ABSTRACT
In this paper, we propose a distributed computational algorithm for solving the distributed optimal power flow problem under deregulated environment. Except for the data of the self-subsystem, the proposed distributed algorithm requires only the data of the boundary buses of the connecting subsystems. Therefore, it is easily implemented in a computer network. We have tested the proposed distributed algorithm on the IEEE 118-bus system, which is arbitrarily partitioned into four subsystems. The test results demonstrate the convergence and the computational efficiency of the proposed distributed algorithm.

KEY WORDS
Distributed optimal power flow, distributed computation, nonlinear programming, power deregulation.

1. Introduction
Power deregulation is a trend in recent years and pushes the power market to be very active. Various research issues are then raised under the deregulated environment; among them, Distributed Optimal Power Flow (DOPF) is one of the most important subjects.

Although the Optimal Power Flow (OPF) problem has a long history in power system research [1-5], the study of DOPF is introduced only recently. Kim and Baldick proposed a course-grained DOPF algorithm in [6], and they also compared three decomposition coordination methods for implementing DOPF algorithm in [7]. Hur et al. had evaluated the convergence rate of the auxiliary problem principle for DOPF algorithm in [8]. In a more recent paper [9], Hur et al. consider the security constraints for DOPF.

In this paper, we consider the DOPF in deregulated environment such that each subsystem will buy (sell) power from (to) the neighboring subsystems if they fall short of (have surplus) power. Since the power flow from the neighboring subsystem is set by the contract, each subsystem would utilize the generated and purchased power in an optimal way. However, to achieve a system-wise optimality, all the subsystems have to cooperate to solve the DOPF in a whole.

From the hardware viewpoint, DOPF is easier to implement nowadays, because the mature computer network technology and the prevailed network infrastructure. Thus, what is most needed is a reliable distributed computation software. Despite the existence of the DOPF algorithms mentioned above, the DOPF problem considered here is different from those in [6-9], because the tie line flows should be equality constraints in our formulation. Therefore, we need to propose a DOPF algorithm that is suitable for the environment of deregulated power systems.

The paper is organized in the following manner. In Section II, we will state the considered DOPF mathematically. In Section III, we will propose a solution method to solve the considered DOPF. In Section IV, we will present distributed algorithmic steps to execute the proposed solution method in the computer network. In Section V, we will test our distributed algorithm on the IEEE 118-bus system, which is arbitrarily partitioned into 4 subsystems. Finally, we make a conclusion in Section VI.

2. Problem Statement
Since the amount of power delivered from the selling subsystem is fixed and set by the contract. Thus, the power flow of the tie lines from the selling subsystem to the buying subsystem is considered to be
the additional line flow constraints in the DOPF. Thus, our DOPF can be stated in the following:

\[
\min \sum_{i=1}^{N} c_i(P_{G_i})
\]

subject to

\[
\begin{align*}
  g_i(x_i, x_{i,b}, P_{G_i}, Q_{G_i}, x_{i,e}) &= 0 \\
  h(x_i, x_{i,b}, P_{G_i}, Q_{G_i}) &\leq 0, i = 1, \ldots, N \\
  P_{ij}(x_{i,b}, x_{j,b}) &= p_{ij} \\
  Q_{ij}(x_{i,b}, x_{j,b}) &= q_{ij}, \quad \forall (i, j) \in M
\end{align*}
\]  

(1)

