A parallel dual-type algorithm for a class of quadratic programming problems and applications

Shieh-Shing Lin a,*, Ch’i-Hsin Lin b, Shih-Cheng Horng c

a Department of Electrical Engineering, St. John’s University, 499, Section 4 Tam King Road, Tamsui, Taipei 25135, Taiwan, ROC
b Department of Electronics Engineering, Kao Yuan University, Kaohsiung, Taiwan, ROC
c Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung, Taiwan, ROC

ARTICLE INFO

Available online xxxx

Keywords:
Optimization
Parallel processing
Dual method
Quadratic programming problems
Power-system state estimation

ABSTRACT

In this paper, we present a parallel dual-type (PDT) algorithm for solving a strictly convex quadratic programming problem with equality and box constraints. The PDT algorithm is suitable for distributed implementation and can be used as a basic optimization module for handling optimization problems of large distributed systems. Besides, combining the proposed algorithm with a successive quadratic programming (SQP) method, we can solve constrained nonlinear programming problems such as power-system state estimation with power-flow balance constraints on no generation and no-load buses. We have demonstrated the computational efficiency of our method, by comparing with the benchmark commercial NCONF and QPROG routines and the state-of-the-art parallel algorithm through the implementation in the sequential version of Sparc workstation and the parallel version of PC network in solving constrained state estimation problems within IEEE 30-bus and IEEE 118-bus systems.

1. Introduction

Most practical network-type systems such as power systems, telephone networks and transportation systems... etc. are large-scale and formed by interconnected subsystems. For such kind of systems, operation management and state control are generally very difficult. The nonlinear constrained optimization problems in large scale interconnected system are computationally difficult because of their large dimension and nonlinearity. Conventionally, a central control center is employed to govern the operations of the whole system. Nowadays, since computer communication technologies have become more mature, decentralized management and control seems to be a trend. For example, they were applied on the current deregulated large power systems (Aghdam & Davison, 1998) formed by area-like subsystems interconnected through tier lines. Numerous decomposition techniques were listed in (Barlas & Veeravalli, 2005; Conejo, Castillo, Minguez, & Garcia-Bertrand, 2006; Cui, Xue, & Nahorst, 2006; Olariu, Schwing, & Zhang, 1992; Rodolakis, Siachalou, & Georgiadis, 2006) and there were many numerical techniques developed for solving these problems, such as the successive quadratic programming (SQP) methods (Burchett, Happ, & Vierath, 1985; Giras & Talukdar, 1981; Talukdar & Giras, 1982), the Lagrange Newton (LN) methods (Sun, Ashly, & Brewer, 1984; Sun, Hu, Lin, Lin, & Chen, 1988), the interior-point (IP) methods (D’Apuzzo & Marino, 2003; Torres & Quintana, 2001; Wu, 2001; Wu, Debs, & Marsten, 1994) and the Lagrange relaxation (LR) method (Biskas & Bakirtzis, 2005). The SQP methods have a good convergence rate but suffer from a dense Hessian matrix; the LN methods successfully exploit the sparse structure of network but encounter a coupling inequality constraint; the LR method belongs to gradient-type method which is easy to program but with slow convergence rate. Furthermore, there were some papers that concerned with the decentralized control algorithms, have been published (Aghdam & Davison, 1998; Conejo & Aguado, 1998; Dunn, Rossi, & Avramovic, 1995). And some methods used in the distributed control on the large scale distributed optimization problem (Baldick, Kim, Chase, & Luo, 1999; Bakirtzis & Biskas, 2003; Biskas & Bakirtzis, 2005), but those methods are based on conventional Lagrange method. They used Lagrange multiplier to relax the constraints and used the Karush–Kuhn–Tucker (KKT) condition to solve the problem. But if there are many coupling inequality constraints between the physical connections of two interconnected subsystems, these methods might become more complicated. Especially, it is difficult to determine the step-size of each individual subsystem in a large scale distributed computing environment.

Moreover, several papers that were concerned with decentralized control algorithms for maintaining the stability of the system have been published (Aghdam & Davison, 1998; Miller & Davison, 1990). However, there are few documents related to distributed algorithms for solving optimization problems of large-scale power systems. Although (Lin & Lin, 1994) have discussed distributed state estimator for power systems, the optimization problem considered in that paper is an unconstrained one. Presenting methods...
with a distributed form to handle a larger class of optimization problems for large distributed systems is the purpose of this paper. It is commonly accepted that handling a large-scale nonlinear constrained optimization problem with only a sequential computer is a difficult subject. Thus, to develop algorithms suitable for implementation in a distributed computer network, we start from an easier quadratic programming problem. Then, combining the proposed parallel quadratic programming technique with a successive quadratic programming method, we can solve nonlinear constrained optimization problems.

In this paper, we assume the considered large-scale system is strongly connected and is formed by \( n \) interconnected subsystems. The system shown in Fig. 1 is an example formed by four interconnected subsystems, in which we denote a state such as the complex voltage of a power network bus by a single bar, and use a line connecting two bars to represent the physical connection such as a transmission line. Let \( x_i \) and \( N_i \) denote the state and the dimension of the subsystem \( i \), respectively. Let \( x_i \) denote the vector of boundary states of subsystem \( i \), and \( x_{ib} \) denotes the vector of boundary states of the other subsystems connecting with subsystem \( i \). It should be noticed that \( x_i \) is a subvector of \( x \).

A function \( f(x) \) is said to be block additive provided that \( f(x) \) can be expressed as \( \sum_{i=1}^{n} f_i(x_i, x_{ib}) \), where the function \( f_i \) denotes a sum of functions, of which each function depends on either states \( x_i \) or on \( x_i \) and \( x_{ib} \). For example, \( f_i(x, x_{ib}) \) can be the sum of measurement error functions of subsystem \( i \) in a power-system state estimation problem: the measurements are such as the real and reactive power transmission line flows which will either contain states in subsystem \( i \) only or also contain boundary state of neighboring subsystems connected by a transmission line.

