Computers and Mathematics with Applications 57 (2009) 1835–1842

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

# Dynamic optimal reactive power dispatch based on parallel particle swarm optimization algorithm☆

Ying Li, Yijia Cao, Zhaoyan Liu, Yi Liu, Quanyuan Jiang*

*College of Electrical Engineering, Zhejiang University, Hangzhou, 310027, China*

ARTICLE INFO

*Keywords:*
Dynamic optimal reactive power dispatch (DORPD)
Parallel computing
Particle swarm optimization (PSO)
Power system

ABSTRACT

In this paper, Message Passing Interface (MPI) based parallel computation and particle swarm optimization (PSO) algorithm are combined to form the parallel particle swarm optimization (PPSO) method for solving the dynamic optimal reactive power dispatch (DORPD) problem in power systems. In the proposed algorithm, the DORPD problem is divided into smaller ones, which can be carried out concurrently by multi-processors. This method is evaluated on a group of IEEE power systems test cases with time-varying loads in which the control of the generator terminal voltages, tap position of transformers and reactive power sources are involved to minimize the transmission power loss and the costs of adjusting the control devices. The simulation results demonstrate the accuracy of the PPSO algorithm and its capability of greatly reducing the runtimes of the DORPD programs.



## 1. Introduction

Optimal power flow (OPF) problem, which was proposed by Carpentier in the early 1960s based on the economic dispatch problem [1], is one of the major issues in operation of power systems. This problem can be divided into two subproblems, optimal reactive power dispatch (ORPD) and real power dispatch. ORPD controls generator terminal voltages, tap position of transformers and outputs of reactive power sources to minimize the transmission loss of the power system and keep the voltage profiles in acceptable range, while satisfying certain operation constraints. Such an objective is considered a classical model of ORPD, which is based on the principle of income maximization without considering costs of control actions. Under real-time circumstances, this sort of solution is not practical because the number of control actions would be too large to be executed in actual power system operation. Hence, dynamic optimal reactive power dispatch (DORPD) was proposed [2]. DORPD deals with different loading conditions during a given future time interval (usually 24 h) instead of one single snapshot of the power network. Compared with traditional ORPD, DORPD is a more complex optimization problem having multiple local minima and nonlinear and discontinuous constraints.

A number of numerical optimization techniques have been proposed to solve the traditional ORPD problem, such as linear programming [3], nonlinear programming [4], interior-point method [5], etc. However, these techniques have limitations in handling nonlinear, discontinuous functions and constraints, and often lead to a local minimum point [8]. Recently, some new heuristic methods like GA [6] and PSO [7,8] have been developed for the ORPD problem. These methods can generate high-quality solutions and have stable convergence characteristic when dealing with ORPD problem and other complex optimization problems in power systems.

☆ This work was supported by Natural Science Fund of Zhejiang Province (Grant Nos. Y105398).

* Corresponding author.
*E-mail addresses:* hnjarod@gmail.com (Y. Li), yijiacao@zju.edu.cn (Y. Cao), ncepulzy@163.com (Z. Liu), yiliu@zju.edu.cn (Y. Liu), jqy@zju.edu.cn (Q. Jiang).

0898-1221/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2008.10.049

Considering the costs of adjusting the control devices (CACD), Zhang et al. [9] presented a novel mathematical model of DORPD, whose objective function is to minimize the sum of power loss in the current time interval and the CACD. They used cataclysmic genetic algorithm to solve this problem. The test results proved that this new model describes the DORPD problem with time-varying loads appropriately because it can decrease active power loss and avoid excessive control actions simultaneously.

However, these techniques cost too much time when dealing with large-scale power systems. To satisfy the real-time control need of large systems, a parallel particle swarm optimization algorithm [10,11] is developed to solve DORPD problem in this paper. PPSO is based on Message Passing Interface which provides a flexible environment for developing high-performance parallel applications. There are several free implementations of MPI available on the Internet, such as MPICH2 [12], LAM/MPI [13] and Open MPI [14]. In this paper, MPICH2 is applied.