where \( c_i(P_{G_i}) \) denote the total generation cost function of subsystem \( i \), \( x_i = (x_{i,b}, x_{i,e}) \) denote the vector of complex voltage of subsystem \( i \), where \( x_{i,b} \) and \( x_{i,e} \) denote the vector of complex voltage of interior buses and boundary buses, respectively; \( g(x_i, x_{i,b}, P_{G_i}, Q_{G_i}, x_{i,e}) = 0 \) denotes the flow balance equations and \( h(x_i, x_{i,b}, P_{G_i}, Q_{G_i}) \leq 0 \) denotes the security constraints such as the thermal limit, voltage magnitude limit, power generate limit of subsystem \( i \); \( x_{i,e} \) denotes the vector of complex voltage of the boundary buses of other subsystems connecting with subsystem \( i \); \( N \) denotes the total number of subsystems in the deregulated power network, and \( P_{ij}(x_{i,b}, x_{j,b}) = p_{ij} \) denotes the real power line flow equation from subsystem \( i \) to \( j \), where \( p_{ij} \) denote the amount of power purchased by subsystem \( j \) from \( i \); \( Q_{ij}(x_{i,b}, x_{j,b}) = q_{ij} \), denote the corresponding reactive power flow of the purchased real power \( p_{ij} \) with reasonal power factor; \( x_{i,e} \) denotes the vector of complex voltage of the boundary bus of subsystem \( j \) such that \( (i, j) \in M \); \( M = \{(i, j)\}'s \) denotes the set of selling and purchasing pairs \( (i, j) \) of power companies \( i \) and \( j \).

We assume each subsystem has its own control center, which is equipped with a computer. The computers of all subsystems are connected through a computer network. The proposed distributed algorithm for solving the DOPF (1) will be carried out in this network.

3. The Solution Method

In order to achieve the system-wise optimal objective, we have to solve the DOPF problem shown in (1) in a whole using a method that can be computed distributedly. In our previous research work, we have developed the DPPQN method based algorithm to solve the centralized OPF problems \([10-12]\) for quite a while. This algorithm gains its computational efficiency and stability by its decomposition effects and the capability of handing the binding inequality constraints. As a matter of fact, the decomposition effects of the DPPQN method can play the key for distributed computation. Thus, we will describe how the DPPQN method solves the DOPF (1) in the following:

First of all, we define the vector functions \( \mathbf{\overline{g}}_i = (g_i, P_{ij}, Q_{ij}) \) and partition \( \mathbf{\overline{g}}_i \) into \( (\mathbf{\overline{g}}_i, \mathbf{\overline{g}}_{i,b}) \) such that the sub-vector function \( \mathbf{\overline{g}}_i \) involves only the variable vector \( (x_i, x_{i,b}, P_{G_i}, Q_{G_i}) \) of subsystem \( i \), while \( \mathbf{\overline{g}}_{i,b} \) involves not only the variable vector of subsystem \( i \) but also the variable vector \( x_{i,e} \) of the subsystems connecting with subsystem \( i \). Using this notation, we can rewrite (1) as

\[
\min \sum_{i=1}^{N} c_i(P_{G_i})
\]

subject to

\[
\begin{align*}
  \mathbf{\overline{g}}_i(x_i, x_{i,b}, P_{G_i}, Q_{G_i}) &= 0 \\
  \mathbf{\overline{g}}_{i,b}(x_i, x_{i,b}, P_{G_i}, Q_{G_i}, x_{i,e}) &= 0 \\
  h_i(x_i, x_{i,b}, P_{G_i}, Q_{G_i}) &\leq 0, i = 1, \ldots, N
\end{align*}
\]

(2)

For the sake of notational simplicity, we define the following variable vector \( y = (y_1, \ldots, y_N) \) where

\[
y_i = (y_i, x_{i,b}) \quad \text{and} \quad y_i = (x_i, P_{G_i}, Q_{G_i}).
\]

The proposed DPPQN method based algorithm is a combination of the Successive Quadratic Programming (SQP) method with the Dual Projected Pseudo Quasi Newton (DPPQN) method such that the Quadratic Programming Problem (QPP) induced in the SQP method is solved by the DPPQN method. The SQP method uses the following iterations to solve (2)
\[ y(k+1) = y(k) + \alpha(k) \Delta y(k) \]  

where \( k \) is the iteration index, \( \alpha(k) \) is a positive step-size and \( \Delta y(k) = (\Delta y_1(k), \ldots, \Delta y_N(k)) \), in which \( \Delta y_i(k) = (\Delta y_i(k), \Delta x_{i,b}(k)) \) and \( \Delta y(k) = (\Delta x_1(k), \Delta P_{G_i}(k), \Delta Q_{G_i}(k)) \). \( \Delta y(k) \) is the solution of the following.