A class of quadratic programming problems (QPPs) of the following form is the target problem of this paper:

\[
\min_{x} \sum_{i=1}^{n} \frac{1}{2} \left[ x_i^T \begin{bmatrix} H_{ib} & H_{ib} \\ H_{ib} & 0 \end{bmatrix} x_i + \begin{bmatrix} r_i \\ r_{ib} \end{bmatrix}^T x_i \right] \tag{1}
\]

subject to

\[
A_i x_i = b_i, \quad i = 1, \ldots, n,
\]

\[
x_i \leq \bar{x}_i, \quad i = 1, \ldots, n,
\]

where the matrix \( \begin{bmatrix} H_{ib} \\ H_{ib} \\ 0 \end{bmatrix} \) has dimension \((N_i + N_{ib}) \times (N_i + N_{ib})\), in which \( N_i \) is the dimension of \( x_i \), and \( \begin{bmatrix} r_i \\ r_{ib} \end{bmatrix} \) is an \((N_i + N_{ib})\)-dimensional constant vector, \( A_i \) is an \( m_i \times N_i \) full-rank real matrix, and \( b_i \) is an \( m_i \)-dimensional constant vector. Eq. (2) denotes the equality constraints of subsystem \( i \), and inequality constraints Eq. (3) denote the bounded constraints on states such that the vectors \( x_i \) and \( \bar{x}_i \) denotes the upper and lower bound of each subsystem’s states \( x_i \) respectively.

An example system formed by four interconnected subsystems is shown in Fig. 1. An example system formed by four interconnected subsystems.

Fig. 1. An example system formed by four interconnected subsystems.
solution processes, adding or dropping the inequality constraints causes unexpected computational burden. For example, when \( \mu_i < 0 \) for the \( j \)th active inequality constraint, say \( x^j_i = x^j_k \) at the end of certain iteration, and \( |\mu_j| < \epsilon_1 \), where \( \epsilon_1 \) is a very small positive real. Theoretically, we should drop the index \( j \) from the set of active indices \( A \) (Luenberger, 1984). However, it is very likely that \( x^j_i > x^j_k \), and \( |x^j_i - x^j_k| < \epsilon_2 \), a very small positive real, at the next iteration. Consequently, we have to add the index \( j \) to \( A \) (Luenberger, 1984). Thus, for the inequality constraints holding on the margin of the bound, a frequent adding or dropping the active indices from \( A \) will occur and cause extra computational burden especially in the large system in which the each iteration requires considerable computation. However, in our dual problem formulation, we do not relax the bounded inequality constraints by another KKT multiplier vector \( \mu \) so as to circumvent the above mentioned cumbersome binding inequality constraints problem (Lin & Lin, 1997a, 1997b).

2.2. The parallel dual-type method

The proposed dual-type method for solving Eq. (4) uses the following iterations:

\[
\lambda_i(t + 1) = \lambda_i(t) + \alpha(t)\Delta\lambda_i(t), \quad i = 1, \ldots, n,
\]

where \( t \) is the iteration index, the positive scalar \( \alpha(t) \) is a step-size, and the increment of Lagrange multiplier vector \( \Delta \lambda_i(t) = (\Delta\lambda_1(t), \ldots, \Delta\lambda_n(t)) \) is the solution of the following quadratic approximate problem of Eq. (4) at \( \lambda(t) \):

\[
\max_{\Delta\lambda} \frac{1}{2} \lambda^T H \Delta\lambda + \nabla \Phi(\lambda(t)) \Delta\lambda.
\]

(9)

The matrix \( \Phi \) in Eq. (9) has dimension \( m \times m \),

\[
\Phi = \begin{bmatrix}
\Phi_1 & 0 & \cdots & 0 \\
0 & \Phi_2 & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
0 & \cdots & \cdots & \Phi_m
\end{bmatrix},
\]

(10)

where the \( m \times m \) block submatrix \( \Phi_i \) is chosen as

\[
\Phi_i = -A_i H_i A_i^T,
\]

(11)

in which \( H_i \) is the upper left submatrix shown in Eq. (1). Note that \( H_i \) is the \( ith \) diagonal block submatrix of \( H \) corresponding to subsystem \( i \) and this is a property of the block additive objective function in Eq. (1). We recall that \( H \) is positive definite, which is assumed in Section 1, thus its \( ith \) block diagonal submatrix \( H_i \) is also positive definite for every \( i \).

Remark 2. The chosen matrix \( \Phi \) defined in Eqs. (10) and (11) is intended to achieve two goals. The first goal is the decomposition effect so that we set \( \Phi \) to be a block diagonal form. The second goal is to achieve a better convergence rate than steepest descent, which choose \( \Phi = I \), and still can, under the decomposition premise, make \( \Phi \) approximate the Hessian of \( \phi(\lambda) \) as much as possible in the following aspect. The Hessian of \( \phi(\lambda) \) is \(-A H A^T \), where

\[
A = \begin{bmatrix}
A_1 & 0 & \cdots & 0 \\
0 & A_2 & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
0 & \cdots & \cdots & A_m
\end{bmatrix}.
\]

However, the \( \Phi \) we choose in Eqs. (10) and (11) is \(-A H A^T \), where \( H = \begin{bmatrix} H_{11} & 0 & \cdots & 0 \\
0 & H_{22} & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
0 & \cdots & \cdots & H_{mm} \end{bmatrix} \) is formed by the block diagonal submatrices of \( H \). Note that the second goal can be proved by applying the Modified Newton Method Theorem (Luenberger, 1984).

