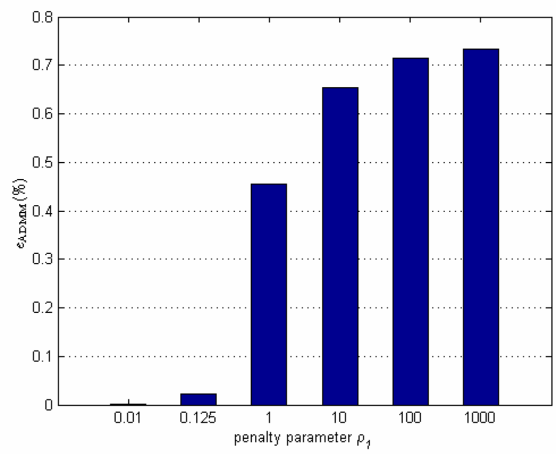


Fig. 5. Comparison of for ADMM with different ρ_1

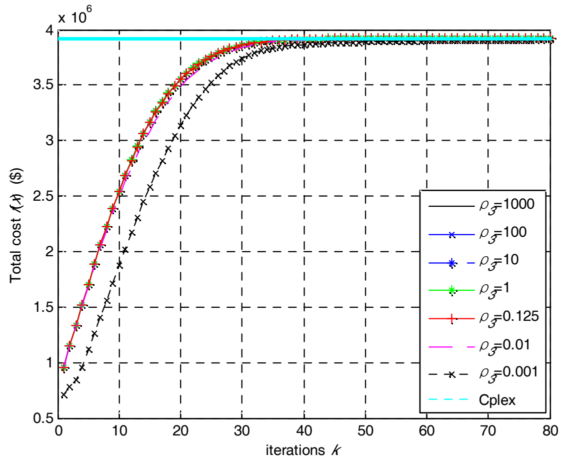


Fig. 6. Convergence comparison of ADM_G with different ρ_3

Fig. 6 gives the comparison of different ρ_3 for algorithm 3 (ADM_G) while solving Case 1. Fig. 7 gives the comparison of different penalty parameters ρ_3 for ADM_G in e_{ADM_G}, which can similarly defined as e_{ADMM}. It can be seen in Fig. 6, ADM_G can converge with the same preset tolerances for all different penalty parameters. But, different to ADMM, the objective value convergence procedures of ADM_G for different penalty parameters are very similar, and the convergence speed is a little slow for the smaller penalty parameters. Now, let's look at Fig. 7. It can be seen that ADM_G obtain the best objective value with smallest e_{ADM_G} when the penalty parameter is 0.125. When the penalty parameter is 0.01 or 0.001, the simulation shows that $e_{ADM\_G} < 0$, then we have $f(P)_{ADM\_G} < f(P)_{Cplex}$. Actually, AMD_G can push e_{ADM_G} to close to 0 by improving the tolerance $\varepsilon_1^{feasible}$.

5.3 Calculation speed comparison

We use four fully distributed algorithms and the solver

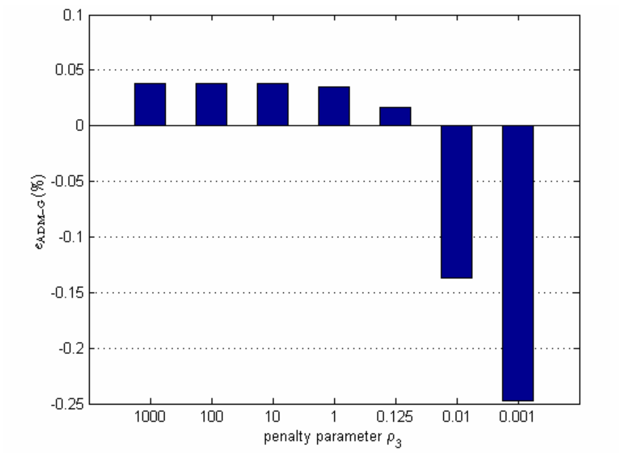


Fig. 7. The comparison of e_{ADM_G} for ADM_G with different ρ_3

of CPLEX to solve the 15 cases of Table 1. The simulation results are shown in Table 2. In this table, column "time" represents the execution time; "$e_X < 0$" represents relative error of objective value for method "X"; "K" reports the number of iterations.