QPP:

\[
\begin{align*}
\min_{\Delta y} & \sum_{i=1}^{N} \Delta P_{G_i}^T \frac{\partial^2 c_i(k)}{\partial P_{G_i}^2} \Delta P_{G_i} + \frac{\partial c_i(k)}{\partial P_{G_i}} \Delta P_{G_i} \\
\text{subject to} & \\
\end{align*}
\]

\[
\begin{align*}
\frac{\partial g_i(k)}{\partial y_j} + \frac{\partial g_{i,b}(k)}{\partial y_j} \Delta y_j + \frac{\partial g_{i,b}(k)}{\partial x_{i,b}} \Delta x_{i,b} = 0 \\
\frac{\partial g_{i,b}(k)}{\partial x_{i,b}} \Delta x_{i,b} = 0 \\
\frac{\partial h_i(k)}{\partial y_j} \Delta y_j + \frac{\partial h_{i,b}(k)}{\partial x_{i,b}} \Delta x_{i,b} \leq 0 \\
\end{align*}
\]

(4)

where \( g_i(k) \) represents \( g_i(y(k), x_{i,b}(k)) \) and the same abbreviation applies to \( g_{i,b}(k) \) and \( h_i(k) \). The QPP in (4) will be solved by the DPPQN method. Thus, most of the computations of the proposed algorithm lie on the DPPQN method to solve (4) for \( \Delta y(k) \), because the SQP method simply update \( y(k) \) by (3) once \( \Delta y(k) \) is obtained in each iteration.

Instead of solving (4) directly, the DPPQN solves the dual problem of (4), which is stated below

\[
\max \phi(\lambda) 
\]

(5)

where the dual function

\[
\phi(\lambda) = 
\]

\[
\min_{\Delta y \in \Omega(k)} \sum_{i=1}^{N} \Delta P_{G_i}^T \frac{\partial^2 c_i(k)}{\partial P_{G_i}^2} \Delta P_{G_i} + \frac{\partial c_i(k)}{\partial P_{G_i}} \Delta P_{G_i} 
\]

\[
+ \sum_{i=1}^{N} \lambda_i \left[ \frac{\partial g_i(k)}{\partial y_i} \Delta y_i + \frac{\partial g_{i,b}(k)}{\partial x_{i,b}} \Delta x_{i,b} \right] 
\]

\[
+ \sum_{i=1}^{N} \lambda_{i,b}^T \left[ \frac{\partial g_{i,b}(k)}{\partial x_{i,b}} \Delta x_{i,b} \right] 
\]

(6)

in which the constraint set

\[
\Omega(k) = \left\{ \Delta y \mid \frac{\partial h_i(k)}{\partial y_j} \Delta y_j + \frac{\partial h_{i,b}(k)}{\partial x_{i,b}} \Delta x_{i,b} \leq 0 \\
i = 1, \ldots, N \right\} 
\]

(7)

The DPPQN method uses the following iterations to solve the dual problem (5)

\[
\lambda(t+1) = \lambda(t) + \beta(t) \Delta \lambda(t) 
\]

(8)

where \( t \) is the iteration index; \( \beta(t) \) is a positive step-size; \( \lambda(t) = (\lambda_1(t), \ldots, \lambda_N(t)) \) in which \( \lambda_i(t) = (\lambda_i(t), \lambda_{i,b}(t)) \), where \( \lambda_i(t) \) corresponds to the equality constraints related with subsystem \( i \) only, while \( \lambda_{i,b}(t) \) corresponds the equality constraints involving \( x_{i,b} \); \( \Delta \lambda(t) = (\Delta \lambda_1(t), \ldots, \Delta \lambda_N(t)) \) denotes the increment of \( \lambda(t) \), where \( \Delta \lambda_i(t) = (\Delta \lambda_i(t), \Delta \lambda_{i,b}(t)) \).

