Distributed Optimal Power Flow With Discrete Control Variables of Large Distributed Power Systems

Ch'i-Hsin Lin and Shin-Yeu Lin

***Abstract*—In this paper, we propose a distributed algorithm to solve the yet explored distributed optimal power flow problem with discrete control variables of large distributed power systems. The proposed algorithm consists of two distinguished features: 1) a distributed algorithm for solving continuous distributed optimal power flow to serve as a core technique in the framework of ordinal optimization (OO) strategy, and 2) implementing the OO strategy in a distributed power system to select a good enough discrete control variable solution. We have tested the proposed algorithm for several cases on the IEEE 118-bus and Tai Power 244-bus systems using a 4-PC network. The test results demonstrate the validity, robustness, and excellent computational efficiency of the proposed distributed algorithm in getting a good enough feasible solution.**

***Index Terms*—Discrete control variables, distributed computation, distributed optimal power flow, nonlinear programming, ordinal optimization.**

NOMENCLATURE

Due to involved notations, in this section we illustrate those frequently appear throughout the paper, and the rest will be illustrated in the context.

N	Total number of subsystems.
x_i	Vector of continuous variables consisting of real and reactive power generations and bus complex voltage corresponding to subsystem i.
$x=(x_1,x_2,\ldots,x_N)$	Vector of continuous variables of the overall system.
$f_i(\cdot)$	Objective function of subsystem i.
$f(\cdot)=\sum_{i=1}^N f_i(\cdot)$	Objective function of the overall system.
J_i	Index set of subsystems connecting with subsystem i.
v_{jb}^i	Vector of complex voltage on the boundary buses of subsystem j, which connects with subsystem i.
$v_{J_ib}^i$	$=(v_{jb}^i)$, $j\in J_i$.
u_{d_i}	r_i–dimensional vector of discrete control variables, such as switching shunt capacitor banks and transformer taps, etc., corresponding to subsystem i.
$u_d=(u_{d_1},\ldots,u_{d_N})$	r–dimensional discrete control variables of the overall system and $r=r_1+\cdots+r_N$.
U_{d_i}	Solution space of u_{d_i} for subsystem i.
$U_d=\bigcup_{i=1}^N U_{d_i}$	Solution space of u_d.
$g_i(x_i,u_{d_i},v_{J_ib}^i)=0$	Real and reactive power balance equations of subsystem i.
$h_i(x_i)\le 0$	Inequality constraints in subsystem i, such as security limits on voltage magnitude, real power line flows and real and reactive power generation limits.
y_i	Vector of continuous variables in the general CDOPF shown in (3) corresponding to subsystem i.
y	$=(y_1,\ldots,y_N)$.
y_{jb}^i	Continuous variables on the boundary buses of subsystem j, which connects with subsystem i.
$y_{J_ib}^i$	$=(y_{jb}^i)$,$j\in J_i$.
$\bar{g}_i(y_i,y_{J_ib}^i)=0$	Equality constraints in the general CDOPF (3).
$\bar{g}_{ib}^j(y_i,y_{jb}^i)$	$\bar{g}_{ib}^j(y_i,y_{jb}^i)=0$ represents the equality constraints on the boundary buses of subsystem i but involving y_{jb}^i.
$\bar{g}_{ib}(y_i,y_{J_ib}^i)$	$=(\bar{g}_{ib}^j(y_i,y_{jb}^i))$,$j\in J_i$.
$(\overset{0}{\bar{g}}_i(y_i)$,$\bar{g}_{ib}(y_i,y_{J_ib}^i))$	Partition of $\bar{g}_i(y_i,y_{J_ib}^i)$.

Manuscript received November 26, 2007; revised February 10, 2008. This work was supported in part by the National Science Council, Taiwan, R.O.C., under Grant NSC95-2221-E-009-099-MY2. Paper no. TPWRS-00865-2007.

C.-H. Lin is with the Department of Electronics Engineering, Kao Yuan University, Kaoshiung, Taiwan 821, R.O.C. (e-mail: chsinlin@cc.kyu.edu.tw).

S.-Y. Lin is with the Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mail: sylin@cc.nctu.edu.tw).

Digital Object Identifier 10.1109/TPWRS.2008.926695

$\bar{h}_i(y_i) \leq 0$	Inequality constraints in the general CDOPF (3).
u_{c_i}	Continuous version of u_{d_i} corresponding to subsystem i.
u_c	$= (u_{c1}, u_{c2}, \ldots, u_{cN})$.
$\square^*$	Optimal $\square$.

I. Introduction

Although the optimal power flow (OPF) problem has a long history in power system research [1]–[5], the study of distributed OPF is introduced only recently. Kim and Baldick proposed a course-grained distributed OPF algorithm in [6], and they also compared three decomposition coordination methods for implementing distributed OPF algorithms in [7]. Hur *et al.* had evaluated the convergence rate of the auxiliary problem principle for a distributed OPF algorithm in [8]. In a more recent paper [9], Hur *et al.* considered the security limits for distributed OPF. Furthermore, Nogales *et al.* proposed a decomposition algorithm for the multiarea OPF problem in [10]. Chang and Lin proposed an MPBSG technique based parallel dual type method for solving distributed OPF problems in [11], and similar technique appeared in [12]. These excellent research works had made distributed OPF possible; however, issues of handling *discrete control variables* such as the switching shunt capacitor banks and transformer taps in a large *distributed* power system are not explored in the above mentioned papers.

Discrete control variables play an important role in centralized OPF and have been studied for years [13]–[16] including more recent papers that use ordinal optimization (OO) approach [17], simulated annealing (SA) method [18], genetic algorithm (GA) [19], tabu search (TS) method, [20], and evolutionary programming (EP) [21] as the solution techniques. The importance of discrete control variables to distributed OPF remains as well. Thus, **d**istributed **o**ptimal **p**ower **f**low with **d**iscrete control variables, which is abbreviated as **DOPFD** in this paper, of large distributed power system is an important research topic to pursue. DOPFD is a large dimension *distributed combinatorial optimization* problem, which, in general, is computationally intractable. Thus, the *purpose* of this paper is to propose a computationally efficient *distributed algorithm* to solve the DOPFD for a *good enough* solution. The proposed distributed algorithm for DOPFD possesses two distinguished features: 1) a distributed algorithm for solving continuous distributed OPF to serve as a core technique in the framework of OO strategy, and 2) implementing the OO strategy [22], [23] in a distributed power system to select a good enough discrete control variable solution.

In addition, we will implement the proposed distributed algorithm in a real PC network to demonstrate its validity. Thus, the *contribution* of this paper is that we not only propose a computationally efficient distributed algorithm to solve the DOPFD for a good enough solution but also implement it in a real computer network.

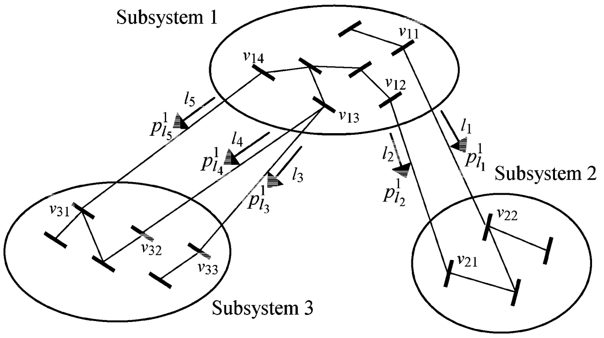


Fig. 1. Example system formed by three interconnected subsystems.

This paper is organized in the following manner. In Section II, we will state the considered DOPFD mathematically. In Section III, we will present the proposed distributed algorithm to solve the DOPFD for a good enough solution. In Section IV, we will test the proposed distributed algorithm on the IEEE 118-bus and Tai Power (TP) 244-bus systems, which are arbitrarily partitioned into four subsystems, using a PC network. Finally, we will draw a conclusion in Section V.

II. Problem Statement

The considered DOPFD problem can be stated in the following:

$$\begin{aligned} &\min f(x)\left(=\sum_{i=1}^{N} f_i(x_i)\right) \\ \text{subject to}\quad & g_i(x_i, v^i_{J_ib}, u_{d_i}) = 0 \\ & h_i(x_i) \leq 0 \\ & u_{d_i} \in U_{d_i},\ i = 1, \ldots, N \\ & \underline{p}_l \leq p^i_l(v^j_{ib}, v^i_{jb}) \leq \bar{p}_l \\ & \forall l \in T(i,j),\ \forall (i,j) \in M \quad (1) \end{aligned}$$

where $T(i,j)$ denotes the set of tie lines between the pair of subsystems i and j, M denotes the set of pair subsystems consisting by tie lines, $p^i_l(v^j_{ib}, v^i_{jb})$ denotes the real power flows over the tie line l from subsystem i to subsystem j measured at the end bus in subsystem i, which is indicated by the superscript in p^i_l; $\underline{p}_l$ and $\bar{p}_l$ denote the lower and upper security power flow limit of $p^i_l(v^j_{ib}, v^i_{jb})$, respectively. A graphical illustration of $v^i_{J_ib}$, v^i_{jb}, v^j_{ib} and the tie line real power flows $p^i_l(v^j_{ib}, v^i_{jb})$ are shown in Fig. 1. For example, $M = \{(1,2),(1,3)\}$, $T(1,2) = \{l_1, l_2\}$, $T(1,3) = \{l_3, l_4, l_5\}$, $J_1 = \{2,3\}$, $v^1_{J_1b} = (v^1_{2b}, v^1_{3b})$, where $v^1_{2b} = (v_{21}, v_{22})$ and $v^1_{3b} = (v_{31}, v_{32}, v_{33})$; furthermore, $v^3_{1b} = (v_{13}, v_{14})$, $v^2_{1b} = (v_{11}, v_{12})$; the tie line real power flows from subsystem 1 to subsystem 3 are $p^1_{l_3}, p^1_{l_4}, p^1_{l_5}$, and from subsystem 1 to subsystem 2 are $p^1_{l_1}, p^1_{l_2}$. We can transform the inequality constraints $\underline{p}_l \leq p^i_l(v^j_{ib}, v^i_{jb}) \leq \bar{p}_l$ (i.e., the security limits) on the tie line real power flow into equality constraints and simple inequality constraints as follows:

$$\begin{aligned} p^i_l(v^j_{ib}, v^i_{jb}) - \bar{z}^i_l - \bar{p}_l &= 0 \\ -p^i_l(v^j_{ib}, v^i_{jb}) - \underline{z}^i_l + \underline{p}_l &= 0 \\ \bar{z}^i_l \leq 0,\ \underline{z}^i_l &\leq 0 \qquad (2) \end{aligned}$$

where $\bar{z}_l^i$ and $\underline{z}_l^i$ denote the slack variables corresponding to $p_l^i(v_{ib}^j, v_{jb}^i)$ in subsystem i.