The remaining is organized as follows. In the next section, mathematical formulation of dynamic optimal reactive power dispatch will be presented. Section 3 details the PPSO algorithm. Simulation results are given in Section 4. Finally, some conclusive remarks are drawn in Section 5.

## 2. Mathematical formulation of DORPD problem

The purpose of the dynamic optimal reactive power dispatch is to minimize the sum of network active power loss and the costs of adjusting the control devices while satisfying a number of operating constraints. The objective function can therefore be formulated as follows:

$$\min(f_Q + f_O) = \sum_{k\in N_E} P_{kloss} + \widetilde{\mathbf{C}}_{\mathbf{u}}\widetilde{\Delta\mathbf{u}} = \sum_{k\in N_E} g_k(V_i^2 + V_j^2 - 2V_iV_j\cos\theta_{ij}) + C_T\sum_{k\in N_T}\Delta u_{Tk} + C_S\sum_{i\in N_C}\Delta u_{Ci} \tag{2.1}$$

where $f_Q$ is the active power loss in the network, $N_E$ is the set of numbers of network branches, $P_{kloss}$ is the active power loss of the $k$th branch, $g_k$ is the conductance of branch $k$, $i \in N_B$, $N_B$ is the set of numbers of total buses, $j \in N_i$, $N_i$ is the set of numbers of buses adjacent to bus $i$, including bus $i$, $V_i$ is the voltage magnitude at bus $i$, $V_j$ is the voltage magnitude at bus $j$, $\theta_{ij}$ is the voltage angle difference between bus $i$ and bus $j$; $f_O$ is the total readjustment lost, $\widetilde{\mathbf{C}}_{\mathbf{u}}$ is readjustment cost vector of discrete control variables, $\widetilde{\Delta\mathbf{u}}$ is the increment vector of the discrete control variables [9], $C_T$ is readjustment cost of a transformer, $\Delta u_{Tk}$ is the increment of tap position of transformer $k$, $N_T$ is the set of numbers of transformers, $C_S$ is readjustment cost of a compensator, $\Delta u_{Ci}$ is the increment of size of compensator $i$, $N_C$ is the set of numbers of possible reactive power source installation buses. The above function is subject to the following operation constraints:

$$0 = P_{Gi} - P_{Di} - V_i\sum_{j\in N_i} V_j(G_{ij}\cos\theta_{ij} + B_{ij}\sin\theta_{ij}), \quad i \in N_0 \tag{2.2}$$

$$0 = Q_{Gi} + Q_{Ci} - Q_{Di} - V_i\sum_{j\in N_i} V_j(G_{ij}\sin\theta_{ij} - B_{ij}\cos\theta_{ij}), \quad i \in N_{PQ} \tag{2.3}$$

and

$$V_i^{\min} \le V_i \le V_i^{\max}, \quad i \in N_B \tag{2.4}$$

$$T_k^{\min} \le T_k \le T_k^{\max}, \quad k \in N_T \tag{2.5}$$

$$Q_{Gi}^{\min} \le Q_{Gi} \le Q_{Gi}^{\max}, \quad i \in N_G \tag{2.6}$$

$$Q_{Ci}^{\min} \le Q_{Ci} \le Q_{Ci}^{\max}, \quad i \in N_C \tag{2.7}$$

where $P_{Gi}$ and $Q_{Gi}$ are the active and reactive power generated by generators at bus $i$, respectively, $Q_{Ci}$ is the reactive power generated by compensators at bus $i$, $P_{Di}$ and $Q_{Di}$ are the active and reactive power load at bus $i$, $G_{ij}$ and $B_{ij}$ are the transfer conductance and susceptance between bus $i$ and $j$, respectively, $N_0$ is the set of numbers of total buses excluding slack bus, $N_{PQ}$ is the set of numbers of PQ buses; $T_k$ is the tap position at transformer $k$, $N_G$ is the set of numbers of generator buses. Power flow equations (2.2) and (2.3) are used as equality constraints, bus voltage restrictions (2.4), transformer tap-setting restrictions (2.5), reactive generation restrictions (2.6) and reactive power source installation restrictions (2.7) are used as inequality constraints.