According to (Luenberger, 1984), the \( \nabla \phi(\lambda(t)) \) in Eq. (9) can be computed by

\[
\nabla \phi(\lambda(t)) = \begin{bmatrix}
A_1 & 0 & \cdots & 0 \\
0 & A_2 & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
0 & \cdots & \cdots & A_m
\end{bmatrix} \begin{bmatrix}
\dot{x}_1(\lambda(t)) \\
\vdots \\
\dot{x}_n(\lambda(t))
\end{bmatrix} - \begin{bmatrix}
b_1 \\
\vdots \\
b_n
\end{bmatrix},
\]

(12)

where \( \dot{x}(\lambda(t)) = (\dot{x}_1(\lambda(t)), \ldots, \dot{x}_n(\lambda(t))) \), is the solution of the minimization problem on the RHS of Eq. (5) when \( \lambda = \lambda(t) \). Furthermore, Eq. (12) can be decomposed into the following \( n \) independent sets of linear equations:

\[
\nabla \dot{x}_i(\lambda(t)) = A_i \dot{x}_i(\lambda(t)) - b_i, \quad i = 1, \ldots, n.
\]

(13)

Decomposition effect. Since \( H_i \) is positive definite, and \( A_i \) is of full rank, the \( \Phi \), given in Eq. (11) as well as the \( \Phi \) given in Eq. (10) is negative definite. Therefore, the objective function in Eq. (9) is concave. Consequently, \( \Delta \lambda(t) \), the solution of the approximate quadratic dual problem Eq. (9) can be obtained by solving the following optimal necessary conditions of Eq. (9):

\[
\Phi \Delta \lambda(t) = -\nabla \phi(\lambda(t))
\]

(14)

which can be decomposed into the following \( n \) independent sets of linear equations

\[
\Phi \Delta \lambda_i(t) = -\nabla \phi(\lambda(t)), \quad i = 1, \ldots, n.
\]

(15)

These \( n \) sets of Eq. (15) can be executed in parallel in a distributed computer network provided that each \( \nabla \phi(\lambda(t)) \) is obtained. In fact, it is this decomposition effect that makes our dual-type method parallel. In addition, this decomposition effect also contributes to the computational efficiency of our algorithm.

2.2.1. Computation of \( \nabla \phi(\lambda(t)) \)

As shown in Eq. (13), to compute \( \nabla \phi(\lambda(t)) \), we need to solve the minimization problem on the RHS of Eq. (5) to obtain \( \dot{x}(\lambda(t)) = (\dot{x}_1(\lambda(t)), \ldots, \dot{x}_n(\lambda(t))) \) first. This can be achieved using the following two-phase algorithm (Lin & Lin, 1997a; Lin & Lin, 1997b):

Phase 1: Solve the following unconstrained minimization problem

\[
\min \sum_{i=1}^{n} \left[ \frac{1}{2} \lambda_i^T H_i \lambda_i + \lambda_i^T H_i x_i + \lambda_i^T H_i x_i + r_i^T x_i + r_i^T x_i + \lambda_i^T A_i \lambda - b_i \right]
\]

(16)

using a modified parallel block scaled gradient method (Lin & Lin, 2003) which is briefly described in the Section 2.2.2, to obtain a solution \( \dot{x}(\lambda(t)) = (\dot{x}_1(\lambda(t)), \ldots, \dot{x}_n(\lambda(t))) \). Note that the unconstrained minimization problem Eq. (16) is the minimization problem on the RHS of Eq. (5) but without the constraints \( x \in \Omega \).

Phase 2: Project \( \dot{x}(\lambda(t)) \), the solution obtained from Phase 1, onto \( \Omega \).

In Phase 2, the constraint set is \( \Omega = \cup^m_{i=1} \Omega_i \), where \( \Omega_i = \{ x_i | x_i \leq \dot{x}_i \leq x_i \} \), and the resulting projection \( \dot{x}(\lambda(t)) = (\dot{x}_1(\lambda(t)), \ldots, \dot{x}_n(\lambda(t))) \) can be computed by

\[
\dot{x}_i(\lambda(t)) = \begin{cases}
\dot{x}_i(\lambda(t)) & \text{if } \dot{x}_i \leq \dot{x}_i(\lambda(t)) \leq x_i \\
\dot{x}_i(\lambda(t)) & \text{if } \dot{x}_i(\lambda(t)) > x_i \\
\dot{x}_i(\lambda(t)) & \text{if } \dot{x}_i(\lambda(t)) < \dot{x}_i
\end{cases}
\]

(17)
Note that the computations required in Phase 2 are nothing but a set of comparison checks as shown in Eq. (17). It can be easily shown that \( x(t) = (\hat{x}_1(t), \ldots, \hat{x}_n(t)) \) obtained from the two-phase algorithm is the solution of the minimization problem on the RHS of Eq. (5) (Lin & Lin, 1997a, 1997b).

### 2.2.2. Modified parallel block scaled gradient method

The parallel block scaled gradient method with decentralized step-size rule proposed in (Lin & Lin, 2003) is developed to solve block additive unconstrained optimization problems. The minimization Eq. (16) that needs to be solved in Phase 1 can be rewritten as follows:

\[
\min x^T \begin{bmatrix} x_i^T H_i x_i & x_i^T H_{ib} x_i & x_i^T H_b x_i \\ \vdots & \vdots & \vdots \\ x_n^T H_i x_n & x_n^T H_{ib} x_n & x_n^T H_b x_n \end{bmatrix} + \left(r_i + A_i^T \lambda_i(t)\right)^T x_i + r_i^T x_i
\]

\[- A_i^T (t) b_i \]  

(18)

which is indeed a block additive unconstrained optimization problem. We define

\[
F_i(x_i, x_n) = \frac{1}{2} \left( x_i^T H_i x_i + x_i^T H_{ib} x_i + x_i^T H_b x_i \right) + \left(r_i + A_i^T \lambda_i(t)\right)^T x_i + r_i^T x_i
\]

\[- A_i^T (t) b_i \]  

(19)

and then

\[
F(x) = \sum_{i=1}^n F_i(x_i, x_n)
\]

denotes the objective function of Eq. (18).