III. Distributed Algorithm for Solving the DOPFD

The difficulties of the DOPFD are twofold. The first one is for a given u_d,(1) is a large dimension **c**ontinuous **d**istributed **o**ptimal **p**ower **f**low, which is abbreviated as **CDOPF** in this paper; thus, to evaluate the performance of a u_d, we need to solve a CDOPF. The second one is the enormous size of U_d, for example if there are r, say 40, control variables in the whole system and each one has l, say four, possible discrete values, then there are l^r $(= 4^{40} \approx 10^{24})$ possible $u_d's$. Therefore, if we employ the *exhaustive search* method to search the *optimal* u_d in U_d, we need to solve more than 10^{24} continuous CDOPFs. This is definitely computationally intractable not to mention the difficulty in developing a distributed algorithm for solving the CDOPF. Thus, to cope with the difficulty caused by the enormous size of U_d, we will employ the OO strategy to select a *good enough* instead of optimal u_d in U_d and simultaneously solve the CDOPF under this u_d. To accomplish this task, we need to 1) propose a distributed algorithm for solving CDOPFs in the framework of OO strategy and 2) implement the OO strategy in a distributed power system to select a *good enough* u_d. In the following, we will present 1) first.

A. Distributed Algorithm for Solving the CDOPF

Since the CDOPF will appear in the OO strategy more than once, we will use a more general expression to describe the CDOPF.

The considered CDOPF can be stated in the following form:

$$\min f(y) \left(= \sum_{i=1}^{N} f_i(y_i)\right)$$
$$\text{subject to} \quad \bar{g}_i(y_i, y_{J_ib}^i) = 0$$
$$\bar{h}_i(y_i) \le 0,\ i = 1, \ldots, N \qquad (3)$$

in which we can partition $\bar{g}_i(y_i, y_{J_ib}^i)$ into $(\overset{0}{\bar{g}}_i(y_i),\bar{g}_{ib}(y_i, y_{J_ib}^i))$ such that $\overset{0}{\bar{g}}_i$ involves y_i only, while $\bar{g}_{ib}$ involves $y_{J_ib}^i$. Thus, we may use the following to replace the equality constants in (3):

$$\overset{0}{\bar{g}}_i(y_i) = 0 \text{ and } \bar{g}_{ib}(y_i, y_{J_ib}^i) = 0. \qquad (4)$$

Our approach for solving the CDOPF is a combination of the successive quadratic programming (SQP) method with the dual pseudo quasi Newton (DPQN) method [24], such that the quadratic programming problem (QPP) induced in the SQP method is solved by the DPQN method. The SQP method uses the following iterations to solve (3):

$$y_i(k+1) = y_i(k) + \alpha_i(k)\Delta y_i(k),\ i = 1, \ldots, N \qquad (5)$$

where k is the iteration index, $\alpha_i(k)$ is a positive step-size.

The $\Delta y(k) = (\Delta y_1(k), \ldots, \Delta y_N(k))$ in (5) is the optimal solution of the following QPP:

$$\min_{\Delta y} \sum_{i=1}^{N} \Delta y_i^T \nabla_{y_i}^2 f_i \Delta y_i + \nabla_{y_i} f_i^T \Delta y_i$$
$$\text{subject to} \quad \overset{0}{\bar{g}}_i + \nabla_{y_i} \overset{0}{\bar{g}}_i^T \Delta y_i = 0$$
$$\bar{g}_{ib} + \nabla_{y_i} \bar{g}_{ib}^T \Delta y_i + \nabla_{y_{J_ib}^i} \bar{g}_{ib}^T \Delta y_{J_ib}^i = 0$$
$$\bar{h}_i + \nabla_{y_i} \bar{h}_i^T \Delta y_i \le 0,\ i = 1, \ldots, N. \qquad (6)$$

For the sake of notation simplicity, we drop the arguments in the functions f_i, $\overset{0}{\bar{g}}_i$, $\bar{g}_{ib}$, and $\bar{h}_i$. As indicated above, the QPP in (6) will be solved by the DPQN method. Therefore, the DPQN method is an inner loop of the SQP method. That means most of the computations of the proposed distributed algorithm for solving the CDOPF lie in the DPQN method, because the SQP method simply updates y_i, $i = 1, \ldots, N$ by (5) once Δy_i, $i = 1, \ldots, N$ is obtained.

Instead of solving (6) directly, the DPQN method solves the dual problem of (6) as stated in the following.

Let λ denote the Lagrange multiplier vector associated with the equality constrains of the overall system in (6) and let λ_i denote the subvector of λ associated with the equality constraints corresponding to subsystem i. We partition λ_i into $(\overset{o}{\lambda}_i, \lambda_{ib})$, such that $\overset{o}{\lambda}_i$ and λ_{ib} associate with the equality constraints $\overset{0}{\bar{g}}_i + \nabla_{y_i} \overset{0}{\bar{g}}_i^T \Delta y_i = 0$ and $\bar{g}_{ib} + \nabla_{y_i} \bar{g}_{ib}^T \Delta y_i + \nabla_{y_{J_ib}^i} \bar{g}_{ib}^T \Delta y_{J_ib}^i = 0$, respectively. Then the dual problem of (6) can be stated as follows[25]:

$$\max \phi(\lambda) \qquad (7)$$

where the dual function $\phi(\lambda)$ is defined by (8) at the bottom of the page, in which we have put the inequality constraints in (6) as the domain of Δy, denoted by Ω and defined by

$$\Omega = \left\{\Delta y \,\middle|\, \bar{h}_i + \nabla_{y_i} \bar{h}_i^T \Delta y_i \le 0\ ,\ i = 1, \ldots, N\right\}. \qquad (9)$$

The DPQN method uses the following iterations to solve the dual problem (7):

$$\lambda_i(t+1) = \lambda_i(t) + \beta_i(t)\Delta\lambda_i(t),\ i = 1, \ldots, N \qquad (10)$$

where t is the iteration index, and $\beta_i(t)$ is a positive step-size. The $\Delta\lambda(t) = (\Delta\lambda_1(t), \ldots, \Delta\lambda_N(t))$ is obtained from solving the following linear equations:

$$\Phi\Delta\lambda + \nabla_\lambda \phi(\lambda) = 0 \qquad (11)$$

where $\nabla_\lambda \phi(\lambda)$ denotes the gradient of $\phi(\lambda)$ with respect to λ; the block diagonal matrix $\Phi = diag\,(\Phi_1, \ldots, \Phi_N)$ is an *approximate Hessian* of the dual function $\phi(\lambda)$ *without* considering the constraints $\Delta y \in \Omega$, and this is the reason why we name (10)–(11) as the dual *pseudo* quasi-Newton method. The

$$\phi(\lambda) = \min_{\Delta y \in \Omega} \sum_{i=1}^{N} \left\{ \left[\Delta y_i^T \nabla_{y_i}^2 f_i \Delta y_i + \nabla_{y_i} f_i^T \Delta y_i\right] + \overset{0}{\lambda}_i^T \left[\overset{0}{\bar{g}}_i + \nabla_{y_i} \overset{0}{\bar{g}}_i^T \Delta y_i\right] + \lambda_{ib}^T \left[\bar{g}_{ib} + \nabla_{y_i} \bar{g}_{ib}^T \Delta y_i + \nabla_{y_{J_ib}^i} \bar{g}_{ib}^T \Delta y_{J_ib}^i\right] \right\} \qquad (8)$$

formulae for calculating Φ and $\nabla_\lambda\phi(\lambda)$ are described below. The ith diagonal block submatrix of Φ denoted by Φ_i is given by [25]

$$\Phi_i = \begin{bmatrix} \Phi_{i00} & \Phi_{i0b} \\ \Phi_{ib0} & \Phi_{ibb} \end{bmatrix} \tag{12}$$

$$\text{where}\quad \Phi_{i00} = -\nabla_{y_i}\overset{0^T}{\bar{g}_i}\, H_i^{-1}\nabla_{y_i}\overset{0}{\bar{g}_i} \tag{13}$$

$$\Phi_{i0b} = -\nabla_{y_i}\overset{0^T}{\bar{g}_i}\, H_i^{-1}\nabla_{y_i}\bar{g}_{ib} \tag{14}$$

$$\Phi_{ib0} = -\nabla_{y_i}\bar{g}_{ib}^T H_i^{-1}\nabla_{y_i}\overset{0}{\bar{g}_i} \tag{15}$$

$$\Phi_{ibb} = -\nabla_{y_i}\bar{g}_{ib}^T H_i^{-1}\nabla_{y_i}\bar{g}_{ib} \tag{16}$$