The \( \Delta \lambda(t) \) in (8) can be obtained by solving the following linear equations.

\[
\Phi \Delta \lambda(t) + \frac{\partial \phi(\lambda(t))}{\partial \lambda} = 0 
\]

(9)

where the block diagonal matrix \( \Phi = diag(\Phi_1, \ldots, \Phi_N) \), and the \( i \)th diagonal block sub-matrix

\[
\Phi_i = \begin{bmatrix} \Phi_{i00} & \Phi_{i0b} \\ \Phi_{i0b} & \Phi_{ibb} \end{bmatrix} 
\]

(10)
The \( (\Delta y_i, \Delta \hat{x}_{j,b}) \), \( i = 1, ..., N \) required in (17) and (18) for \( i = 1, ..., N \) is the optimal solution, \( \Delta \hat{y}_i \), of the minimization problem on the RHS of (6), which can be solved by solving \( N \) independent minimization subproblems as follows. For the given \( \lambda(t) = (\lambda_1(t), ..., \lambda_N(t)) \), the minimization problem on the RHS of (6) can be decomposed into the following \( N \) independent minimization subproblems:

\[
\min_{\Delta y_i, \Delta \hat{x}_{j,b}} \Delta P_{G_i}^T \frac{\partial^2 C_i(k)}{\partial P_{G_i}^2} \Delta P_{G_i} + \frac{\partial C_i(k)}{\partial P_{G_i}} P_{G_i} + \dot{\lambda}_i \left[ \frac{\partial}{\partial y_i} (\lambda_k(k)) \right] \frac{\partial}{\partial y_i} \Delta y_i + \frac{\partial}{\partial y_i} \Delta \hat{x}_{j,b} \left[ \frac{\partial}{\partial y_i} \Delta \hat{x}_{j,b} \right] \leq 0
\]

where

\[
\Omega_i(k) = \left\{ \Delta y_i \left[ \frac{\partial h_i(k)}{\partial y_i} \Delta y_i + \frac{\partial h_i(k)}{\partial y_i} \Delta \hat{x}_{j,b} \leq 0 \right] \right\}
\]

and \( \Omega(k) = \bigcup_{i=1}^N \Omega_i(k) \). \( J_i \) denotes the index set of subsystems connecting with subsystem \( i \), and \( \lambda_{j,b} \) denotes the Lagrange multipliers associated with the equality constraints of subsystem \( j \) involving the complex voltage of the boundary buses of subsystem \( i \). Once the optimal solution of (19) is obtained, we can calculate \( \frac{\partial \phi(\lambda(t))}{\partial \lambda} \) by (17) and (18). Then we can solve \( \Delta \lambda(t) \) from (9) by first decomposing (9) into the following \( N \) independent set of linear equations,

\[
\Phi_i \Delta \lambda_i(t) + \frac{\partial \phi(\lambda(t))}{\partial \lambda} = 0, \ i = 1, ..., N
\]
method will continue until they converge to an optimal \( \lambda^* \) for the dual problem (5). The corresponding optimal solution on the RHS of (6) when \( \lambda = \lambda^* \) will be \( (\Delta y_i(k), \Delta x_{i,j}(k)), i = 1, \ldots, N \) which will be used to update \( y(k+1) \) in (3) using an Armijo-type step-size \( \alpha(k) \) [10]. Then we will start the next iteration of the SQP method. The iterations of the SQP method will continue until it converges to the optimal solution of the DOPF (1).

4. The Distributed Algorithm

From the computational formulae of the SQP method and the DPPQN method described above, we see that the SQP method does nothing but update \( y(k) \) in each iteration as shown in (3), which of course can be carried out in individual subsystems. The major computations required in the DPPQN method are solving the optimization problems (19) and the linear equations (20), which are already decomposed such that each subsystem can execute its part of the computations as long as the necessary data from the connecting subsystems are passed through the computer network. This indicates that the proposed solution method is very suitable for implementation in a distributed computer network.