As can be seen in Table 2, among four distributed methods, D_ADMM has the best performance in our simulations. D_ADMM consumes the second less average CPU times and achieves the best objective values (there are only 3 cases' "e_{D_ADMM}" that are 0.01%). E_ADMM consumes the third less average CPU times and achieves the second better objective values (there are 4 cases' "e_{E_ADMM}" that are 0.01%). ADMM consumes the least average times but achieves the significant worse objective values. As for ADM-G, this method uses the most times, but achieves the worst objective values. In Algorithm 3, we see that the algorithm have a correction step at each iteration. Through the correction, we ensure that the value of the variables change within a certain range after each iteration, so we can see that the objective function value of the ADM_G changes as shown in Fig. 1 and Fig. 6. These cause the convergence of the algorithm to slow down. In addition, we note that there is a higher e_{E_ADMM} for Case 6 using the ADMM algorithm, which indicates that the relative error between the result of the ADMM algorithm and the solver of Cplex is larger. The main reason for this case is that, the changed trends of the total cost $f(x)$ are similar for the 6th test case and the other test cases, but ADMM meets the requirement of accuracy with notable fewer iterations while solving the 6th test case.

5.4 Parallel performance comparison

As we discussed in Section 4 when we describing these distributed algorithms, ADMM, D_ADMM, and E_ADMM can be carried out in parallel. And all those steps which can be executed parallel have been marked in the descriptions of these algorithms.

Table 2. Comparison of four fully distributed methods in performances

No.	Cplex	ADMM				D_ADMM				ADM_G				E_ADMM			
	f(P)($)	*f(P)*($)	time(s)	e_{ADMM}	*K*	*f(P)*($)	time(*s*)	e_{D_ADMM}	*K*	*f(P)*($)	time(s)	e_{ADM_G}	*K*	*f(P)*($)	time(s)	e_{E_ADMM}	*K*
1	3918748	3919626	2.78	0.02%	38	3918749	3.34	0.00%	49	3920229	4.52	0.04%	64	3918742	3.95	0.00%	58
2	4930182	4932364	3.74	0.04%	40	4930257	4.10	0.00%	48	4934050	6.79	0.08%	75	4930238	5.24	0.00%	59
3	5333629	5336358	3.95	0.05%	35	5334089	4.82	0.01%	46	5344252	9.52	0.20%	86	5333887	6.45	0.00%	60
4	5043949	5042868	3.59	-0.02%	34	5044086	5.25	0.00%	52	5046557	9.42	0.05%	81	5043929	6.58	0.00%	64
5	5618227	5621163	4.14	0.05%	34	5618476	6.06	0.00%	51	5630697	10.76	0.22%	87	5618520	7.32	0.01%	62
6	4692242	4706561	1.89	0.31%	16	4692315	8.99	0.00%	81	4700552	11.06	0.18%	92	4692160	9.57	0.00%	83
7	6094415	6096244	4.47	0.03%	36	6094509	5.88	0.00%	49	6101022	11.55	0.11%	88	6094494	7.43	0.00%	62
8	5487161	5488912	3.85	0.03%	31	5487336	7.60	0.00%	63	5498977	11.81	0.22%	94	5487368	9.13	0.00%	76
9	5882288	5884095	3.90	0.03%	30	5882110	7.09	0.00%	58	5899410	11.64	0.29%	89	5882243	7.85	0.00%	63
10	5398276	5399975	4.10	0.03%	32	5398578	6.77	0.01%	55	5413215	12.94	0.28%	98	5398535	8.41	0.00%	67
11	16550901	16558734	12.36	0.05%	37	16551041	15.54	0.00%	50	16564824	67.05	0.08%	199	16551194	21.38	0.00%	55
12	18062626	18068295	12.79	0.03%	33	18063188	18.09	0.00%	50	18098600	92.92	0.20%	232	18065206	20.12	0.01%	40
13	17792083	17799154	13.34	0.04%	34	17792777	20.78	0.00%	57	17828363	93.06	0.20%	232	17793204	29.74	0.01%	63
14	20809395	20820371	15.61	0.05%	36	20810941	18.59	0.01%	46	20860861	105.66	0.25%	237	20810925	29.26	0.01%	55
15	18225919	18231444	13.16	0.03%	32	18226198	25.97	0.00%	67	18269991	102.38	0.24%	234	18225882	36.10	0.00%	73