and the matrix H_i in (13)–(16) is defined by the following:

$$H_i = \nabla_{y_i}^2 f + \delta I \tag{17}$$

in which the matrix I is an identity matrix with dimensions of $|y_i| \times |y_i|$, and δ is a small positive real number to make H_i positive definite. Note that in (13)–(16), we do not consider the constraint $\Delta y \in \Omega$. Since Φ is block diagonal, we can decompose (11) into the following:

$$\Phi_i\Delta\lambda_i + \nabla_{\lambda_i}\phi(\lambda) = 0,\ i = 1,\ldots,N \tag{18}$$

where $(\nabla_{\lambda_1}\phi(\lambda),\ldots,\nabla_{\lambda_N}\phi(\lambda)) = \nabla_\lambda\phi(\lambda)$. Clearly, Φ_i is a *negative definite* matrix for every $i = 1,\ldots,N$, so is Φ. Consequently, $\Delta\lambda(t)$ in (10) obtained from solving (18) is an *assent direction* of $\phi(\lambda)$ at $\lambda(t)$. We can partition $\nabla_{\lambda_i}\phi(\lambda)$ into $(\nabla_{\overset{0}{\lambda_i}}\phi(\lambda), \nabla_{\lambda_{ib}}\phi(\lambda))$, which can be computed by the following [25]:

$$\nabla_{\overset{0}{\lambda_i}}\phi(\lambda) = \overset{0}{\bar{g}_i} + \nabla_{y_i}\overset{0^T}{\bar{g}_i}\,\Delta\hat{y}_i \tag{19}$$

$$\nabla_{\lambda_{ib}}\phi(\lambda) = \bar{g}_{ib} + \nabla_{y_i}\bar{g}_{ib}^T\Delta\hat{y}_i + \nabla_{y^i_{J_ib}}\bar{g}_{ib}^T\Delta\hat{y}^i_{J_ib} \tag{20}$$

in which $\Delta\hat{y}_i, i = 1,\ldots,N$, is the optimal solution of the minimization problem on the RHS of (8). Thus to compute $\nabla_\lambda\phi(\lambda)$ using (19) and (20), we need to solve the minimization problem on the RHS of (8) first as stated in the following. The constraint set Ω in (9) can be expressed as $\Omega = \bigcup_{i=1}^{N}\Omega_i$, where $\Omega_i = \{\Delta y_i \,|\bar{h}_i + \nabla_{y_i}\bar{h}_i^T\Delta y_i \le 0\}$ and $\Omega_i \cap \Omega_j$ = empty set if $i \ne j$. Though not that trivial, the objective function of the minimization problem on the RHS of (8) is separable as illustrated below. The coupling between subsystems is the last term inside the big bracket in (8). We define λ^j_{ib} as the subvector of λ_{ib} associated with the constraint $\bar{g}^j_{ib}+\nabla_{y_i}\bar{g}^{jT}_{ib}\Delta y_i+\nabla_{y^i_{jb}}\bar{g}^{jT}_{ib}\Delta y^i_{jb} = 0$, which is part of $\bar{g}_{ib} + \nabla_{y_i}\bar{g}^T_{ib}\Delta y_i + \nabla_{y^i_{J_ib}}\bar{g}^T_{ib}\Delta y^i_{J_ib} = 0$ that involves y^i_{jb}. Thus $\lambda_{ib} = (\lambda^j_{ib}), j \in J_i$ and the last term inside the big bracket in (8) regarding subsystem i can be rewritten as

$$\lambda^T_{ib}[\bar{g}_{ib} + \nabla_{y_i}\bar{g}^T_{ib}\Delta y_i] + \sum_{j\in J_i}\lambda_{ib}{}^{jT}[\nabla_{y^i_{jb}}\bar{g}^{jT}_{ib}\Delta y^i_{jb}]. \tag{21}$$

The relationship between λ_{ib}, λ^j_{ib}, $\bar{g}_{ib}$, $\bar{g}^j_{ib}$, $\Delta y^i_{J_ib}$ and Δy^i_{jb} can be illustrated by the aid of Fig. 2.

In this figure, each boundary bus is associated with the equality constraint, Lagrange multiplier and the variable; however, due to space limitation, we only mark necessary notations. Thus, we have $J_1 = \{2,3\}$; $\bar{g}_{1b} = (\bar{g}^2_{1b},\bar{g}^3_{1b})$, where $\bar{g}^2_{1b} = (\bar{g}_{11},\bar{g}_{12})$, and $\bar{g}^3_{1b} = (\bar{g}_{13},\bar{g}_{14})$; $\lambda_{1b} = (\lambda^2_{1b},\lambda^3_{1b})$, where $\lambda^2_{1b} = (\lambda_{11},\lambda_{12})$ and $\lambda^3_{1b} = (\lambda_{13},\lambda_{14})$; $\Delta y^1_{J_1b} = (\Delta y^1_{2b},\Delta y^1_{3b})$, where $\Delta y^1_{2b} = (\Delta y_{21},\Delta y_{22})$ and $\Delta y^1_{3b} = (\Delta y_{31},\Delta y_{32},\Delta y_{33})$. Therefore, the complicating variables in (21) is Δy^i_{jb}, $j \in J_i$, which do not belong to subsystem i. However, taking the summation outside the big bracket in (8), $\sum_{i=1}^N$, into account and suitably rearranging terms, we can rewrite $\sum_{i=1}^N(\sum_{j\in J_i}\lambda^{jT}_{ib}[\nabla_{y^i_{jb}}\bar{g}^{jT}_{ib}\Delta y^i_{jb}])$, where the term inside the parenthesis is the last term in (21), as $\sum_{i=1}^N\sum_{j\in J_i}\lambda^{iT}_{jb}\nabla_{y^j_{ib}}\bar{g}^{iT}_{jb}\Delta y^j_{ib}$, where λ^i_{jb} denotes the Lagrange multiplier vector associated with the equality constraints in subsystem $j \in J_i$ but involving the variables in subsystem i; for example, in Fig. 2, $\lambda^1_{2b} = [\lambda_{21},\lambda_{22}]$ and $\lambda^1_{3b} = [\lambda_{31},\lambda_{32},\lambda_{33}]$. Therefore, after such rearrangement, the minimization problem on the RHS of (8) is separable and can be decomposed into the following N independent minimization subproblems: For $i = l,\ldots,N$

$$\min_{\Delta y_i\in\Omega_i}[\Delta y_i^T\nabla^2_{y_i}f_i\Delta y_i + \nabla_{y_i}f_i^T\Delta y_i] + \overset{o^T}{\lambda_i}[\overset{o}{\bar{g}_i} + \nabla_{y_i}\overset{o^T}{\bar{g}_i}\Delta y_i]$$
$$+\lambda^T_{ib}[\bar{g}_{ib} + \nabla_{y_i}\bar{g}^T_{ib}\Delta y_i] + \sum_{j\in J_i}[\lambda^{iT}_{jb}\nabla_{y^j_{ib}}\bar{g}^{iT}_{jb}\Delta y^j_{ib}]. \tag{22}$$

Once the optimal solution of (22), $\Delta\hat{y}_i$, is obtained for all i, we can calculate $\nabla_{\lambda_i}\phi(\lambda)$ by (19) and (20). Subsequently, $\Delta\lambda_i$, $i = 1,\ldots,N$ can be solved from (18).

Remark 1: As indicated previously, $\Delta\lambda(t)$ is an *ascent direction* for the dual function $\phi(\lambda)$ at $\lambda(t)$. Thus if the step-size $\beta_i(t)$ is determined by the Armijo's rule [24] or is a small enough constant, the DPQN method (10) and (18) will converge. Similarly, $\Delta y(k)$ is a descent direction of the objective function $f(y) = \sum_{i=1}^N f_i(y_i)$ at $y(k)$, which is true because typical $f(\cdot)$ for OPF such as total generation cost or total system losses is locally convex, thus if the step-size $\alpha_i(k)$ is determined by the Armijo's rule [24] or is a small enough constant, the SQP method (5) and (6) will converge. Since Armijo's rule is a centralized step-size determination rule, for the sake of implementation in distributed computer network, we had better employ a constant step-size. Based on our extensive simulation experiences, 0.9 is a good choice of small enough constant for both $\alpha_i(k)$ and $\beta_i(t)$.

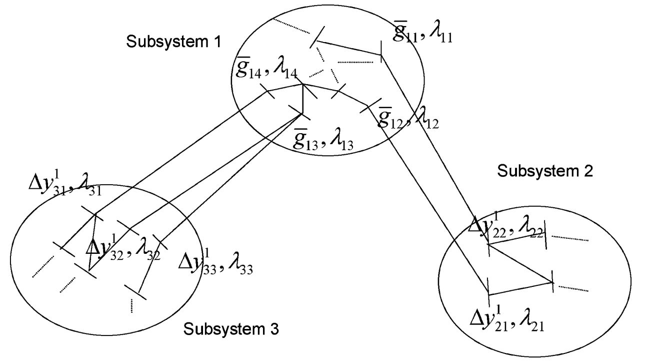

Subsystem 1
$\bar{g}_{11},\lambda_{11}$
$\bar{g}_{14},\lambda_{14}$
$\bar{g}_{12},\lambda_{12}$
$\bar{g}_{13},\lambda_{13}$
Subsystem 2
$\Delta y^1_{31},\lambda_{31}$
$\Delta y^1_{32},\lambda_{32}$
$\Delta y^1_{33},\lambda_{33}$
$\Delta y^1_{22},\lambda_{22}$
$\Delta y^1_{21},\lambda_{21}$
Subsystem 3

Fig. 2. Relationship between λ_{ib}, λ^j_{ib}, $\bar{g}_{ib}$, $\bar{g}^j_{ib}$, $\Delta y^i_{J_ib}$, and Δy^i_{jb}.