In most of the nonlinear optimization problems, the constraints are considered by generalizing the objective function using penalty terms. In the DORPD problem, the control variables, including the generator bus voltages $V_{PV}$ and $V_{slack}$, the tap position of transformer $T$, and the amount of the reactive power source installation $Q_C$ are self-constrained. Voltages of PQ-bus $V_{PQ}$ and injected reactive power of PV-bus $Q_G$ are constrained by adding them as penalty terms to the objective function (2.1). The above problem is generalized as follows:

$$\min F_Q = f_Q + f_O + \sum_{i\in N_V^{\lim}} \lambda_{Vi}(V_i - V_i^{\lim})^2 + \sum_{i\in N_Q^{\lim}} \lambda_{Gi}(Q_{Gi} - Q_{Gi}^{\lim})^2 \tag{2.8}$$

where $N_V^{\lim}$ is the set of numbers of buses on which voltage magnitudes outside limits, $N_Q^{\lim}$ is the set of numbers of buses on which injected reactive power outside limits, $\lambda_{Vi}$ and $\lambda_{Gi}$ are the penalty factors, and both penalty factors are large positive

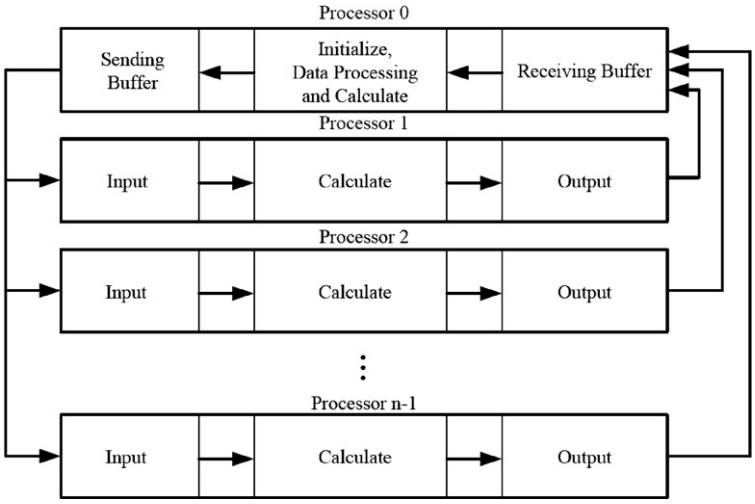


**Fig. 1.** An example of parallel computation based on MPI.

constants, $V_i^{\lim}$ and $Q_{Gi}^{\lim}$ are defined as

$$V_i^{\lim} = \begin{cases} V_i^{\max}; & V_i > V_i^{\max} \\ V_i^{\min}; & V_i < V_i^{\min} \end{cases} \tag{2.9}$$

$$Q_{Gi}^{\lim} = \begin{cases} Q_{Gi}^{\max}; & Q_{Gi} > Q_{Gi}^{\max} \\ Q_{Gi}^{\min}; & Q_{Gi} < Q_{Gi}^{\min}. \end{cases} \tag{2.10}$$

## 3. Parallel particle swarm optimization algorithm

The details of the parallel particle swarm optimization algorithm and how it is implemented to solve the DORPD problem are described in this section.

### 3.1. Parallel computation based on MPI

There are a few approaches to apply parallel computation for DORPD, such as the shared-memory approach and the message passing interface approach, *etc*. The MPI approach is chosen in this paper as it has the following advantages:

- MPI is probably the world's most widely-supported communications library for high-performance computing. Free implementations of MPI are easy to get from Internet.
- With MPI's approximately 125 functions, programmers needn't to implement common communication structures themselves.
- It is easy to learn MPI, as a complete message passing program can be written with only six basic functions: MPI_Init, MPI_Finalize, MPI_Comm_rank, MPI_Comm_size, MPI_Send, and MPI_Recv.
- MPI has portability to almost all the major platforms.