Table 3. Results of parallel ADMM in a local computer

No.	1 worker	2 workers		4 workers	
	time(s)	time(s)	p_{spu}	time(s)	p_{spu}
1	2.78	2.29	1.21	1.99	1.4
2	3.74	2.76	1.36	2.4	1.56
3	3.95	2.83	1.4	2.48	1.59
4	3.59	2.69	1.33	2.3	1.56
5	4.14	2.87	1.44	2.46	1.68
6	1.89	1.35	1.4	1.15	1.64
7	4.47	3.11	1.44	2.56	1.75
8	3.85	2.69	1.43	2.24	1.72
9	3.9	2.7	1.44	2.23	1.75
10	4.1	2.95	1.39	2.42	1.69
11	12.36	8.3	1.49	6.09	2.03
12	12.79	8.52	1.5	6.52	1.96
13	13.34	8.81	1.51	6.53	2.04
14	15.61	10.24	1.52	7.52	2.08
15	13.16	8.8	1.5	6.51	2.02

Table 4. Results of parallel D_ADMM in a local computer

No.	1works	2works		4works	
	time(s)	time(s)	p_{spu}	time(s)	p_{spu}
1	3.34	2.79	1.2	2.39	1.4
2	4.1	3.06	1.34	2.6	1.58
3	4.82	3.47	1.39	2.81	1.72
4	5.25	3.96	1.33	3.21	1.64
5	6.06	4.19	1.45	3.31	1.83
6	8.99	6.46	1.39	5.24	1.72
7	5.88	4.01	1.47	3.19	1.84
8	7.6	5.09	1.49	4.17	1.82
9	7.09	4.8	1.48	3.88	1.83
10	6.77	4.57	1.48	3.76	1.8
11	15.54	8.96	1.73	6.73	2.31
12	18.09	10.05	1.8	7.7	2.35
13	20.78	11.49	1.81	8.71	2.39
14	18.59	10.26	1.81	7.76	2.4
15	25.97	14.66	1.77	10.83	2.4

Table 5. Results of parallel E_ADMM in a local computer

No.	1works	2works		4works	
	time(s)	time(s)	p_{spu}	time(s)	p_{spu}
1	3.95	3.18	1.24	2.95	1.34
2	5.24	3.77	1.39	3.5	1.5
3	6.45	4.45	1.45	3.99	1.62
4	6.58	4.7	1.4	4.16	1.58
5	7.32	5.17	1.42	4.45	1.64
6	9.57	6.87	1.39	5.7	1.68
7	7.43	5.17	1.44	4.61	1.61
8	9.13	6.36	1.44	5.48	1.67
9	7.85	5.31	1.48	4.56	1.72
10	8.41	5.89	1.43	5.06	1.66
11	21.38	15.25	1.4	11.98	1.78
12	20.12	14.76	1.36	12.28	1.64
13	29.74	20.72	1.44	16.35	1.82
14	29.26	21.24	1.38	16.75	1.75
15	36.10	26.88	1.34	20.31	1.78

Table 6. Results of parallel ADMM in distributed computers

No.	2 workers		4 workers	
	time(s)	p_{spu}	time(s)	p_{spu}
1	5.11	0.57	3.07	0.94
2	6.2	0.61	3.79	1
3	6.52	0.63	5.2	0.78
4	6.69	0.57	4.82	0.79
5	6.62	0.64	4.94	0.86
6	3.25	0.6	2.31	0.85
7	7.34	0.66	5.94	0.82
8	6.03	0.64	5.22	0.74
9	6.36	0.64	5.01	0.81
10	6.42	0.66	5.36	0.79
11	13.3	0.93	10.2	1.21
12	13.6	0.94	10.5	1.22
13	14.2	0.94	10.9	1.23
14	15.8	0.99	11.4	1.37
15	14	0.94	10.3	1.28