In order to govern the synchronization of the convergence of the DPPQN and SQP methods in the distributed computing network, we assign a root sub-system among the connecting sub-systems to be responsible for this task. Following is the distributed algorithmic steps for each subsystem \( i \):

Step 0: Initially guess \( y_i \); initially guess \( \lambda_i \).
Step 1: Send \( \lambda_{i,b} \) to connecting subsystems.
Step 2: Once receiving all \( \lambda_{j,b}, j \in J_i \) from all connecting subsystems, compute \( \Delta\hat{y}_i \) from solving (19).
Step 3: Send \( \Delta\hat{y}_i \) to connecting subsystems.
Step 4: Once all \( \Delta x_{j,b}, j \in J_i \) are received, calculate \( \frac{\partial \phi(\lambda)}{\partial \lambda_j} = \left( \frac{\partial \phi(\lambda)}{\partial \lambda_j}, \frac{\partial \phi(\lambda)}{\partial \lambda_{j,b}} \right) \) by (17) and (18).
Step 5: Compute \( \Phi_i \) and solve \( \Delta\lambda_i \) from (20).
Step 6: If \( \|\Delta\lambda_i\| < \varepsilon \), send a signal to the root subsystem to inform the convergence of DPPQN method in this subsystem.
Step 7: Update \( \lambda \) by (8) and return to Step 1.
Step 8: Once receiving the signal to update \( y_i \) from the root subsystem, compute \( \Delta y_i \) from (19) and update \( y_i \) by (3). If \( \|\Delta y_i\| < \varepsilon \), send a signal to the root sub-system to inform the convergence of the SQP method in this subsystem and return to Step 1.

As indicated above, an extra task for the root subsystem in addition to the above distributed algorithmic execution steps is checking the system-wise convergence of the DPPQN and SQP methods. Thus, in Step 6, if the root subsystem receives the signal indicating the convergence of the DPPQN method from all subsystems, it will send a signal to all subsystems to update \( y \). Similarly, if the root subsystem receives the convergence signal of the SQP method from all subsystems, it will send a signal to all subsystems to stop the algorithm and output the solution.

5. Test Results

We have applied our distributed algorithm to solve the DOPF on the IEEE 118-bus system, which is arbitrarily partitioned into four subsystems. These four subsystems are indexed by A1, A2, A3 and A4, respectively. The interconnecting relationships of these four subsystems is shown in Fig. 1. The power selling and purchasing pairs are (A1,A2), (A1, A4), (A1, A3), and (A2,A3). The cost function of each generation bus in each subsystem is a quadratic function of the real power generation. We have arbitrarily assumed the amount of power sale in each selling and purchasing pair and distribute the power among the tie lines.

Since we do not have a distributed computing network at hand, we currently simulate our distributed algorithm in a Pentium IV PC. The final objective value we obtain is 46,414 dollars/hour, and the consumed sequential CPU time is 0.62 seconds. However, if we take the parallel computation effect into account but not including the data communication time, the CPU time for the longest subsystem is 0.24 seconds. To verify our result, we also solve the DOPF using the centralized DPPQN method based algorithm [10] and obtain the following results: the final objective value is 46,412 dollars/hour, and the consumed CPU time is 0.49 seconds. This demonstrate that our distributed algorithm does converge to the true solution, and the consumed
CPU time is much faster than the centralized method if we take the parallel computation effect into account.

6. Conclusion
Due to the trend of deregulation, DOPF becomes a focus in power system research. We have proposed a DOPF algorithm under deregulated environment in this paper. The communication requirement in the algorithm can be easily handled. The test results demonstrate the superiority of the proposed distributed algorithm.

7. Acknowledgements
This research work is supported by the National Science Council in Taiwan, R.O.C. under the grant NSC93-2218-E-009-036.

References

Figure 1. The interconnecting relationships between the four subsystems.