Fig. 1 shows an example of MPI based parallel computation using $n$ processors. First, processor 0 reads and preprocesses all the data before scattering them to other processors' memory. After processor 0's initialization work, all the processors calculate with the data that are sent by processor 0 through message passing interface (Obviously, processor 0 already has the required data in its memory, so there is no communication inside it). When the calculations are done, results are sent back to processor 0's receiving buffer to be dealt with. Then processor 0 decides whether to continue computing or output final results and terminate program.

### 3.2. PSO

PSO is a stochastic optimization technique based on the movement of swarms and inspired by social behavior of bird flocking. Assume there are $M$ particles in a swarm, along with geometrical $N$ parameters to be optimized ($N$ is denoted as the dimension of the solution space). Each of these particles traverses the search space looking for the global minimum or maximum. The position of each particle corresponds to a candidate solution to the optimization problem. As the particles fly through the $N$-dimensional problem hyperspace, the velocity of each particle is determined by the distances from its current position to two important locations, which are denoted by *pBest* and *gBest*. The *pBest* is the location where each particle attains its best fitness value up to the present iteration. And the *gBest* represents the location where the best fitness value was attained by any particle in this swarm.

The velocity and position of $i$th particle on dimension $d$ should be updated according to the following formulas [16]:

$$v_{i,d} = k * (v_{i,d} + \phi_1 * rand() * (pBest_d - x_{i,d}) + \phi_2 * rand() * (gBest_d - x_{i,d})) \tag{3.1}$$

$$x_{i,d} = x_{i,d} + v_{i,d} \tag{3.2}$$

where $v_{i,d}$ and $x_{i,d}$ are the velocity and position of $i$th particle on dimension $d$, respectively, $pBest_d$ is the $d$th dimensional value of $pBest$, $gBest_d$ is the $d$th dimensional value of $gBest$, $\phi_1$ and $\phi_2$ are acceleration constants, $rand()$ is a uniform random value in the range [0, 1], $k$ is the constriction factor which is a function of $\phi_1$ and $\phi_2$ as reflected in the following equation:

$$k = \frac{2}{|\ 2 - \phi - \sqrt{\phi^2 - 4\phi}\ |} \tag{3.3}$$

where $\phi = \phi_1 + \phi_2$ and $\phi > 4$.

In the above procedures, the particle's velocities are limited by some maximum values to enhance the local exploration of the problem hyperspace. Each upper bound, represented as $v_d^{max}$, is often experimentally set at 10%–20% of the dynamic range of the variable on the $d$th dimension.

### 3.3. PPSO implementation for DORPD

When implementing PSO method to solve DORPD problem of a large-scale power system it is often necessary to calculate system power flow for thousands of times to obtain the final solution. The computational time would be too long to meet the real-time control need of the power system if the program is running on a single-processor computer. However, PSO algorithm is by nature parallel since the calculation of each particle's fitness value is independent. Noticing that, MPI based parallel computation is combined with PSO algorithm to form the proposed PPSO method for solving DORPD problem in this study. In this new method, the optimization process is implemented on a multi-processor supercomputer to reduce the computational time drastically.

In its basic form, the proposed PPSO method can only handle continuous variables. However, tap position of transformations and reactive power source installation are discrete variables or integer variables in DORPD problem. In this paper, PPSO has been extended to handle mixed variables. To handle integer variables, simply truncating the real values to integers [8] to calculate fitness value will not affect the search performance significantly. The truncation is only performed when evaluating the fitness function. That means the particles will fly in a continuous search hyperspace regardless of the variable types.