The modified parallel block scaled gradient method (Lin & Lin, 2003) uses the following iterations

\[ x_i(k + 1) = x_i(k) + \gamma_i(k) s_i(k) \]  

(21)

to solve Eq. (18) for each subsystem \( i, i = 1, \ldots, n \), where \( k \) is the iteration index, \( \gamma_i(k) \) is a positive decentralized step-size, and \( s_i(k) \) is a descent direction obtained from

\[ M_i(x_i(k), x_n) s_i(k) = -\nabla x_i F_i(x_i(k)). \]  

(22)

The matrix \( M_i(x_i(k), x_n) \) in Eq. (22) is a predetermined positive definite matrix. In general, for a large distributed system, for example a power system, formed by strongly interconnected subsystems, a meaningful matrix \( M_i(x_i(k), x_n) \), due to the topological structure, is usually a sparse matrix which allows the linear Eq. (22) to be solved by a sparse matrix technique. For such a case, this efficient technique also contributes to the computational efficiency of our algorithm.

The major difficulty of determining the decentralized step-size is for any subsystem \( i \), when Eqs. (21) and (22) are executed, the stored values of other subsystem’s boundary states \( x_j \) in subsystem \( i \) are not yet updated. This causes some type of asynchronous effect. Therefore, the decentralized step-size rule takes two steps. The first step, after obtaining \( \hat{s}_i(k) \) from Eq. (22), applies the Armijo’s rule (Luenberger, 1984) to each individual subsystem to determine a step-size \( \gamma_i(k) \), then update \( x_i(k+1) \) by Eq. (21) (Lin & Lin, 2003). The second step, after receiving the updated \( x_n \) from neighboring subsystems and based on a criteria which insure the decrease of the updated objective function in Eq. (18), will determine whether we need to further reduce the step-size \( \gamma_i(k) \). As shown in Eq. (Lin & Lin, 2003), if the step-size is small enough, then the criteria of insuring the decrease of the objective function in Eq. (18) is satisfied at iteration \( k \). Thus, \( \gamma_i(k) \) will need not be further reduced and the convergence of the modified parallel block scaled gradient method is still guaranteed. Therefore, we can have a flexible decentralized step-size.

### 2.2.3. Summary of the parallel dual-type method

We can now summarize our parallel dual-type method for solving the dual problem Eq. (4) of QPP Eqs. (1)–(3) as follows.

We initially guess a Lagrange multiplier vector \( \lambda_i \) for each subsystem \( i \). To update \( \lambda_i \) for each subsystem \( i \), we need to compute \( \Delta \lambda_i(t) \) in Eq. (8) first. The increment \( \Delta \lambda_i(t) \) in Eq. (8) for each subsystem \( i \) at iteration \( t \) is obtained by solving Eq. (15) in which \( \Phi_i \) is given in Eq. (11) and \( \nabla \phi(\lambda(t)) \) is computed by Eq. (13). The \( x_i(\lambda(t)) \) in Eq. (13) is obtained by using the two-phase method to solve the minimization problem on the RHS of Eq. (5). The first phase is using the modified parallel block scaled gradient method (Lin & Lin, 2003) to solve the unconstrained minimization problem Eq. (16). This method uses iterations Eq. (21) for each subsystem \( i \), in which the descent direction \( s_i(t) \) is obtained by solving the linear Eq. (22), and the step-size \( \gamma_i(k) \) is determined based on a decentralized step-size rule. The second phase is to project the solution obtained from phase 1 onto the set \( \Omega_k \). The projection formula is shown in Eq. (17). After setting up Eq. (15), \( \Delta \lambda_i(t) \) can be computed. We will then determine the step-size \( \alpha_i(t) \) to update \( \lambda_i(t+1) \) by Eq. (8) and proceed with the next iteration \( t+1 \); the proposed algorithm will terminate when the sequence of \( \lambda_i(t) \), \( i = 1, \ldots, n \), generated by Eq. (8) converges. Note that all the computations for the \( n \) subsystem required above can be executed in parallel as the equations show, except for the step-size \( \alpha_i(t) \) in Eq. (8), which will be discussed in the following subsection.

### 2.3. Algorithm and convergence of the parallel dual-type method

In this section we will describe the detailed algorithmic steps of the parallel dual-type method and discuss its convergence.

**Ascent Direction.** As indicated previously, both \( \Phi_i \) and \( \Phi \) are negative definite matrices. This implies that the \( \Delta \lambda_i(t) = (\Delta \lambda_1(t), \ldots, \Delta \lambda_n(t)) \) in Eq. (8), which is obtained by solving Eq. (15), will be an ascent direction of the dual problem Eq. (4).

**Step-size.** There are two choices for the step-size \( \alpha_i(t) \) in Eq. (8). The first one can be a decentralized step-size like that proposed in the modified parallel block scaled gradient method (Lin & Lin, 2003). The second one can be the centralized Armijo’s step-size (Luenberger, 1984). Since the update of \( x_i(t+1) \) in Eq. (8) is not as often as \( x_i(k+1) \), the communication overhead induced in the centralized Armijo’s rule will not be a problem as can be seen from the algorithmic steps described below. Therefore, we choose the centralized Armijo’s rule to determine the step-size \( \alpha_i(t) \) in Eq. (8). To do so, we can define

\[
\psi_i(\lambda(t)) = F_i(x_i(\lambda(t)), \lambda(\lambda(t))),
\]

(23)

where \( \lambda(\lambda(t)) = (\lambda_1(\lambda(t)), \ldots, \lambda_n(\lambda(t))) \) is the optimal solution of the minimization problem on the RHS of Eq. (5). From the definition of \( F_i \) in Eq. (19) and the \( \phi(\lambda) \) shown in Eq. (7), Eqs. (23) and (18) leads to the following

\[
\phi(\lambda(t)) = \sum_{i=1}^n \psi_i(\lambda(t)).
\]

(24)

The centralized Armijo’s rule (Luenberger, 1984) sets the step-size \( \alpha(t) \) to be \( \beta^m \delta \), where \( m(t) \) is the smallest nonnegative integer \( m \) for which the following condition holds

\[
\sum_{i=1}^n \psi_i(\lambda(t) + \beta^m \Delta \lambda(t)) > \sum_{i=1}^n \psi_i(\lambda(t)) + \delta \beta^m \| \Delta \lambda(t) \|_2^2.
\]

(25)

where \( \delta > 0 \). When \( \lambda(t) = \beta^m \delta \) and \( \lambda(t+1) = \lambda(t) + \alpha \Delta \lambda(t) \) as updated by Eq. (8), then the inequality Eq. (25) implies that
which insures an ascent direction to solve (4).