1) Complete Decomposition and Parallel Computation (Resolving the Difficulty Caused by Large Dimension): All the computation formulae in the SQP and DPQN methods (5), (10), (13)–(16), (18)–(20), and (22) are decoupled and can be carried out independently and in parallel. This property resolves the difficulty caused by large dimensionality of CDOPF.

2) Required Data Communication: As indicated previously, all the computation formulae in the SQP and DPQN methods are decoupled, however there are data communications required in performing (14)–(16), (20) and (22). Specifically, to prepare Φ_i in (18), we need to perform (13)–(16) in subsystem i; while preparing $\nabla_{y_i}\bar{g}_{ib}$ in (14)–(16), we require the data of $y^i_{J_ib}$ or y^i_{jb}, $j \in J_i$, from subsystems $j \in J_i$, because $\bar{g}_{ib}(= \bar{g}_{ib}(y_i, y^i_{J_ib}))$ involves $y^i_{J_ib}$. Similar situations occur to computing $\nabla_{\lambda_{ib}}\phi(\lambda)$ in (20), in which we need from subsystem $j \in J_i$ the data $y^i_{J_ib}$ to prepare $\bar{g}_{ib}$ and $\nabla_{y^i_{J_ib}}\bar{g}_{ib}$, and the data $\Delta\hat{y}^i_{J_ib}$ to perform (20). In addition, to obtain $\Delta\hat{y}_i$ from solving (22), we need the data λ^i_{jb}, and $\nabla_{y^j_{ib}}\bar{g}^i_{jb}$, $j \in J_i$ from subsystems $j \in J_i$. However, to prepare the just mentioned data $\nabla_{y^j_{ib}}\bar{g}^i_{jb}$ in subsystem j, we need the data y^j_{ib} from subsystem i. Fortunately, the data required in subsystem $i(j)$ from subsystems $j \in J_i$ (subsystem i) are only those on the boundary buses. Therefore, the burden of data communication is very light. This indicates that the proposed distributed algorithm for solving CDOPF is very suitable for implementation in a distributed computer network.

3) Convergence Determination and Synchronization of Distributed Computation: Regarding convergence determination, we assign a subsystem namely the *root subsystem* to monitor the convergence of the SQP and DPQN methods in the distributed computer network. Following is the distributed algorithmic steps for solving CDOPF in each subsystem i, and the steps for determining the convergence will be executed in the root subsystem only, which will be indicated specifically. Regarding the *synchronization* in each algorithmic step, we employ the concept of *asynchronous synchronization*, which implies the computations of an algorithmic step will start only when all the required data from the connecting subsystems $j \in J_i$ are received.

4) Distributed Algorithm I for Subsystem i: Now we are ready to state the distributed algorithmic steps for subsystem i to solve the CDOPF (3).

Step 0) Initially guess $y_i(0)$ and $\lambda_i(0)$; set $k = 0, t = 0$.

Step 1) Calculate the values of $\nabla_{y_i} f_i$, $\nabla^2_{y_i} f_i$, $\overset{0}{g_i}$, $\nabla_{y_i}\overset{0}{g_i}$; send $y^j_{ib}(k)$ to subsystem j for every $j \in J_i$.

Step 2) Once receiving $y^i_{jb}(k)$ from all subsystems $j \in J_i$. calculate $\bar{g}_{ib}$, $\nabla_{y_i}\bar{g}_{ib}$, $\nabla_{y^i_{J_ib}}\bar{g}_{ib}$ (i.e., $(\nabla_{y^i_{jb}}\bar{g}^j_{ib})$, $j \in J_i$), and Φ_i (by (13)–(16)); send $\nabla_{y^i_{jb}}\bar{g}^j_{ib}$ to subsystem j for every $j \in J_i$.

Step 3) Once receiving $\nabla_{y^j_{ib}}\bar{g}^i_{jb}$ from all subsystems $j \in J_i$, go to **Step 4**.

Step 4) Send $\lambda^j_{ib}(t)$ to subsystem j for every $j \in J_i$.

Note: The reason why we send $\nabla_{y^i_{jb}}\bar{g}^j_{ib}$ and λ^j_{ib} to subsystem j for every $j \in J_i$ in separate steps is because $\nabla_{y^i_{jb}}\bar{g}^j_{ib}$ is constant for the whole DPQN method in iteration k of the SQP method, while $\lambda^j_{ib}(t)$ varies for each iteration t of the DPQN method as can be seen in **Step 10**.

Step 5) Once receiving all $\lambda^i_{jb}(t)$, $j \in J_i$, obtain $\Delta\hat{y}_i$ from solving the ith minimization subproblem in (22).

Step 6) Send $\Delta\hat{y}^j_{ib}$ to subsystem j for every $j \in J_i$.

Step 7) Once all $\Delta\hat{y}^i_{jb}, j \in J_i$ are received, calculate $\nabla_{\lambda_i}\phi(\lambda(t))$ by (19) and (20).

Step 8) Solve $\Delta\lambda_i(t)$ from (18).

Step 9) If $\|\Delta\lambda_i(t)\|_\infty < \varepsilon$, send a signal to the root subsystem to inform the convergence of the DPQN method in this subsystem and go to **Step 11** if $i = \text{root subsystem}$ or go to **Step 12** if $i \neq \text{root subsystem}$. If $\|\Delta\lambda_i(t)\|_\infty \geq \varepsilon$, go to **Step 10**.

Step 10) Update $\lambda_i(t+1)$ by (10) with an experienced step-size $\beta(t)(= 0.9)$, set $t = t + 1$ and return to **Step 4**.

Step 11 **(for root subsystem only)** Once receiving the signal indicating the convergence of the DPQN method from all subsystems, send a convergence signal of the DPQN method to subsystem i for all $i = 1, \ldots, N$.

Step 12) Once receiving the convergence signal of the DPQN method from the root subsystem, set $\Delta y_i(k) = \Delta\hat{y}_i$. If $\|\Delta y_i(k)\|_\infty < \varepsilon$, send a signal to the root subsystem to inform the convergence of the SQP method in this subsystem and wait for further convergence signal from the root subsystem; otherwise, update $y_i(k+1)$ by (5) with an experienced step-size $\alpha(k)(= 0.9)$, set $k = k + 1$ and return to **Step 1**.

Step 13 **(for root subsystem only)** Once receiving the signal indicating the convergence of the SQP method from all subsystems, send a signal to all subsystems to continue the algorithmic steps in Distributed Algorithm II, which will be presented later.

B. OO Theory-Based Distributed Algorithm to Solve DOPFD for a Good Enough Solution

For real-time application purpose, we would rather obtain a *good enough* solution within reasonable computation time than get the optimal solution using incredibly long time. The OO theory [22], [23] is a recently developed optimization technique to solve hard optimization problems, such as the combinatorial optimization problem, for a *good enough* solution with *high probability* using limited computation time. Based on the observation that the *performance order* of discrete solutions is likely preserved even evaluated by a *surrogate model*, the OO theory concludes the following: *Suppose we simultaneously evaluate a large set of alternatives very approximately and order them according to the approximate evaluation. Then there is high probability that we can find the actual good alternatives if we limit ourselves to the top n% of the observed good choices.* According to this conclusion, we can quickly evaluate the estimated performances of all discrete solutions in the *candidate* discrete solution set using a surrogate model and rank them to select a set of top ranked solutions. Suppose we employ a more refined surrogate model, there will be more actual good discrete solutions contained in the selected set of top ranked solutions, however at the cost of more evaluation time for each discrete solution. Therefore, if the size of the *primitive* discrete solution set is huge, the above evaluation, ranking and selection process can repeat for more than one stage, such that 1) the employed surrogate models will be refined stage by stage, and 2) the set of top ranked solutions selected in one stage will serve as the candidate solution set of next stage. In the final stage, the exact model will be used to evaluate all discrete solutions of the largely trimmed candidate solution set, and the resulted best discrete solution is the *good enough* solution that we seek. Thus, comparing with the exhaustive search method, in which the exact model is used to evaluate every discrete solution, the proposed OO strategy is a process to select a *good enough* u_d from the *enormous* U_d using limited computation time. However, some ranking and selection are centralized behaviors. Thus, we need to assign the root subsystem, which is responsible for determining the convergence of SQP and DPQN methods in Distributed Algorithm I, to carry out this task. Consequently, our idea to carry out some centralized OO concept in a distributed power system is as follows. Each subsystem i will evaluate the estimated performances of $u_{d_i}'s$ and send the evaluation results to the root subsystem. The root subsystem will then rank and select a set of top ranked $u_d's$ based on the gathering estimated performances of $u_{d_i}'s$ sent from all subsystems $i = 1, \ldots, N$ and send the subvector $u_{d_i}'s$ of the selected $u_d's$ to subsystem i. Based on this idea, the proposed distributed algorithm for solving DOPFD for a *good enough* solution consists of three stages as described below.

1) Stage 1: There are two parts in this stage. The first part is to reduce the size of the primitive candidate solution set, U_d, to 2^r based on the optimal solution of the CDOPF.