The optimization program flow can be described as follows (Assume there are $n$ processors running this program simultaneously. Processor 0 is the master node. The others are slave nodes.):

(i) Processor 0 - processor $n-1$ initialize PPSO parameters and MPI environment.
(ii) Processor 0 reads power system data and broadcasts public data through message passing interface, such as time-varying load data, bus data, generator data, branch data and transformer data, *etc*. Processor 1 - processor $n-1$ receive and store public data.
(iii) In this DORPD problem, there are $N_t$ time intervals with different load conditions to be optimized. Let the time interval counter $n_t = 1$.
(iv) Processor 0 - processor $n-1$ replace the $(n_t-1)th$ time interval load data of the system with the $n_t th$ time interval load data. Processor 0 randomly generates a swarm of particles, each of which stands for a candidate solution to DORPD problem. Let the size of particle population be $p$.
(v) Let $m = p/n$, $m$ is an integer. If $n$ can't be divided exactly, let $m = m + 1$, thus $m * n > p$.
(vi) Processor 0 partitions the swarm to $n$ parts, each of which has $m$ or $m-1$ particles. Then processor 0 sends these parts to other processors. Each processor receives $m$ or $m-1$ particles from processor 0.
(vii) Each processor evaluates the fitness of its own particles based on the Fast-Decoupled power flow method [15].
(viii) Processor 0 gathers the fitness values of all particles from processor 1-processor $n-1$.
(ix) Processor 0 executes the PSO operator and adjusts all particles' positions in the search hyperspace.
(x) Processor 0 outputs the best solution of the current time interval. If the time interval counter $n_t < N_t$, let $n_t = n_t + 1$, go to step (iv). Otherwise, go to step (xi).
(xi) All processors exit MPI environment and terminate the program.

### 3.4. Speedup factor and efficiency

To evaluate the parallel performance of the PPSO algorithm, the speedup factor $S_N$ and the efficiency $E_N$ are adopted in this paper. These two indices are calculated as follows:

$$S_N = t_1/t_N \tag{3.4}$$

$$E_N = S_N/N \tag{3.5}$$

where $N$ is the number of processors, $t_N$ is the program running time on $N$ processors, $t_1$ is the program running time on 1 processor.

**Table 1**
Basic data of the IEEE test power systems.

| | Number of generators | Number of compensators | Number of transformers |
|---|---|---|---|
| IEEE 14-bus system | 1 | 6 | 6 |
| IEEE 30-bus system | 6 | 3 | 4 |
| IEEE 57-bus system | 7 | 3 | 17 |
| IEEE 118-bus system | 54 | 12 | 9 |
| IEEE 300-bus system | 69 | 8 | 107 |

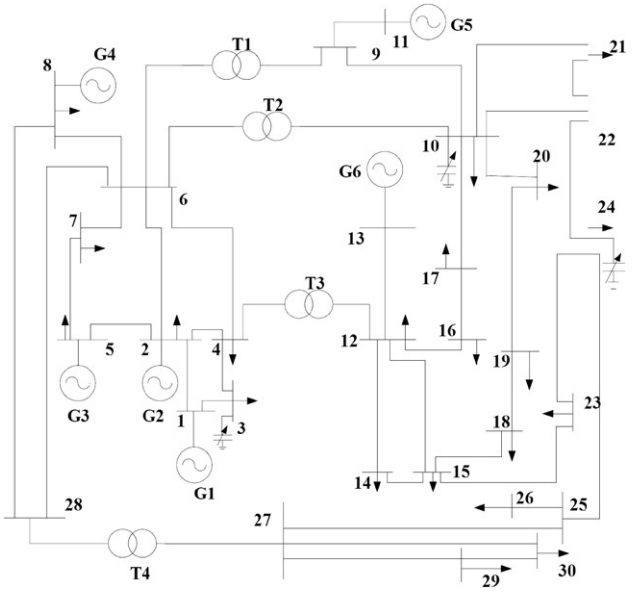


**Fig. 2.** IEEE 30-bus power system.

## 4. Simulation results

A group of IEEE test power systems with time-varying loads are used to verify the effectiveness and efficiency of the proposed PPSO based dynamic reactive power optimization approach. Table 1 shows the basic data of these test systems. The PPSO has been implemented in C programming language with MPICH2 library and simulations have been carried on a SGI Onyx 3900 supercomputer, which has 64 processors. Each of them is 600 MHz and owns a local memory 1 GBytes. Processors communicate and coordinate with each other through message passing interface.