Algorithm. Now we are ready to state the parallel dual-type (PDT) algorithm in the following:

**On every subsystem $i$:**

1. **Step 0**: Set parameters $\delta > 0$, $\beta < 1$, initial value $\lambda(0)$, and set $t = 0$, $m = 0$. Compute $\Phi_i$ by Eq. (11).
2. **Step 1**: If $t > 0$ and $m = 0$, go to Step 4.
3. **Step 2**: Use modified parallel block scaled gradient method (Lin & Lin, 2003) to solve (16) and obtain $\lambda_t(t)$.
4. **Step 3**: Calculate $k_t(\lambda_t(t))$ from Eq. (17).
5. **Step 4**: Compute $\nabla_{\lambda_t} \phi(\lambda(t))$ by Eq. (13).
6. **Step 5**: Compute $\Delta \lambda_t(t)$ by Eq. (15) and calculate $\|\Delta \lambda_t(t)\|^2$.
7. **Step 6**: Set $m = 0$.
8. **Step 7**: Set $\lambda_{t+1}(t) = \lambda_t(t) + \beta^m \Delta \lambda_t(t)$ (Pseudo update).
9. **Step 8**: Use modified parallel block scaled gradient method (Lin & Lin, 2003) to solve Eq. (16) and obtain $\lambda_{t+1}(t)$.
10. **Step 9**: Calculate $k_{t+1}(\lambda_{t+1}(t))$ by Eq. (17) and calculate $\psi_{t+1}(\lambda_{t+1}(t))$ by Eq. (23).
11. **Step 10**: Send the values of $\psi_{t+1}(\lambda_{t+1}(t))$ and $\|\Delta \lambda_{t+1}(t)\|^2$ to the central computer.

**On the central computer**: The computer will check whether the inequality Eq. (25) holds and send the result to each subsystem.

1. **Step 11**: If Eq. (25) holds, set $\lambda_{t+1}(t + 1) = \lambda_{t+1}(t) + \beta^m \Delta \lambda_{t+1}(t)$, set $t = t + 1$ and go to Step 1; otherwise, set $m = m + 1$ and go to Step 7.

In the above algorithm, only Step 10.i is executed on the central computer and the centralized Armijo’s step-size rule (Luenberger, 1984) is performed by the loop of Step 7 through Step 11. Though the dual function $\psi_i$ and the step-size $\beta$ are heavily dependent on $\beta$, in most of the application problems, the step-size can be obtained in the first iteration of the centralized Armijo’s rule, that is $m = 0$. Under such condition, we indeed do not waste any computation because if $t > 0$ and $m = 0$, we have already obtained $\lambda_{t+1}(t)$ for $\lambda = \lambda(t) + \Delta \lambda(t)$ from Steps 8 and 9, thus, we can skip Steps 2 and 3 as indicated in Step 1.

Convergence. Since $\Delta \lambda(t)$ is an ascent direction, and the centralized Armijo’s rule employed to determine the step-size $\beta(t)$ will ensure the increase of the dual function in Eq. (4) as shown in Eq. (26), therefore, the parallel algorithm (8) will converge to $\tilde{\lambda}$ which maximizes $\Phi(\lambda)$. Consequently, by the Duality Theory (Luenberger, 1984), $\tilde{\Phi}(\tilde{\lambda}) = \{k_i(\tilde{\lambda}_i), \ldots, k_n(\tilde{\lambda}_n)\}$, the solution of the minimization problem on the RHS of Eq. (5) when $\lambda = \tilde{\lambda}$, is the solution of the considered QPP given in Eqs. (1)–(3).

Furthermore, from the algorithmic steps described above we can see that our algorithm can be carried out in a distributed computer network.

3. Application

3.1. The constrained weighted least squares problem

The power system state estimation problem can be formulated as a weighted least squares (WLS) problem based on the measurements of the real and reactive power injections, real and reactive power transmission line flows, bus voltage magnitudes, etc. (Schweppe, 1970; Schweppe & Wildes, 1970). However, to increase the accuracy of the estimated states, we can add some sure equality constraints to this WLS problem. These equality constraints are the real and reactive power-flow balance equations on transmission buses which neither possess any power injection nor load demand. Furthermore, we will focus on the state estimation under normal operating conditions, thus we can constrain the bus voltage magnitudes within secure range. Consequently, the WLS problem becomes a nonlinear constrained optimization problem. In the following, we will describe this problem. However, people who are interested in bad data detection and identification should refer to (Lin & Lin, 1994).

Recently, the deregulation of the power market has spread out worldwide (Dunn et al., 1995). This situation makes the distributed state estimation for a power system formed by interconnected subsystems more attractive, because each subsystem can make its own control decision based on the obtained estimated states. Thus, the parallel dual-type method proposed in this paper especially suits this distributed control and management purpose.

We assume that the considered power system is formed by $n$ interconnected subsystems, and let $N_i$ denote the number of buses in the $i$th subsystem. The state vector $x_i$ of the $i$th subsystem consists of $N_i$ voltage magnitudes and $N_i$ phase angles. We let $x_i$ denote the state vector of the boundary buses of subsystem $I_s$, $I_t$ is involved with boundary buses of subsystem $i$ owing to a measurement, for example, a measurement of real power flow on the transmission line connecting a boundary state in subsystem $i$ and a boundary state in a neighboring subsystem $l$, $l \neq i$; $z_i$ denotes the $m_i$-dimensional vector of measurement values in the $i$th subsystem, the measurements are such as real and reactive power transmission line flows, bus voltage magnitudes, real and reactive power injections, and transformer-tap ratios, etc. (Debs, 1988); $h_i$ denotes the $m_i$-dimensional nonlinear measurement vector function of states corresponding to $z_i$; the random vector $\eta_i$ with dimension $m_i$ denotes the vector of Gaussian measurement error corresponding to $z_i$; the $m \times m_i$ matrix $R_i$ denotes the diagonal covariance matrix of the random vector $\eta_i$.