To achieve this, replacing the discrete u_{d_i} in (1) by its continuous version u_{c_i} and replacing the inequality constraints on tie line real power flows by the transformed equality and simple inequality constraints, (2), we obtain a CDOPF shown in the following:

$$\begin{aligned} \min f(x) &\left(= \sum_{i=1}^{N} f_i(x_i) \right) \\ \text{subject to} \quad g_i(x_i, v^i_{J_i b}, u_{c_i}) &= 0 \\ h_i(x_i) &\le 0, \; i = 1, \ldots, N \\ p^i_l(v^j_{ib}, v^i_{jb}) - \bar{z}^i_l - \bar{p}_l &= 0 \\ -p^i_l(v^j_{ib}, v^i_{jb}) - \underline{z}^i_l + \underline{p}_l &= 0 \\ \bar{z}^i_l \le 0, \; \underline{z}^i_l \le 0, \; \forall l \in T(i,j), \; &\forall (i,j) \in M. \end{aligned} \tag{23}$$

Note that the upper and lower bounds of u_{c_i} is represented by the maximum and minimum values of u_{d_i}, respectively, and these bounded constraints on u_{c_i} are included in the inequality constraints $h_i(y_i) \le 0$ in (23). We define the functions $\bar{p}^i_l(v^j_{ib}, v^i_{jb}, \; \bar{z}^i_l, \bar{p}_l) = \bar{p}^i_l(v^j_{ib}, v^i_{jb}) - \bar{z}^i_l - \bar{p}_l$ and $\underline{p}^i_l(v^j_{ib}, v^i_{jb}, \underline{z}^i_l, \underline{p}_l) = -p^i_l(v^j_{ib}, v^i_{jb}) - \underline{z}^i_l + \underline{p}_l$, and denote the vector functions $\bar{p}^i_L = (\bar{p}^i_l(v^j_{ib}, v^i_{jb}, \bar{z}^i_l, \bar{p}_l))$, $l \in T(i,j)$ and $\underline{p}^i_L = (\underline{p}^i_l(v^j_{ib}, v^i_{jb}, \underline{z}^i_l, \underline{p}_l))$, $l \in T(i,j)$. We define $z^i_l = (\bar{z}^i_l, \underline{z}^i_l)$ and denote the vector of slack variables $z^i_L = (z^i_l)$, $l \in T(i,j)$. Setting $(x_i, u_{c_i}, z^i_L) = y_i$, $v^i_{J_i b} = y^i_{J_i b}$, $(g_i, \bar{p}^i_L, p^i_L) = \bar{g}_i$, and $(h_i, z^i_L) = \bar{h}_i$ then we can use the Distributed Algorithm I to solve the CDOPF, (23). It is worth noting that the optimal objective value of (1) cannot compete with that of (23), because the optimal u_d for (1) is only a feasible solution of (23). Since most of the objective functions considered in the OPF, such as total generation cost and total system losses, are continuously differentiable and locally convex, the 2^r neighboring discrete control solutions of the *continuous optimal solution*, u^*_c, of (23) should consist of good enough discrete control solutions of (1). Thus, we have reduced the size of candidate solution set, U_d, to 2^r.

However, 2^r, for example $r = 40$, is still a very large number. Thus, the second part of this stage is to further reduce the size of the candidate solution set from 2^r to $2^{\sum_{i=1}^{N} n_i}$ based on sensitivity analysis, where n_i, a faction of r_i say $(1/4)r_i$, is predetermined. To achieve this, we proceed as follows. Since some components of $u^*_{c_i}$ may already be very close to the closest discrete values or the discrete steps of some discrete control variables are very small such as the transformer tap ratio, we can fix those components of $u^*_{c_i}$ to their closest discrete values if the corresponding deviations do not affect the optimal objective value of the CDOPF significantly. Thus, in the rest of this stage, we will employ the *sensitivity theory* [25, Ch. 10, Sec. 10.7, p. 312] to find such components. The sensitivity theory states that the sensitivity, or the gradient, of f with respect to the value change of the equality constraint function $\bar{g}_i (= (\overset{0}{\bar{g}}_i, \bar{g}_{ib}))$ equals the negative Lagrange multiplier, $-\lambda_i (= (-\overset{0}{\lambda}_i, -\lambda_{ib}))$. We let $u^*_{c_{i,j}}$ and $u_{d_{i,j}}$ denote the jth component of $u^*_{c_i}$ and u_{d_i}, respectively, and define $\lceil \Delta u_{c_{i,j}} \rceil = \lceil u^*_{c_{i,j}} \rceil - u^*_{c_{i,j}}$ and $\lfloor \Delta u_{c_{i,j}} \rfloor = u^*_{c_{i,j}} - \lfloor u^*_{c_{i,j}} \rfloor$, where $\lceil u^*_{c_{i,j}} \rceil$ and $\lfloor u^*_{c_{i,j}} \rfloor$ denote the closest discrete value on the right-hand side and left-hand side of $u^*_{c_{i,j}}$, respectively. The deviation $\lceil \Delta u_{c_{i,j}} \rceil$(or $\lfloor \Delta u_{c_{i,j}} \rfloor$) will cause the value change on $\overset{0}{\bar{g}}_i(y^*_i)$ and $\bar{g}_{ib}(y^*_i, y^{i*}_{J_i b})$. Then the deviation of the overall optimal objective value of (23), Δf^*, caused by the deviation $\lceil \Delta u_{c_{i,j}} \rceil$ (or $\lfloor \Delta u_{c_{i,j}} \rfloor$), denoted by $\Delta f^*(\lceil \Delta u_{c_{i,j}} \rceil)$(or $\Delta f^*(\lfloor \Delta u_{c_{i,j}} \rfloor)$) can be calculated based on the above mentioned sensitivity theory and *chain rule* by

$$\begin{aligned} \Delta f^*(\lceil \Delta u_{c_{i,j}} \rceil \approx -\overset{o}{\lambda}{}^T_i \nabla_{u_{c_{i,j}}} \overset{o}{\bar{g}}_i(y^*_i)^T \lceil \Delta u_{c_{i,j}} \rceil \\ -\lambda^T_{ib} \nabla_{u_{c_{i,j}}} \bar{g}_{ib}(y^*_i, y^{i*}_{J_i b})^T \lceil \Delta u_{c_{i,j}} \rceil . \end{aligned} \tag{24}$$

Smaller $|\Delta f^*(\lceil \Delta u_{c_{i,j}} \rceil)|$ (or $|\Delta f^*(\lfloor \Delta u_{c_{i,j}} \rfloor)|$) implies that $\lceil \Delta u_{c_{i,j}} \rceil$ (or $\lfloor \Delta u_{c_{i,j}} \rfloor$) will affect f^* very lightly. We let $|\Delta f^*(u_{c_{i,j}})| = \min\{|\Delta f^*(\lceil \Delta u_{c_{i,j}} \rceil)|, |\Delta f^*(\lfloor \Delta u_{c_{i,j}} \rfloor)|\}$ and rank $u_{c_{i,j}}, j = 1, \ldots, r_i$ based on the values of $|\Delta f^*(u_{c_{i,j}})|$ such that the smaller the latter, the higher rank the former. Then, for each of the top ranked $r_i - n_i u_{c_{i,j}}'s$, we will fix the corresponding discrete control variable $u_{d_{i,j}}$ at $\lceil u^*_{c_{i,j}} \rceil$ if $|\Delta f^*(u_{c_{i,j}})| = |\Delta f^*(\lceil \Delta u_{c_{i,j}} \rceil)|$ or $\lfloor u^*_{c_{i,j}} \rfloor$ if $|\Delta f^*(u_{c_{i,j}})| = |\Delta f^*(\lfloor \Delta u_{c_{i,j}} \rfloor)|$.

Now, since each of n_i yet fixed discrete control variables in subsystem i can take two neighboring discrete values, there are 2^{n_i} possible $u_{d_i}'s$ in subsystem i, and we denote them by $u_{d_i}(l)$, $l = 1, \ldots, 2^{n_i}$. Combination of the N subsystem's $n_i u_{d_i}(l)'s$ results in $2^{\sum_{i=1}^{N} n_i}$ possible $u_d's$. In other words, we have $2^{\sum_{i=1}^{N} n_i}$ possible $u_d's$ from the overall system point of view, thus we have further reduced the size of the candidate solution set from 2^r to $2^{\sum_{i=1}^{N} n_i}$.

2) Stage 2: In this stage, we will estimate the performance of the $2^{\sum_{i=1}^{N} n_i} u_d{}'s$ obtained in Stage 1 using sensitivity model and select the top ranked s, say 50, $u_d{}'s$.

To do so, we will compute the estimated deviation of the optimal objective value due to the deviation $\Delta u_{c_i}(l) = u_{c_i}^* - u_{d_i}(l)$ for each of the $2^{n_i} u_{d_i}(l)'s$ in subsystem i by

$$\Delta f^*(\Delta u_{c_i}(l)) \approx -\overset{o}{\lambda}{}_i^T \nabla_{u_{c_i}} \overset{o}{\overline{g_i}}(y_i^*)^T \Delta u_{c_i}(l) \\ -\lambda_{ib}^T \nabla_{u_{c_i}} \overline{g}_{ib}(y_i^*, {y_{J_i b}^i}^*)^T \Delta u_{c_i}(l). \quad (25)$$