Table 7. Results of parallel D_ADMM in distributed computers

No.	2 workers		4 workers	
	time(s)	p_{spu}	time(s)	p_{spu}
1	4.92	0.68	3.46	0.97
2	5.53	0.84	3.78	1.23
3	6.59	0.77	4.21	1.2
4	7.69	0.74	5.45	1.04
5	7.91	0.78	5.85	1.06
6	12.8	0.74	9.01	1.05
7	7.74	0.78	5.7	1.06
8	10.2	0.77	7.34	1.07
9	9.59	0.83	7.12	1.12
10	9.1	0.77	6.79	1.04
11	15.9	1.02	11.4	1.43
12	18.4	1.08	13.5	1.47
13	21.1	1.03	15.5	1.41
14	18	1.08	12.9	1.5
15	26.1	1.05	18.9	1.45

Table 8. Results of parallel E_ADMM in distributed computers

No.	2 workers		4 workers	
	time(s)	p_{spu}	time(s)	p_{spu}
1	5.05	0.79	6.67	0.6
2	10.41	0.55	8.12	0.71
3	12.42	0.52	10.4	0.62
4	13.44	0.5	11.2	0.6
5	14.18	0.53	11.4	0.66
6	18.77	0.52	15.3	0.64
7	14.36	0.54	11.6	0.67
8	17.73	0.53	14.1	0.67
9	14.96	0.54	11.8	0.69
10	16.16	0.54	13.1	0.66
11	30.17	0.73	24.7	0.9
12	27.28	0.81	23.1	0.96
13	40.14	0.77	33.6	0.92
14	38.05	0.81	31.8	0.97
15	48.55	0.94	40.4	1.12

First, we implement the parallel versions for these three algorithms in Matlab with Parallel Computing Toolbox (PCT), and test ADMM, D_ADMM, and E_ADMM with 1 worker, 2 workers, and 4 workers in a local computer, respectively. And the results are listed in Table 3, 4, and 5. In these tables, column "time" represents the execution time; column " p_{spu} " represents the parallel speedup.

It can be seen that, with the increase of case scale, the speedups of all algorithms gradually increase, and the speedups with 4 workers are greater than the ones with 2 workers. At the same time, we can see that the D_ADMM has better parallel performance than the other two algorithms when 2 workers and 4 workers are used.

Next, we use Matlab Distributed Computing Engine (MDCE) services to achieve distributed cluster computing functions. Here, we connect two computers with the same configuration, by turning on one worker and two workers on each computer, respectively, to achieve fully distributed parallel computing. The statistics of three algorithms are listed in Table 6, Table 7, and Table 8.

By comparing with Tables 3, 4 and 5, it is found that using this distributed parallel configuration method will take more time, which is related to computer communication, network delay and so on, resulting in decreased performance.

6. Conclusion

In this paper, we solve the ED problem in four fully distributed manners based on ADMM. The separable objective function and physical constraints for each unit can be decoupled in four fully distributed methods. Except ADM_G, the other three fully distributed methods all can be carried out in master-slave distributed and parallel schema and can protect privacy for independent power producer. Simulation results show that the D_ADMM and the E_ADMM can obtain high-quality solutions in reasonable times, and the D_ADMM has the better parallel performance, the ADMM can meet the tolerance in the shortest possible times, which suit for solving large-scale ED problems.

Acknowledgements

This work was supported by the Natural Science Foundation of China(51767003, 51407037, 61661004, 61762007, and 61462006), the Guangxi Natural Science Foundation(2016GXNSFDA380019, 2014GXNSFBA1182 74, and 2014GXNSFAA118391), Guangxi Key Laboratory of Power System Optimization and Energy Technology Foundation (15-A-01-11).

References

[1] B.H. Chowdhury and S. Rahman, "A review of recent advances in economic dispatch," *IEEE Trans. Power Systems*, vol. 5, no. 1, pp. 1248-1259, Nov. 1990.