In all studied cases, the following PPSO parameters are used: the population size of the swarm is 80, the total number of iterations is 100, the acceleration constants $\phi_1 = \phi_2 = 2.05$, the constriction factor $k = 0.7298$, the penalty factors $\lambda_{Vi} = 1000$, $\lambda_{Gi} = 1000$.

### 4.1. IEEE 30-bus test case

Fig. 2 shows the IEEE 30-bus system whose data have been given in Ref. [6]. The 24-h load curves of this tested system can be found in Fig. 3.

The network consists of 48 branches, 6 generators and 20 loads. Four branches, (6, 9), (6, 10), (4, 12) and (27, 28), are under load tap setting transformer branches. The possible reactive power source installation buses are 3, 10 and 24. Six buses are selected as PV-buses and slack bus as follows: PV-buses: bus 2, 5, 8, 11, 13, slack bus: bus 1. The others are PQ-buses. The variable limits are given in [6]. The transformer taps and the reactive power source installation are discrete variables with the change step of 0.02 and 0.01 p.u., respectively.

To verify the effectiveness of the proposed method, 3 schemes for solving this problem have been compared here. They are listed in Table 2. Because scheme I and scheme II use traditional ORPD model, both of them have to run the optimization program for 24 times to obtain the final results of the 24-h time interval.

Owing to the randomness in these heuristic algorithms, the whole optimization procedures are executed 30 times when applied to the test system. The results of 3 schemes are compared in Table 3. In this table, $P_{loss}$ is the network transmission power loss in 24 h, $\Sigma N_T$ is the total operating times of transformer taps, $\Sigma N_C$ is the total operating times of capacitors, CACD is the cost of adjusting control devices. The unit adjusting cost of transformer taps is set as 6 kW/times, and that of capacitors is set as 4 kW/times [9]. It is clear that the proposed method has the lowest total cost and least operating times.

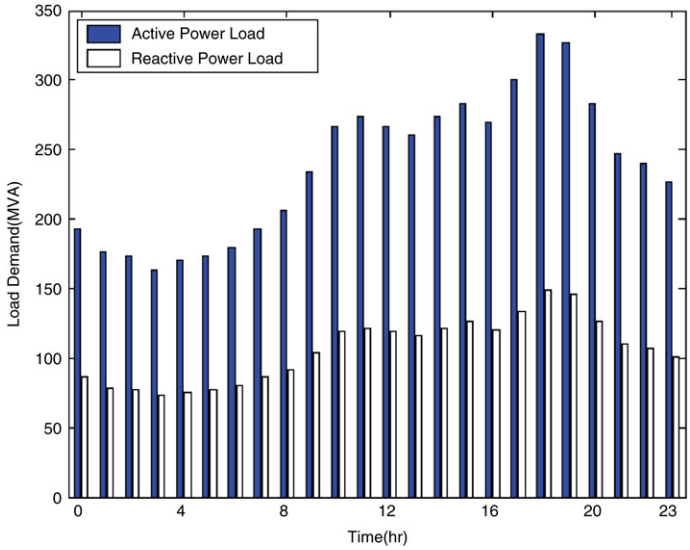


**Fig. 3.** IEEE 30-bus power system's hourly loads.

**Table 2**
3 tested schemes.

| | Optimization algorithm | Mathematical model |
|---|---|---|
| I | SGA | Traditional ORPD |
| II | PSO | Traditional ORPD |
| III | PPSO | DORPD |

**Table 3**
Comparison of ORPD solutions of 3 schemes (MW).

| | $P_{loss}$ | $\Sigma N_T$ | $\Sigma N_C$ | CACD | Tatal cost |
|---|---|---|---|---|---|
| I | 77.18 | 224 | 216 | 2.21 | 79.39 |
| II | 75.94 | 204 | 212 | 2.07 | 78.01 |
| III | 76.53 | 54 | 24 | 0.42 | 76.95 |

**Table 4**
Comparison of average execution time (s).

| | SGA using 1 processor | PSO using 1 processor | PPSO using 40 processors |
|---|---|---|---|
| Average running time | 240.37 | 150.28 | 6.786 |

In addition, the average execution time summarized in Table 4 show that the PPSO method is much faster than SGA and traditional PSO in speed.