The reason that supports (25) is exactly the same as that supports (24) except for $\Delta u_{c_i}(l)$ being a vector, while $\lceil \Delta u_{c_{i,j}} \rceil$ is a component. Then, subsystem i will send the 2^{n_i} pairs of $(u_{d_i}(l), \Delta f^*(\Delta u_{c_i}(l)))'s$ to the root subsystem. Now in the root subsystem, we label these $2^{\sum_{i=1}^{N} n_i}$ possible $u_d{}'s$ as $u_d(j)$, $j = 1, \ldots, 2^{\sum_{i=1}^{N} n_i}$ and $u_{d_i}(j)$, the subvector of $u_d(j)$ corresponding to subsystem i, is one of the $2^{n_i} u_{d_i}(l)'s$ sent from subsystem i. Due to the *linear property* of the sensitivity theory [25, Ch. 10, Sec. 10.7, p. 312], we have $\Delta f^*(\Delta u_c(j)) = \sum_{i=1}^{N} \Delta f^*(\Delta u_{c_i}(j))$, where $\Delta u_c(j) = u_c^* - u_d(j)$, $\Delta u_{c_i}(j)$ is the subvector of $\Delta u_c(j)$ corresponding to subsystem i, and $\Delta f^*(\Delta u_{c_i}(j))$ is one of the $2^{n_i} \Delta f^*(\Delta u_{c_i}(l))'s$ sent from subsystem i. As indicated previously, the optimal objective value of (1) is larger than that of (23), because the optimal u_d for (1) is only a feasible solution for (23). Therefore, smaller $|\Delta f^*(\Delta u_c(j))|$ implies a smaller deviation of the optimal objective value of (23) caused by the deviation $\Delta u_c(j)$. Thus, the root subsystem will then rank these $2^{\sum_{i=1}^{N} n_i} u_d(j)'s$ based on the corresponding values of $|\Delta f^*(\Delta u_c(j))|$ such that the $u_d(j)$ with smaller $|\Delta f^*(\Delta u_c(j))|$ has higher rank. Subsequently, we can pick the top ranked $s u_d(j)'s$ and relabel them as $u_d(j_m)$, $m = 1, \ldots, s$. Then the root subsystem will send to subsystem i the corresponding subvectors $u_{d_i}(j_m)$, $m = 1, \ldots, s$, for every $i = 1, \ldots, N$. Thus, we have further reduced the size of the candidate solution set from $2^{\sum_{i=1}^{N} n_i}$ to s. In the meantime, the root subsystem will inform each subsystem to proceed with next stage.

3) Stage 3: In this stage, we will use the exact model to evaluate the $s\, u_d(j_m)'s$ resulted in Stage 2, and the best one is the good enough u_d that we seek.

The exact model for evaluating the s discrete control solutions $u_d(j_m)$, $m = 1, \ldots, s$ obtained in Stage 2 is (1), but in which the u_{d_i} is replaced by the fixed $u_{d_i}(j_m)$ and becomes a CDOPF. Replacing the inequality constraints on tie line real power flows by the transformed equality and simple inequality constraints (2), the exact model will be the same as (23) except for substituting the continuous variables u_{c_i} by the fixed $u_{d_i}(j_m)$. Using the similar treatment as to (23), we can set $(x_i, z_L^i) = y_i$, $v_{J_i b}^i = y_{J_i b}^i$, $(g_i, \bar{p}_L^i, \underline{p}_L^i) = \bar{g}_i$ and $(h_i, z_L^i) = \bar{h}_i$, where $\bar{p}_L^i, \underline{p}_L^i$ and z_L^i have been defined in the paragraph followed by (23). Thus, we can use Steps 1–13 of the Distributed Algorithm I to solve the resulted CDOPF. Note that we do not need Step 0 of Distributed Algorithm I, because the resulted operating point from Stage 2 (that is the optimal solution of the CDOPF (23), but in which u_c^* is replaced by $u_d(j_m)$) will be the initial operating point of this Stage. Thus we can proceed with picking the best $u_d(j_m)$ among $u_d(j_1), \ldots, u_d(j_s)$ as follows. When receiving the corresponding subvectors of the s discrete control solutions resulted in Stage 2 from the root subsystem, all subsystems will cooperate to solve the s CDOPFs. We let $x^*(u_d(j_m)) = (x_1^*(u_d(j_m)), \ldots, x_N^*(u_d(j_m)))$ denote the optimal solution of the CDOPF for the fixed $u_d(j_m)$. Once the s CDOPFs are solved, each subsystem will send the s pairs of $(u_{d_i}(j_m), f_i(x_i^*(u_d(j_m))))$ to the root subsystem. The root subsystem will calculate the objective value of the overall system for the given $u_d(j_m)$ by taking the sum $\sum_{i=1}^{N} f_i(x_i^*(u_d(j_m))$. We denote $u_d(j_m')$ as the $u_d(j_m)$ corresponding to the smallest $\sum_{i=1}^{N} f_i^*(x_i^*(u_d(j_m)))$, among $m = 1, \ldots, s$. Then the $u_d(j_m')$, associated with the $x^*(u_d(j_m'))$ will be the good enough solution that we seek.

Remark 2: Solving $s(= 50)$ CDOPFs seems to be computationally very intensive. In fact, it is not, because each $u_d(j_m)$ is neighboring to u_c^*, and the initial operating point resulted from Stage 2 is already close to the solution. Therefore, in almost all the s CDOPFs, it takes only one iteration of the Distributed Algorithm I excluding Step 0 to obtain the solution.

Remark 3: We say that a u_d is *feasible* if the CDOPF resulted by setting the u_d in (1) to be the given u_d has optimal solution. Now, one of the conventional approaches to centralized OPF with discrete control variables is using an *approximating technique* to obtain an approximate discrete control solution then rounding off to the closest discrete values. However, *arbitrarily* rounding off may cause *infeasibility* problem as indicated in [16]. Our approach can circumvent such undesirable situation, because if there is at least one feasible u_d among the $s\, u_d{}'s$ resulted in Stage 2, the good enough solution obtained in Stage 3 must be a feasible one. In other words, we significantly increase the probability of obtaining a good enough *feasible* solution. For example, suppose *half* of the $2^r u_d{}'s$ resulted in the first part of Stage 1 are feasible. The probability of getting *feasible* u_d by arbitrarily rounding off is then 0.5. However, the probability that the good enough u_d obtained by our approach is *feasible* is

$$1 - (\text{the probability that none of the } s u_d{}'s \text{ is feasible}) \\ = (0.5)^{50} \approx 1.0.$$

Now we are ready to state the algorithmic steps of the distributed algorithm for each subsystem i to solve DOPFD for a good enough solution, and the steps executed only in the root subsystem will be specifically indicated.

C. Distributed Algorithm II for Subsystem i

Step 0 **(for root subsystem only)** Command all subsystems to start.

Step 1) When receiving the command from the root subsystem, perform the first part of Stage 1 using Distributed Algorithm I.

Step 2) When receiving the convergence signal of CDOPF from the root subsystem, perform the second part of Stage 1, such that $n_i - r_i u_{d_{i,j}}{}'s$ will be fixed at one side of the closest discrete value. For the rest yet fixed n_i components of u_{d_i}, we have 2^{n_i} possible $u_{d_i}{}'s$, which are relabeled as $u_{d_i}(l)$, $l = 1, \ldots, 2^{n_i}$. Compute $\Delta f^*(\Delta u_{c_i}(l))$ by (25), where $\Delta u_{c_i}(l) = u_{c_i}^* - u_{d_i}(l)$ as indicated in Stage 2. Send the 2^{n_i}

pairs of $(u_{d_i}(l), \Delta f^*(\Delta u_{c_i}(l))), l = 1, \ldots, 2^{n_i}$ to the root subsystem.

Step 3 **(for root subsystem only)** When receiving the 2^{n_i} pairs of $(u_{d_i}(l), \Delta f^*(\Delta u_{c_i}(l))), l = 1, \ldots, 2^{n_i}$ from subsystem i for all $i = 1, \ldots, N$, the root subsystem will pick the best s from the $2^{\sum_{i=1}^{N} n_i} u_d's$ based on the sensitivity model as stated in Stage 2. Relabel the picked $s\, u_d's$ as $u_d(j_m)$, $m = 1, \ldots, s$ and send $u_{d_i}(j_m), m = 1, \ldots, s$ to subsystem i for all $i = 1, \ldots, N$.

Step 4) Once receiving the s subvectors $u_{d_i}(j_m), m = 1, \ldots, s$ from the root subsystem, start to solve the s CDOPFs using Steps 1–13 of Distributed Algorithm I as stated in Stage 3. Once the s CDOPFs are solved, send the s pairs of $(u_{d_i}(j_m), f_i(x_i^*(u_d(j_m)))), m = 1, \ldots, s$, to the root subsystem.

Step 5 **(for root subsystem only)** When receiving the s pairs of $(u_{d_i}(j_m), f_i(x_i^*(u_d(j_m)))), m = 1, \ldots, s$, from all subsystems $i = 1, \ldots, N$, the root subsystem will take the sum $\sum_{i=1}^{N} f_i(x_i^*(u_d(j_m)))$ for each $m = 1, \ldots, s$, and based on which pick the best $u_d(j_m)$ as stated in Stage 3. Relabel the best $u_d(j_m)$ as $u_d(j'_m)$ and send $u_{d_i}(j'_m)$ to subsystem i, for all $i = 1, \ldots, N$.

Step 6) Once receiving the good enough subvector $u_{d_i}(j'_m)$ from the root subsystem, stop the algorithm and output the solution $(x_i^*(u_d(j'_m)), u_{d_i}(j'_m))$.

IV. Test Results

In this section, we will demonstrate 1) the validity of Distributed Algorithm II by implementing it in a real PC network and 2) the computational efficiency and the goodness of the obtained good enough solutions by indirect comparisons with existing centralized global searching techniques. (To our best knowledge, there is no method dealing with the DOPFD considered in this paper so far, indirect comparison is all we can do.)