[2] G.F. Reid and L. Hasdorff, "Economic Dispatch Using Quadratic Programming," *IEEE Trans. PAS*, vol. PAS-92, no. 6, pp. 2015-2023, Nov. 1973.

[3] C.B. Somuah and N. Khunaizi, "Application of linear programming redispatch technique to dynamic generation allocation," *IEEE Trans. Power Systems,* vol. 5, no. 1, pp. 20-26, Feb. 1990.

[4] J.G. Waight, F. Albuyeh, and A. Bose, "Scheduling of Generation and Reserve Margin Using Dynamic and Linear Programming," *IEEE Trans. PAS*, vol. PAS-100, no. 5, pp. 2226-2230, May 1981.

[5] C. Wang and S. M. Shahidehpour, "Optimal generation scheduling with ramping costs," *IEEE Trans. Power Systems,* vol. 10, no.1, pp. 60-67, Feb. 1995.

[6] T. Guo, M.I. Henwood, and M.V. Ooijen, "An algorithm for combined heat and power economic dispatch," *IEEE Trans. Power Systems*, vol. 11, no. 4, pp. 1778-1784, Nov. 1996.

[7] R.R. Shoults, S.K. Chang, S. Helmick, and W.M. Grady, "A Practical Approach to Unit Commitment, Economic Dispatch, and Savings Allocation for Multiple-Area Pool Operation with Import/Export Constraints," *IEEE Trans. PAS*, vol. PAS-99, no. 2, pp. 625-635, Mar. 1980.

[8] L. Imen, B. Mouhamed, and L. Djamel, "Economic dispatch using classical methods and neural networks," *in Proc. of International Conference on Electrical and Electronics Engineering*, Turkey, Nov. 2013.

[9] A. Mohammadi, M.H. Varahram, and I. Kheirizad, "Online Solving of Economic Dispatch Problem Using Neural Network Approach and Comparing it with Classical Method," *in Proc. of International Conference on Emerging Technologies*, Pakistan, Nov. 2006.

[10] G.B. Sheble and K. Brittig, "Refined genetic algorithm-economic dispatch example," *IEEE Trans. Power Systems*, vol. 10, no. 1, pp. 117-124, Feb. 1995.

[11] Z. L. Gaing, "Particle swarm optimization to solving the economic dispatch considering the generator constraints," *IEEE Trans. Power Systems*, vol. 18, no. 3, pp. 1187-1195, Jul. 2003.

[12] T. Sen and H.D. Mathur, "A new approach to solve Economic Dispatch problem using a Hybrid ACO–ABC–HS optimization algorithm," *International Journal of Electrical Power & Energy Systems*, vol. 78, pp. 735-744, Jun. 2016.

[13] A.Y. Saber, "Economic dispatch using particle swarm optimization with bacterial foraging effect," *International Journal of Electrical Power & Energy Systems*, vol. 34, no. 1, pp. 38-46, Jan. 2012.

[14] C.C. Fung, S.Y. Chow, and K.P. Wong, "Solving the economic dispatch problem with an integrated parallel genetic algorithm," *in Proc. of International Conference on Power System Technology*, Australia, Dec. 2000.

[15] P. Subbaraj, R. Rengaraj, S. Salivahanan, and T.R. Senthilkumar, “Parallel particle swarm optimization with modified stochastic acceleration factors for solving large scale economic dispatch problem,” *International Journal of Electrical Power & Energy Systems*, vol. 32, no. 9, pp. 1014-1023, Nov. 2010.

[16] F. Guo, C. Wen, and L. Xing, “A distributed algorithm for economic dispatch in a large-scale power system,” *in Proc. of International Conference on Control, Automation, Robotics and Vision*, Nov. 2016.

[17] H. Yang, D. Yi, J. Zhao, and Z. Dong, “Distributed optimal dispatch of virtual power plant via limited communication,” *IEEE Trans. Power Systems*, vol. 28, no. 3, pp. 3511-3512, Mar. 2013.

[18] G. Chen, C. Li, and Z. Dong, “Parallel and Distributed Computation for Dynamical Economic Dispatch,” *IEEE Trans. Smart Grid*, vol. 8, no. 2, pp. 1026-1027, Mar. 2017.