The measurements in each subsystem $i$ can be expressed as

$$z_i = h_i(x_i, x_s) + \eta_i.$$  

We denote the state vector, measurement value vector, nonlinear measurement vector function, the measurement-error vector and the corresponding covariance matrix of the whole system by $x_{all}, h, \eta$, and $R$, respectively. Then $x = (x_1, x_2, \ldots, x_n), z = (z_1, z_2, \ldots, z_n), h = (h_1, h_2, \ldots, h_n), \eta = (\eta_1, \eta_2, \ldots, \eta_n)$, and $R = (\begin{bmatrix} R_1 & 0 & \ldots & 0 \\ 0 & R_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & R_n \end{bmatrix})$. Furthermore, $N = N_1 + N_2 + \ldots + N_n$ and $m = n_1 + m_2 + \ldots + m_n$ denote the total number of buses and total number of measurements of the system, respectively. Based on the above notions and Eq. (27), the measurements of the whole system can be expressed as

$$z = h(x) + \eta.$$  

Thus, the weighted-squares-error function is

$$\frac{1}{2} \|z - h(x)\|^2 R^{-1}.$$  

Furthermore, the real and reactive power-flow balance on no generation and no-load buses can be expressed as

$$\sum_{j \in J_g} p_j = 0, \quad \sum_{j \in J_q} q_j = 0.$$  

where $l \in L_{no}$ denotes the set of all no generation and no-load buses, and $J_g$ denotes the set of buses connecting with bus $l$; $p_l$ and $q_l$ denote the real and reactive power flow from bus $l$ to bus $j$, respectively. These equality constraints are natural constraints, the inclusion of which will improve the accuracy of the estimated states. Here, we assume that these no generation and no-load buses are in the interior of each subsystem; consequently, its linearization will meet the assumption on the equality constraints Eq. (2) of the QPP considered in Section 1. Since we focus on normal operations of power system,
the bus voltage magnitude should lie within the secure interval $[V_i, V_j]$, and it is commonly accepted that $V_i = 0.95$ p.u. and $V_j = 1.05$ p.u. (Debs, 1988). We note that the states of a bus $i$, $x_i = (V_i, \varphi_i)$, where $\varphi_i$ is the phase angle of bus $i$’s complex voltage. Thus, the constrained WLS problem of the power-system state estimation (Schwepppe, 1970; Schwepppe & Wildes, 1970) can be stated as

$$\min \frac{1}{2} \| \hat{x} - h(x) \|^2 R^{-1} \| \hat{x} - h(x) \|$$

subject to

$$\sum_{i=1}^{n} x_i = 0, \quad \sum_{j=1}^{n} q_i = 0, \quad \forall i \in \mathbb{N}_0,$$

$$V_i \leq V_j \leq V_i, \quad i = 1, \ldots, n,$$

where $x_i$ and $q_i$ denote the real and reactive power flow from bus $i$ to bus $j$ in subsystem $i$, respectively; $\mathbb{N}_0$ denotes the set of no-generation and no-load buses in subsystem $i$.

3.2. Combining successive quadratic programming method with the proposed algorithm

We use a successive quadratic programming method (Luenberger, 1984) with the following iterations to solve the constrained WLS problem Eqs. (33)–(35):

$$x(w + 1) = x(w) + \tau(w) \Delta x(w),$$

where $w$ is the iteration index, $\tau(w)$, a positive step-size, is usually taken to be 1 in typical power-system optimization problem applications; $\Delta x(w)$ is the solution of the following approximate QPP of Eqs. (33)–(35) at iteration $w$:

$$\min \sum_{i=1}^{n} \frac{1}{2} \left[ \sum_{i=1}^{n} \frac{1}{2} \left( T_{xx_i} x_i \right) \right] + \sum_{i=1}^{n} \frac{1}{2} \left( T_{x_i x_i} x_i \right)$$

subject to

$$\sum_{i=1}^{n} \frac{1}{2} \left( R_{xx_i} x_i \right) + \sum_{i=1}^{n} \frac{1}{2} \left( R_{x_i x_i} x_i \right) = 0,$$

$$\forall i \in \mathbb{N}_0, \quad i = 1, \ldots, n,$$

where we have defined

$$G_i(w) = \frac{1}{2} \left[ h_i(x_i) - h_i(x_i) \right] R_i^{-1} \left[ h_i(x_i) - h_i(x_i) \right],$$

with $h_i(x_i)$ denotes $h_i(x_i, x_i)$. Then

$$\nabla_x G_i(w) = -\left[ h_i(x_i, x_i) \right] R_i^{-1} \nabla_x h_i(x_i, x_i),$$

and the matrix $H_i(w)$ in Eq. (37) is defined as

$$H_i(w) = \begin{bmatrix} \nabla_x h_i^T(w) R_i^{-1} \nabla_x h_i(w) + \sigma I_n & \nabla_x h_i^T(w) R_i^{-1} \nabla_x h_i(w) \\ \nabla_x h_i^T(w) R_i^{-1} \nabla_x h_i(w) & 0 \end{bmatrix}.$$
bus and bus 5 is neither a generating nor a loading bus. The same
descriptions are also shown in Fig. 3. Therefore, in Fig. 2 (IEEE 30-
bus system with three subsystems), the total number of generation
and no-load buses is 8, with 4 in Area 1, 1 in Area 2, and 3 in
Area 3, respectively. In Fig. 3 (IEEE 118-bus system with four sub-
systems), the total number of no generation and no load buses is
67, with 17 in Area 1, 18 in Area 2, 14 in Area 3, and 18 in Area
4, respectively. We considered four different cases including differ-
cent numbers of equality constraints and inequality constraints for
the IEEE 30-bus system and the IEEE 118-bus system. We set the
measured values in various test cases as the value obtained from
power–load-flow runs (Debs, 1988). The states, the voltage magni-
tudes and phase angles of all buses in the test system are to be esti-
mated by solving Eqs. (33)–(35).