### *4.2. Speedup factor and efficiency*

Simulations are performed on the five IEEE test systems introduced in Table 1. On each of these systems, the PPSO program runs using different numbers of processors to demonstrate the parallel performance. Each case has been tested for 30 times and Table 5 shows the average executing time (the running time on 16 and 27 processors are not listed in this table). Obviously, the running time decreases fast as the number of processors grows.

Figs. 4 and 5 show the satisfactory speedup factors and efficiencies of the proposed parallel particle swarm optimization method. The highest speedup factor reaches 35.02 and the corresponding efficiency is 87.6%. It can be seen in Fig. 4 that a larger system's speedup factor is bigger than a smaller system's one when the number of processors are the same. And the relation between the speedup factors and the number of processors is almost linear in the 300-bus power system. It implies that on large-scale power systems the increase of processors will not affect the efficiency much.

So it is not hard to predict that, a more satisfactory speedup factor and a higher efficiency can be carried out when applying the PPSO algorithm in a practical large-scale power system with more than 1000 buses.

**Table 5**
Comparison of average execution time (s).

| | Number of processors | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 | 20 | 27 | 40 |
| 14-bus | 46.386 | 24.966 | 13.980 | 8.520 | 5.784 | 5.280 | 4.788 | 4.434 |
| 30-bus | 151.31 | 77.268 | 40.392 | 21.654 | 12.186 | 10.386 | 8.526 | 6.786 |
| 57-bus | 392.11 | 196.72 | 100.46 | 53.988 | 28.956 | 24.270 | 19.296 | 14.820 |
| 118-bus | 1366.1 | 690.71 | 348.82 | 179.74 | 94.980 | 77.838 | 60.90 | 44.250 |
| 300-bus | 7241.4 | 3695.6 | 1854.8 | 930.43 | 473.27 | 385.12 | 293.41 | 206.77 |