[19] 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, 2010.

[20] M.J. Feizollahi, M. Costley, S. Ahmed, and S. Grijalva, “Large-scale decentralized unit commitment,” *International Journal of Electrical Power & Energy Systems*, vol. 73, pp. 97-106, Dec. 2015.

[21] S. Magnusson, P.C. Weeraddana, and C. Fischione, “A distributed approach for the optimal power flow problem based on ADMM and sequential convex approximations,” *IEEE Trans. on Control of Network Systems*, vol. 2, no. 3, pp. 238-257, Sept. 2015.

[22] M. Fukushima, “Application of the alternating direction method of multipliers to separable convex programming,” *Computational Optimization and Applications*, vol. 1, no. 1, pp. 93-111, Oct. 1992.

[23] C.H. Chen, B.S. He, Y.Y. Ye, and X.M. Yuan, “The direct extension of ADMM for multiblock convex minimization problems is not necessarily convergent,” *Math. Program.*, vol. 155, no. 1-2, pp. 57-79, Jan. 2016.

[24] Y.G. Peng, A. Ganesh, J. Wright, W.L. Xu, and Y. Ma, “Robust alignment by sparse and low-rank decomposition for linearly correlated images,” *IEEE Trans. PAMI*, vol. 34, no. 11, pp. 2233-2246, Nov. 2012.

[25] M. Tao and X.M. Yuan, “Recovering low-rank and sparse components of matrices from incomplete and noisy observations, *SIAM J. Optim.*, vol. 21, no. 1, pp. 57-81, Jan. 2011.

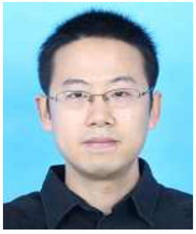
[26] B.S. He, M. Tao, and X.M. Yuan. “Alternating direction method with Gaussian Back substitution for separable convex programming,” *SIAM J. Optim.*, vol. 22, no. 2, pp. 313-340, Apr. 2012.

[27] J.H Mathews and K.K. Fink, *Numerical Methods Using Matlab(4th Edition)*, Prentice-Hall Inc., 2004.

[28] J. Ostrowski, M.F. Anjos, and A. Vannelli, “Tight mixed integer linear programming formulations for the unit commitment problem,” *IEEE Trans. Power Systems*, vol. 27, no. 1, pp. 39-46, Feb. 2012.

[29] L.F. Yang, C. Zhang, J.B. Jian, K. Meng, Y. Xu, and Z.Y. Dong, “A novel projected two-binary-variable formulation for unit commitment in power systems. *Applied Energy*, vol. 187, pp. 732-745, Feb. 2017.

[30] M. Nick, R. Cherkaoui, and M. Paolone, “Optimal siting and sizing of distributed energy storage systems via alternating direction method of multipliers,” *International Journal of Electrical Power & Energy Systems*, vol. 72, pp. 33-39, Nov. 2015.



Linfeng Yang received the M.S. degree from Shanghai University, Shanghai, China, in 2005, and the Ph.D. degree in electrical engineering from Guangxi University, Nanning, China, in 2012. Now he is an associate professor in Guangxi University. His research interests include parallel and distributed computing and optimization in power systems.

Tingting Zhang received the M.S. degree from Guangxi University in 2017. Her research interest is distributed computing.

Guo Chen received the Ph.D. degree in electrical engineering from the University of Queensland, Brisbane, Australia, in 2010. Now he is an Australian Research Council DECRA fellow and lecturer at School of Electrical Engineering and Telecommunications, the University of New South Wales. His research interests include optimization and control, complex network, dynamical systems, intelligent algorithms, and their applications in smart grid.

Zhenrong Zhang received the Ph.D degree in Nanyang Technological University, Singapore, in 2006. Now he is a professor in Guangxi University. His research interests include optimization of network and energy internet.

Jiangyao Luo is a postgraduate of Guangxi University. And his research interest is distributed computing.

Shanshan Pan receive the B.S. degree from Guangxi University in 2012. Now she is pursuing her Ph.D degree in Guangxi University. Her research interest is economic dispatch of power systems.