4.2. The test setup and results

We made two types of tests for our algorithm. The first one is to
test the computational efficiency of the sequential version of the
proposed algorithm (PDT algorithm) in a sequential Sparc worksta-
ion. The software implementations were coded using Fortran in
the sequential version of Sparc workstation. We have run 100 state
estimation examples for each system in each case. In Fig. 2 (IEEE 30-
bus system), the total number of variables is 60, with 22 in Area
1, 20 in Area 2, and 18 in Area 3, respectively. In Fig. 3 (IEEE 118-
business), the total number of variables is 236, with 64 in Area 1,
56 in Area 2, 52 in Area 3, and 64 in Area 4, respectively. The initial
values of the states we used for each test example are all 1.0 p.u.
fors voltage magnitudes and 0 radians for phase angles, and the ini-
tial values of the Lagrange multiplier are, \( \lambda_i(0) = 0, i = 1, 2, \ldots, n \).
In the experiments, we set the following data: \( \beta = 0.90, \delta = 0.95, \sigma = 0.01 \),
and the termination criteria for the iterative procedure are \( \| \Delta x \|_\infty < 0.001 \) in all cases of the test system. The total numbers
of variables ( # of var.), the total numbers of equality constraints ( #
of e.c.), the total numbers of inequality constraints ( # of i.c.), the
average CPU times (Times) expressed in seconds consumed in
100 runs and the final objective value (Obj.) are shown in Tables
1 and 2.

For the purpose of comparison, we also solve the same test
examples using the NCONF and QPROG routines in the same com-
puter with the same initial values and terminating criteria. To
investigate the efficiency of our method, we also solve the same
test examples using the KKT dualization method embedded with
MPBSG (Lin & Lin, 2003) with the same initial values and terminat-
ing criteria. This method is abbreviated as KKTMPBSG. The average
CPU times (Times) consumed for the test examples expressed in
seconds and the final objective value (Obj.) for both the IEEE 30-
bus system and the IEEE 118-bus system are also reported in
Tables 1 and 2, respectively. The KKTMPBSG is efficient when com-
pared with the NCONF and QPROG routines. The main reason is
that MPBSG enable the proposed method to execute in a distrib-
uted system. The dramatic computational efficiency of the pro-
posed PDT algorithm can be observed from the column of the
average CPU times ( Times) consumption in Tables 1 and 2, which
shows that the speed-up ratio of the sequential version of the PDT
algorithm versus NCONF and QPROG routines are around 315 and
280 times in solving the constrained state estimation problems on
the IEEE 30-bus system, and the speed-up ratio of the PDT algo-
rithm versus QPROG is around 850 times in solving the constrained
state estimation problems on the IEEE 118-bus system. This
addresses the computational efficiency of the proposed algorithm.
In addition, the proposed algorithm is more suitable for handling
large-scale systems, because we successfully employ decomposi-
tion technique and sparse matrix technique in our algorithm which
results in the reduction of memory requirements and increases the
computational speed.

In Fig. 4a and b, we show the detailed progression of the objec-
tive function \( F(X) \) with respect to the CPU time (seconds) when
solving the constrained state estimation problem on cases (1d)
and (2d) on IEEE 30-bus and IEEE 118-bus systems, respectively.
Each circle in the figures represents the one iteration of the SQP
method.
Thus, from Fig. 4a and b, we see that our method exhibit a good convergence rate. The CPU time consumed in between circles represent the CPU time consumed by the proposed parallel dual-type method for solving Eqs. (37)–(39).

Table 2
Comparison of the computational efficiency of the sequential version of the PDT algorithm with the NCONF and QPROG routines and KKTMPBSG method on a Sparc workstation in solving constrained state estimation problems on the IEEE 118-bus system

<table>
<thead>
<tr>
<th>Cases</th>
<th># of variables</th>
<th># of equality constraints</th>
<th># of inequality constraints</th>
<th>PDT Algorithm</th>
<th>KKT MPBSG</th>
<th>NCONF</th>
<th>QPROG</th>
<th>Speed-up ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>60</td>
<td>8</td>
<td>30</td>
<td>0.47</td>
<td>5.68</td>
<td>5.29</td>
<td>5.69</td>
<td>146.81</td>
</tr>
<tr>
<td>(ii)</td>
<td>60</td>
<td>10</td>
<td>30</td>
<td>0.48</td>
<td>5.69</td>
<td>5.74</td>
<td>5.70</td>
<td>150.83</td>
</tr>
<tr>
<td>(iii)</td>
<td>60</td>
<td>12</td>
<td>30</td>
<td>0.51</td>
<td>5.69</td>
<td>5.39</td>
<td>5.71</td>
<td>161.79</td>
</tr>
<tr>
<td>(iv)</td>
<td>60</td>
<td>16</td>
<td>30</td>
<td>0.56</td>
<td>5.70</td>
<td>6.94</td>
<td>5.72</td>
<td>182.17</td>
</tr>
</tbody>
</table>

Table 3
Comparison of the computational efficiency of the sequential version of the PDT algorithm with the NCONF and QPROG routines and KKTMPBSG method on a Sparc workstation in solving constrained state estimation problems on the IEEE 118-bus system

<table>
<thead>
<tr>
<th>Cases</th>
<th># of variables</th>
<th># of equality constraints</th>
<th># of inequality constraints</th>
<th>PDT Algorithm</th>
<th>KKT MPBSG</th>
<th>NCONF</th>
<th>QPROG</th>
<th>Speed-up ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>236</td>
<td>70</td>
<td>118</td>
<td>1.69</td>
<td>8.12</td>
<td>27.46</td>
<td>8.13</td>
<td>–</td>
</tr>
<tr>
<td>(iii)</td>
<td>236</td>
<td>110</td>
<td>118</td>
<td>1.81</td>
<td>8.14</td>
<td>31.20</td>
<td>8.15</td>
<td>–</td>
</tr>
<tr>
<td>(iv)</td>
<td>236</td>
<td>134</td>
<td>118</td>
<td>1.85</td>
<td>8.14</td>
<td>30.12</td>
<td>8.15</td>
<td>–</td>
</tr>
</tbody>
</table>

* The solution diverges when NCONF routine is used.