We have implemented our Distributed Algorithm II in a 4-PC network to solve the DOPFD of the IEEE 118-bus and TP 244-bus systems, both of which are arbitrarily partitioned into four subsystems namely S_1, S_2, S_3, S_4 and T_1, T_2, T_3, T_4, respectively. Each subsystem is associated with a PC. Some details regarding number of buses, number of transmission lines and number of generation buses in each subsystem are shown in Table I. It should be noted that the values of conductance of the transmission lines in the TP 244-bus system are much higher than that of the IEEE 118-bus system on the average. We consider two types of objective function: the minimum total real power generation cost $\sum_{G_i} a_i P_{G_i}^2 + b_i P_{G_i} + c_i$ and the minimum system losses $\sum_l P_{loss}(l)$, where P_{G_i} denotes the real power generation of generation bus G_i, a_i, b_i and c_i are cost coefficients, and $P_{loss}(l)$ denotes the real power loss on transmission line l. The set of pair subsystems consisting of tie lines in the IEEE 118-bus and TP 244-bus systems denoted by M_{118} and M_{244}, respectively, are $M_{118} = \{(S_1, S_2), (S_1, S_4), (S_1, S_3), (S_2, S_3)\}$ and $M_{244} = \{(T_1, T_2), (T_1, T_3), (T_1, T_4), (T_2, T_3), (T_4, T_3)\}$. We

TABLE I
Contents of the Four Subsystems in the IEEE 118-Bus and TP 244-Bus Systems

System	IEEE-118 bus				TP-244 bus			
Subsystem	S_1	S_2	S_3	S_4	T_1	T_2	T_3	T_4
# of buses	32	28	26	32	63	57	63	61
# of lines	49	33	30	27	122	98	79	107
# of gen. buses	8	5	5	5	14	9	10	13

TABLE II
Two Sets of Discrete Control Variables in Each Subsystem of the IEEE 118-Bus and TP 244-Bus System

Case	IEEE 118-bus system								TP 244-bus system							
	D = 1				D = 2				D = 1				D = 2			
Subsystem	S_1	S_2	S_3	S_4	S_1	S_2	S_3	S_4	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4
# of Cap.	8	4	4	4	12	6	6	6	16	8	8	8	24	12	12	12
# of Trans.	8	4	4	4	12	6	6	6	16	8	8	8	24	12	12	12
r_i	16	8	8	8	24	12	12	12	32	16	16	16	48	24	24	24
n_i	4	2	2	2	12	3	3	3	8	4	4	4	12	6	6	6

TABLE III
Final Objective Value Obtained by and the Consumed CPU Time of the Distributed Algorithm II Implemented in 4-PC Network, the Corresponding Centralized Version, the Centralized GA, and Centralized TS Method Implemented in Single PC for the Eight Cases

Case	Objective value						CPU time (seconds)			
	Distri.	Centr. (I)	GA (II)	$\frac{\text{II}-\text{I}}{\text{I}}$	TS (III)	$\frac{\text{III}-\text{I}}{\text{I}}$	Distri.	Centr.	GA	TS
(1,1,1)	5694§	5694§	6595§	16%	6816§	20%	0.53	1.61	235	215
(1,1,2)	5670§	5670§	7167§	26%	7312§	29%	0.52	1.62	240	245
(1,2,1)	62.95†	62.95†	75.41†	20%	74.06†	18%	0.47	1.55	245	250
(1,2,2)	46.36†	46.36†	56.45†	22%	56.74†	22%	0.49	1.57	235	230
(2,1,1)	11949§	11949§	14974§	25%	15184§	27%	1.81	5.71	880	880
(2,1,2)	11887§	11887§	15275§	29%	15574§	31%	1.88	5.85	890	900
(2,2,1)	138.37†	138.37†	170.81†	23%	173.49†	25%	1.72	5.49	840	850
(2,2,2)	109.24†	109.24†	137.09†	25%	136.62†	25%	1.76	5.55	850	845

§: with unit in $\$/hr$. †: with unit in MW.

Distri.: Distributed Algorithm II implemented in 4-PC network.

Centr.: Centralized version of Distributed Algorithm II implemented in single PC.

assume each switching capacitor is equipped with four capacitor banks, and the capacity of a bank is 14 MVAR. We assume each transformer tap has 32 discrete steps such that each step is 5/8% of the nominal transformer tap ratio. We consider two sets of discrete control variables, namely $D = 1$ and $D = 2$, in each system and the number of switching capacitors and transformers in each subsystem for each set are shown in Table II. The values of r_i of each subsystem i can be easily calculated by adding the number of switching capacitors and transformers as shown in the fifth row of Table II, and we set $n_i = r_i/4$ for each subsystem i as shown in the last row of Table II.

For each system, each objective function and each discrete control variable set, we have tested eight cases, which are described by three arguments (S, O, D), such that $S = 1$ indicates the test system is the IEEE 118-bus system and 2 the TP 244-bus system, $O = 1$ indicates the objective function is total generation cost and 2 the total system losses, $D = 1$ indicates the first discrete control variable set and 2 the second discrete control variable set for the corresponding system as shown in Table II. These eight cases are shown in the first column of Table III. We apply Distributed Algorithm II to solve the DOPFD of these eight cases in the 4-PC network. The four PCs are of the same model: Pentium IV, 2.66-GHz processor and 1.25 GB of RAM. We assign subsystems S_1 and T_1 as the root subsystems of the IEEE 118-bus and TP 244-bus systems, respectively. The program is written in C. We employ the TCPIP as the communication protocol in the 4-PC network. We set $\varepsilon = 10^{-4}$ in Steps 9 and 12 in Distributed Algorithm I and $s = 50$ in Step 3 of Distributed Algorithm II. The final objective value of the overall system in each case obtained by the Distributed Algorithm II and corresponding CPU time consumed in the 4-PC network is shown in the second and the eighth columns in Table III. The consumed CPU time including the communication overhead in the 4-PC network is counted until the root subsystem determines the good enough discrete control solution and sends the corresponding subvector to each subsystem.

To illustrate the reduction of the search space of discrete control variables in our approach, we take the case (1,1,1) as an illustrative example. The size of the original search space U_d is $33^{(8+4+4+4)} \times 5^{(8+4+4+4)} (= 2.237 \times 10^{44})$, because each transformer tap has 32 discrete steps and each switching capacitor has four capacitor banks. After executing Step 1 of Distributed Algorithm II (i.e., part 1 of Stage 1), the size of search space is reduced from 2.237 $\times 10^{44}$ to $2^{(8+4+4+4+8+4+4+4)} (= 1.1 \times 10^{12})$. At this point, each discrete control variable $u_{d_{i,j}}$ has two choices of discrete values, $\lceil u^*_{c_{i,j}} \rceil$ or $\lfloor u^*_{c_{i,j}} \rfloor$. After executing Step 2 (i.e., part 2 of Stage 1), we found that the optimal objective value of the CDOPF (23) is very insensitive to the transform tap ratio due to two reasons: 1) the values of $\overset{0}{\bar{g}}_i(y^*_i)$ and $\bar{g}_{ib}(y^*_i, {y^i_{J_i b}}^*)$ are insensitive to the deviation of transformer tap ratio and 2) the discrete step of the transform tap ratio is very small. On the other hand, optimal objective value is more sensitive to capacitor due to larger discrete step unless the optimal continuous capacitor value is already close to one of the neighboring discrete values. Thus, all the transformer taps and half of the switching capacitors of each subsystem are fixed at the side of closest discrete value that achieves $|\Delta f^*(u_{c_{i,j}})|$ as defined in part 2 of Stage 1. At this point, we have further reduced the size of the search space from 1.1×10^{12} to $2^{(4+2+2+2)} (= 1024)$. After executing Step 3 (i.e., Stage 2), the best 50 out of the 1024 possible $u_d's$ resulted from Step 2 are selected based on the sensitivity model. At this point, we have further reduced the size of the search space from 1024 to 50. Applying the exact model to evaluate the 50 possible $u_d's$ resulted from Step 3, that is executing Steps 4 and 5 (i.e., Stage 3), the best u_d is the good enough discrete control variable solution. From this case, we see that if we apply the exhaustive search method to find the optimal u_d for (1), we need to solve 2.237 $\times 10^{44}$ CDOPFs. However, we only need to solve 51 CDOPFs to obtain a good enough u_d. Moreover, the 50 CDOPFs solved in Steps 4 and 5 take one iteration only. This manifests the dramatic computation time reduction in our approach.

To verify our results, we also implement the Distributed Algorithm II in single PC and apply to the eight cases. The final objective values shown in the third column of Table III are exactly the same as that obtained in the 4-PC network, and the consumed CPU times are shown in the ninth column.

This demonstrates that Distributed Algorithm II is successfully implemented in a computer network, and the consumed CPU time is less than one third but more than one fourth of the CPU time consumed by the centralized version. The reason that the Distributed Algorithm II is not four times faster than the centralized version when using the 4-PC network is because the sizes of the four subsystems are different, and there exists some but very slight communication overhead. It is worth noting that the good enough solutions we obtained in all the eight cases are *feasible*. This reflects the significantly improved probability of our approach in getting feasible discrete control solutions as commented in Remark 3. Although we cannot find any competing methods in dealing with the DOPFD considered in this paper so far, we can treat (1) as a centralized OPF with discrete control variables for all the eight cases and solve them using the global searching techniques GA and TS methods associated with the combination of SQP with the DPQN methods to solve the centralized CDOPF. In the employed GA [26], we use a simple coding scheme of 0 and 1 strings to represent all possible $u_d's$ in U_d, and each u_d represents a population in GA. We randomly select 20 $u_d's$ from U_d as our initial populations. The fitness of a population u_d is set to be the reciprocal of the objective value of (1), in which u_d is set to be the population u_d, and is solved by the combination of the SQP and DPQN methods in single PC. The members in the mating pool are selected from the pool of populations using roulette wheel selection scheme based on the fitness values. We set the probability of selecting members in the mating pool to serve as parents for crossover to be $p_c = 0.7$. We use a single point crossover scheme and assume the mutation probability to be $p_m = 0.02$. For each of the eight cases, we stop GA when it consumes around 150 times of the CPU time consumed by the centralized version of Distributed Algorithm II and record the best so far objective values and the consumed CPU time in the fourth and the tenth columns of Table III. The iterative mechanism of the employed TS method is stated in the following. Starting from a randomly selected u_d from U_d, in each iteration of the TS method, we randomly evaluate one third of the neighboring $u'_d s$ of the current u_d and accept the best one to be the new u_d based on the tabu list and a criterion of global aspiration by objective [26]. Noting that evaluating the objective value of (1) for a given u_d in the TS method is the same as in GA, we apply TS method to all eight cases. For each of the eight cases, we also stop the method when it consumes around 150 times of the CPU time consumed by the centralized Distributed Algorithm II and record the best so far objective value and the consumed CPU times in the sixth and the eleventh columns in Table III. From the fourth and sixth columns of Table III, we see that in most of the cases, GA outperforms TS method, because of its capability of decentralization. From the fifth and seventh columns, we can observe that when the number of discrete control variables increases in the same system for the same objective function, the performance of both GA and TS methods degrade, because the improvement of the best so far objective values becomes more sluggish. Furthermore from the third, fifth, seventh, ninth, tenth and eleventh columns, we find that when both GA and TS methods consumed around 150 times of the CPU time consumed by the centralized version of Distributed Algorithm II, their best so far objective values are still 23.25% and 24.63%, on the average, more than the objective values obtained by the centralized version of