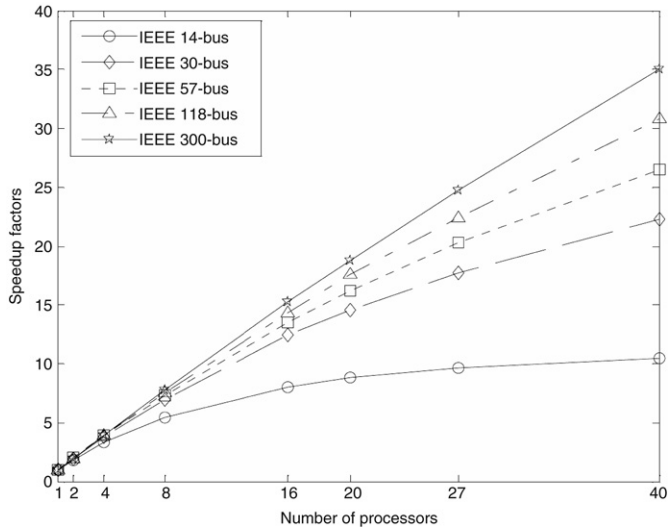


**Fig. 4.** Speedup factors.

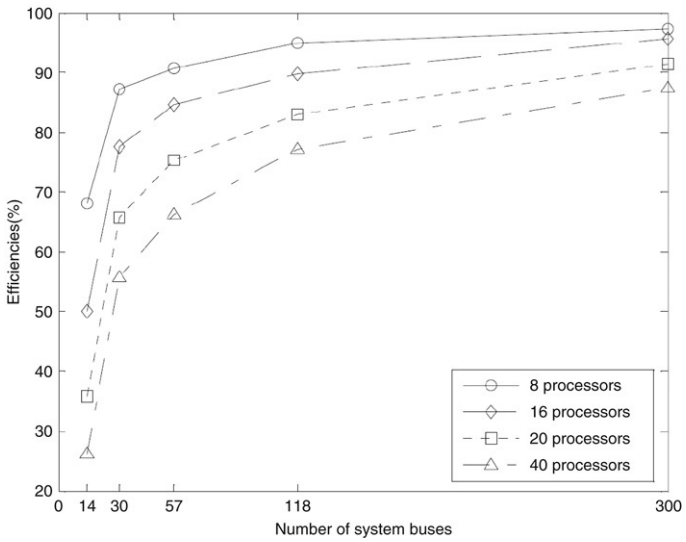


**Fig. 5.** Efficiencies.

## 5. Conclusions

This paper proposes a MPI based parallel particle swarm optimization algorithm for solving dynamic optimal reactive power optimization problem. In the proposed algorithm, the DORPD problem is divided into smaller ones, which can be carried out concurrently by multi-processors. Simulation results obtained on a supercomputer demonstrate the accuracy of the proposed algorithm and its capability of greatly reducing the runtimes of the DORPD programs. It is clear that when

implementing this method in a practical large-scale power system's DORPD problem, an accurate solution can be obtained and a lot of time can be saved.

## Acknowledgements

The authors would like to thank the Center for Engineering and Scientific Computation of Zhejiang University and the organizers of 'The Second International Conference on Bio-Inspired Computing: Theories and Applications'.

## References

[1] J. Carpentier, Contribution to the economic dispatch problem, Bulletin Society Francaise Electriciens 8 (1962) 431–447.
[2] S.S. Sharif, J.H. Taylor, Dynamic optimal reactive power flow, in: Proceeding of American Control Conference, 1998, pp. 3410–3414.
[3] N. Deeb, S.M. Shahidehpour, Linear reactive power optimization in a large power network using the decomposition approach, IEEE Transactions on Power Systems 5 (1990) 697–711.
[4] M.O. Mansour, T.M. Abdel-Rahman, Non-linear VAR optimization using decomposition and coordination, IEEE Transactions on Power Systems 9 (1994) 597–598.
[5] S. Granville, Optimal reactive dispatch through interior point methods, IEEE Transactions on Power Systems 9 (1994) 136–146.
[6] Q.H. Wu, Y.J. Cao, J.Y. Wen, Optimal reactive dispatch using an adaptive genetic algorithm, Electric Power & Energy Systems 20 (1998) 563–569.
[7] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, Y. Nakanishi, A particle swarm optimization for reactive power and voltage control considering voltage security assessment, IEEE Transactions on Power Systems 17 (2000) 723–729.
[8] B. Zhao, C.X. Guo, Y.J. Cao, A multiagent-based particle swarm optimization approach for optimal reactive power dispatch, IEEE Transactions on Power Systems 20 (2005) 1070–1078.
[9] Y.J. Zhang, Z. Ren, Optimal reactive power dispatch considering costs of adjusting control devices, IEEE Transactions on Power Systems 20 (2005) 1349–1356.
[10] S. Cui, D.S. Weile, Application of a parallel particle swarm optimization scheme to the design of electromagnetic absorbers, IEEE Transactions on Antennas and Propagation 53 (2005) 3616–3624.
[11] N. Jin, Y. Rahmat-Samii, Parallel particle swarm optimization and finite-difference time-domain (PSO/FDTD) algorithm for multiband and wide-band patch antenna designs, IEEE Transactions on Antennas and Propagation 53 (2005) 3459–3468.
[12] Available: http://www-unix.mcs.anl.gov/mpi/mpich/.
[13] Available: http://www.lam-mpi.org/.
[14] Available: http://www.open-mpi.org/.
[15] B. Scott, O. Alsac, Fast decoupled load flow, IEEE Transactions on Power Systems 93 (1974) 859–869.
[16] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (2002) 58–73.