Fig. 5. The experimental PC network.

The processors workload of each PC in the parallel version of the IEEE 118-bus system with four subsystems are described in the following: the total number of variables (# of vars.) is 236, with 64 in Area 1, 56 in Area 2, 52 in Area 3, and 64 in Area 4 in all cases. And we have different numbers of equality and inequality constraints in each case of the test system. For example, in Table 3 (case 3d) the total numbers of equality and inequality constraints (# of e.c., # of i.e.) are 134 and 118, and there are 34 equality constraints and 32 inequality constraints in Area 1, 36 equality constraints and 28 inequality constraints in Area 2, 28 equality constraints and 26 inequality constraints in Area 3, and 36 equality constraints and 32 inequality constraints in Area 4. So as well there are different numbers of equality and inequality constraints in each case of (3a), (3b), and (3c). The computation times of our parallel algorithm (Parallel) is taken to be the longest computation time including communication overhead among the four Pentium IV PCs. The resulting average computation times (Times) expressed in seconds and the final objective value (Obj.) of the 100 state estimation examples in each case on the IEEE 118-bus system are...
shown in Table 3. We also solve the same state estimation examples in each case using the sequential version (Sequential) of our algorithm on a single Pentium IV PC, and the average computation times (Times) expressed in seconds and the final objective value (Obj.) are also shown in Table 3.

From Table 3, we see by the speed-up ratio that the parallel version (Parallel) of our algorithm using four PCs is around 2.2 times faster than the sequential version (Sequential). The reason that the speed-up ratio does not equal 4 is because of the communication load, and the weight of communication expressed in percentage (Comm. Weight %) is also shown in Table 3.

To show the computational efficiency, we compare with the existing state-of-the-art parallel algorithms for quadratic programming (D’Apuzzo & Marino, 2003). The paper deals with a parallel implementation of an interior-point algorithm for solving sparse convex quadratic programs with bound constraints. The method...
presented in (D’Apuzzo & Marino, 2003) used an iterative approach based on conjugate gradient method and on a block diagonal pre-conditioning technique. Moreover, they apply an incomplete Cholesky factorization with limited memory into each block, in order to put together the high degree of parallelism of diagonal pre-conditioning techniques and the greater effectiveness of factorizations procedures. We abbreviate this method as BDPCG algorithm. Based on parallel approach, we used the BDPCG algorithm and the proposed algorithm (PDT algorithm) to solve the same constrained state estimation problems of the IEEE 118-bus system with four subsystems on a 4 PCs network with the same initial condition and the same stop criteria.

The resulting average computation times (Times) expressed in seconds and the final objective value (Obj.) of the 20 state estimation problems in each case on the IEEE 118-bus system are shown in Table 4. From Table 4, we see that the speed-up ratio of the PDT algorithm exceeds about 35 times over than the BDPCG algorithm in solving constrained state estimation problems on the IEEE 118-bus system. At the same time, we find that while the increase of the numbers of inequality constraints and equality constraints, the computational efficiency of our method represents better result in solving the constrained WLS problems. This addresses the computational efficiency of our algorithm.

5. Conclusions

We presented a parallel dual-type (PDT) algorithm for solving a class of quadratic programming problems with block additive objective function. Our algorithm combined with a successive quadratic programming method can be used to solve a class of nonlinear constrained optimization problems such as power-system state estimation problems. The sequential version of our method achieved a dramatic speed-up ratio compared with the NCONF and QPROG routines in solving quite a few power-system state estimation examples in a Sparc workstation. A further speed-up effect was observed when we implemented the proposed algorithm on a PC network. This combined speed-up ratio of our parallel algorithm was credited to the decomposition technique we used to parallelize our algorithm. As demonstrated, our algorithm can be implemented in a distributed computer network. Therefore, the proposed algorithm is suitable for solving constrained optimization problems of large distributed systems.

References


Table 4

Comparison of the computational efficiency of the parallel version of the PDT algorithm with the BDPCG algorithm on a PC network in solving constrained state estimation problems of the IEEE 118-bus system

<table>
<thead>
<tr>
<th>Cases</th>
<th># of variables</th>
<th># of equality constraints</th>
<th># of inequality constraints</th>
<th>PDT algorithm Times (s)</th>
<th>Final objective value</th>
<th>BDPCG algorithm Times (s)</th>
<th>Final objective value</th>
<th>Speed-up ratio(i)/ (i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4a)</td>
<td>236</td>
<td>70</td>
<td>108</td>
<td>96.22</td>
<td>8.13</td>
<td>3387.91</td>
<td>8.13</td>
<td>35.21</td>
</tr>
<tr>
<td>(4b)</td>
<td>236</td>
<td>90</td>
<td>108</td>
<td>97.84</td>
<td>8.13</td>
<td>3602.47</td>
<td>8.13</td>
<td>36.82</td>
</tr>
<tr>
<td>(4c)</td>
<td>236</td>
<td>110</td>
<td>118</td>
<td>99.25</td>
<td>8.14</td>
<td>3674.23</td>
<td>8.14</td>
<td>37.02</td>
</tr>
<tr>
<td>(4d)</td>
<td>236</td>
<td>134</td>
<td>118</td>
<td>107.71</td>
<td>8.15</td>
<td>4076.83</td>
<td>8.15</td>
<td>37.85</td>
</tr>
</tbody>
</table>