Distributed Algorithm II, respectively. On the other hand, using less than one third of the CPU time, the Distributed Algorithm II implemented in 4-PC network can obtain the same objective value as that obtained by the centralized version. These indirect comparisons demonstrate the computational efficiency of Distributed Algorithm II and the goodness of the obtained good enough solutions. The feasibility and the goodness of the obtained good enough solutions in all eight cases confirm the robustness of our algorithm.

V. Conclusion

In this paper, we have proposed a distributed algorithm to deal with DOPFD of large distributed power systems. We use a 4-PC network to implement the proposed distributed algorithm and apply to the DOPFD on the IEEE 118-bus and TP 244-bus systems. We have ascertained the robustness of our distributed algorithm in the aspects of feasibility and goodness of the obtained solution; moreover, the computational speed of our distributed algorithm is at least three times faster than the centralized version in all the test cases when using a 4-PC network.

Acknowledgment

The authors would like to thank the anonymous reviewers who gave them constructive comments and helpful suggestions to improve the readability of their paper. The authors also would like to thank Mr. C.-Z. Lin for his efforts in setting up the PC network, establishing the communication between PCs and typing this paper.

References

[1] B. Stott, J. L. Marinho, and O. Alsac, "Review of linear programming applied to power system rescheduling," in *Proc. IEEE PICA Conf.*, 1979, pp. 142–154.

[2] T. C. Giras and S. N. Talukdar, "Quasi-Newton method for optimal power flows," *Int. J. Elect. Power Energy Syst.*, vol. 3, no. 2, pp. 59–64, Apr. 1981.

[3] R. C. Burchett, H. H. Happ, and D. R. Vierath, "Quadratically convergent optimal power flow," *IEEE Trans. Power App. Syst.*, vol. PAS-103, no. 10, pp. 2864–2880, Oct. 1984.

[4] D. I. Sun, I. Hu, G. Lin, C. J. Lin, and C. M. Chen, "Experiences with implementing optimal power flow for reactive scheduling in the Taiwan power system," *IEEE Trans. Power Syst.*, vol. 3, no. 3, pp. 1193–1200, Aug. 1988.

[5] Y. C. Wu, A. S. Debs, and R. E. Marsten, "A direct nonlinear predictor-corrector primal-dual interior point algorithm for optimal power flows," *IEEE Trans. Power Syst.*, vol. 9, no. 2, pp. 876–883, May 1994.

[6] B. H. Kim and R. Baldick, "Coarse-grained distributed optimal power flow," *IEEE Trans. Power Syst.*, vol. 12, no. 2, pp. 932–939, May 1997.

[7] B. H. Kim and R. Baldick, "A comparison of distributed optimal power flow algorithms," *IEEE Trans. Power Syst.*, vol. 15, no. 2, pp. 599–604, May 2000.

[8] D. Hur, J.-K. Park, and B. H. Kim, "Evaluation of convergence rate in the auxiliary problem principle distributed optimal power flow," *Proc. Inst. Elect. Eng., Gen., Transm., Distrib.*, vol. 149, no. 5, pp. 525–532, Sep. 2002.

[9] D. Hur, J.-K. Park, B. H. Kim, and K.-M. son, "Security constrained optimal power flow for the evaluation of transmission capability on korea electric power system," in *Proc. IEEE Power Eng. Soc. Summer Meeting*, 2001, vol. 2, pp. 1133–1138.

[10] F. J. Nogales, F. J. Prieto, and A. J. Conejo, "A decomposition methodology applied to the multi-area optimal power flow problem," *Ann. Oper. Res.*, vol. 120, no. 1-4, pp. 99–116, Apr. 2003.

[11] S.-S. Lin and H. Chang, "An efficient algorithm for solving BCOP and implementation," *IEEE Trans. Power Syst.*, vol. 22, no. 1, pp. 275–284, Feb. 2007.

[12] H. Chang and S.-S. Lin, "A MPBSG technique based parallel dual-type method for solving distributed optimal power flow problems," *IEICE Trans. Fundam. Electron., Commun., Comp. Sci.*, pp. 260–269, Jan. 2006.

[13] A. Bakirtzis and A. Meliopoulos, "Incorporation of switching operation in power system corrective control computations," *IEEE Trans. Power Syst.*, vol. 2, no. 3, pp. 669–676, Aug. 1987.

[14] A. Monticelli and W. Liu, "Adaptive movement penalty method for the Newton optimal power flow," *IEEE Trans. Power Syst.*, vol. 7, no. 1, pp. 334–340, Feb. 1992.

[15] W. Liu, A. Papalexopoulos, and W. Tinney, "Discrete shunt controls in a Newton optimal power flow," *IEEE Trans. Power Syst.*, vol. 7, no. 4, pp. 1509–1520, Nov. 1992.

[16] W. Tinney, J. Bright, K. Demaree, and B. Hughes, "Some deficiencies in optimal power flow," in *Proc. IEEE PICA Conf.*, May 1987, pp. 164–169.

[17] S.-Y. Lin, Y.-C. Ho, and C.-H. Lin, "An ordinal optimization theory based algorithm for solving the optimal power flow problem with discrete control variables," *IEEE Trans. Power Syst.*, vol. 19, no. 1, pp. 276–286, Feb. 2004.

[18] L. Chen, H. Suzuki, and K. Katou, "Mean field theory for optimal power flow," *IEEE Trans. Power Syst.*, vol. 12, no. 4, pp. 1481–1486, Nov. 1997.

[19] A. Bakirtzis, P. Biskas, C. Zoumas, and V. Petridis, "Optimal power flow by enhanced genetic algorithm," *IEEE Trans. Power Syst.*, vol. 17, no. 2, pp. 229–236, May 2002.

[20] T. Kulworawanichpong and S. Sujitjorn, "Optimal power flow using tabu search," *IEEE Power Eng. Rev.*, vol. 22, no. 6, pp. 37–40, Jun. 2002.

[21] J. T. Ma and L. L. Lai, "Evolutionary programming approach to reactive power planning," *Proc. Inst. Elect. Eng., Gen., Transm., Distrib.*, vol. 143, no. 4, pp. 365–370, Jul. 1996.

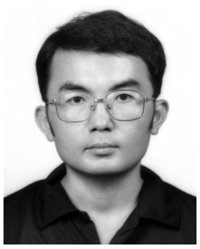
[22] Y. C. Ho, *Soft Optimization for Hard Problem*. Cambridge, MA: Harvard Univ. Press, 1996, Lecture Notes.

[23] T. W. E. Lau and Y.-C. Ho, "Universal alignment probability, and subset selection for ordinal optimization," *J. Optim. Theory Appl.*, vol. 93, no. 3, pp. 455–489, Jun. 1997.

[24] C.-H. Lin and S.-Y. Lin, "A new dual-type method used in solving optimal power flow problems," *IEEE Trans. Power Syst.*, vol. 12, no. 4, pp. 1667–1675, Nov. 1997.

[25] D. Luenberger, *Linear and Nonlinear Programming*, 2nd ed. Reading, MA: Addison-Wesley, 1984.

[26] S. Sait and H. Youssef, *Iterative Computer Algorithms With Application in Engineering: Solving Combinatorial Optimization Problems*. Los Alamitos, CA: IEEE Computer Society, 1999.



Ch'i-Hsin Lin was born in Taiwan, R.O.C. He received the B.S. degree in electrical engineering from Feng Chia University, Taichung, Taiwan, the M.S. degree in electrical engineering from National Tsing Hua University, Hsinchu, Taiwan, and Ph.D. degree in electrical and control engineering from Chiao Tung University, Hsinchu, in 1989, 1991, and 1996, respectively.

He joined the Department of Electronic Engineering at the Kao Yuan University, Kaoshiung, Taiwan, R.O.C., in 1998 and has been a Associate Professor since 2003. His major research interests include large-scale power systems and ordinal optimization theory and applications.

Shin-Yeu Lin was born in Taiwan, R.O.C. He received the B.S. degree in electronics engineering from National Chiao Tung University, Hsinchu. Taiwan, R.O.C., the M.S. degree in electrical engineering from University of Texas at El Paso, and the D.Sc. degree in systems science and mathematics from Washington University, St. Louis, MO, in 1975, 1979, and 1983, respectively.

From 1984 to 1985, he was with Washington University working first as a Research Associate and then a Visiting Assistant Professor. From 1985 to 1986, he was with GTE Laboratory working as a Senior MTS. He joined the Department of Electrical and Control Engineering at National Chiao Tung University in 1987 and has been a Professor since 1992. His major research interests include optimal power flow, ordinal optimization theory and applications, and distributed computations.