Research and application of Distributed Parallel Genetic Algorithm Based on PC Cluster

Keyan Liu[†] Wanxing Sheng[††] and Yunhua Li[†],

Beijing University of Aeronautics and Astronautics, Beijing, China,
China Electric Power Research Institute, Beijing, China

Summary
A distributed parallel genetic algorithm based on PC cluster was proposed, aiming at the disadvantage of traditional genetic algorithm, such as the bad searching quality and long computation time. It adopts the improved genetic simulated annealing algorithm and distributed parallel technique MPI, to implement the distributed computing on PC cluster. The algorithm uses the individual migration strategy to collaboratively optimize every process. The dynamic demes are adopted to balance the CPU load. Computing efficiency is introduced to judge the computing load state. A singular function and an IEEE 14 test system in power system are tested. The results reveal that the algorithm has a good stable searching capacity and good parallel efficiency. Hence the proposed method should have a bright future in the application.

Key words:
Parallel Genetic Algorithm, Load balance, Individual migration, Cluster computing

1. Introduction

Genetic algorithm (GA) is a global optimal method of evolutionary algorithm which is a branch of artificial intelligence. The method has been applied in the multiple application domains[1] becausee it has the features of easy to realize, obvious effect and loose initial requirement and so on. With the improvement of problem complexity and hardness, the average searching quality and solving speed becomes a problem to solve. The difficulty is as follows: 1) the searching time are too long to satisfy the online requirement; 2) the average searching quality is low because the characteristic of randomicity; 3) the prematurity of genetic algorithm in solving multi peak problem. Parallel Genetic Algorithm (PGA) has become to a manner of solving above problems.

Parallel genetic algorithm can be classified as: global parallelization, coarse grain and fine grain[2]. Global parallelization just disposes fitness computation in slave computation unit. The population is divided to sub population for independent evolvement, and individual can be exchanged in sub population in coarse grain style. More sub populations are divided in fine grain, and the size of sub population is smaller. Coarse grain is more popular for its easily realization, and the attractive feature is that it can be simulated or implemented in single processor.

During the past decade there has been a growing concern in power systems about reactive power operation and optimization. Reactive power optimization is a complicated optimization problem that may have several local solutions. The control variables are mixed with continuous and discrete variables. Traditional optimization algorithms, such as linear programming, nonlinear programming, and interior point programming, have the deficiency in solving the reactive power optimization problem while GA can not solve the problem successfully for the time-wasting.

With the improvement of PC performance and the fast development of network technique, PC cluster is becoming a competitive computer system in parallel research. Cluster computing becomes an efficient method of distributed parallel computing. Distributed Parallel Genetic Algorithm (DPGA), uses PC cluster or computer resources on network to compute application. The current research of DPGA focus on multi population, immigration topology and multi agent collaboration etc[3-6] .

To solve online reactive power optimization, this paper proposed a distributed parallel genetic algorithm based on PC cluster, aiming at the disadvantage of traditional genetic algorithm, such as the inferior searching quality and long computation time. Excellent individual immigration between Multi populations in round topology and a strategy of CPU load balance is used in the new algorithm. A DPGA environment is built using distributed library MPI and PC cluster. It adopts the improved genetic simulated annealing algorithm, to implement the distributed computing on PC cluster.

The algorithm uses the individual migration strategy to collaboratively optimize every process. The dynamic populations are adopted to balance the CPU load. Computing efficiency is introduced to judge the computing load state. A common mathematical singular function and an IEEE 14 power test system are tested. The results reveal that the algorithm has a good stable searching capacity and good parallel efficiency. The proposed method should have a bright future in the application and can be used to solve the reactive power optimization of large-scale power system.

Manuscript received February 5, 2007
Manuscript revised February 25, 2007

2. Problem Formulation

The reactive power optimization dispatch (RPOD) has a significant influence on secure and economic operation of power systems.The reactive power optimization is a nonlinear optimization problem in power system, with control variables such as generators voltages, static VAR compensators, OLTC, shunts capacitors, etc. RPOD is usually modeled as a large-scale mixed integer nonlinear programming problem.

The objective of the reactive power optimization is to minimize the active power loss in the transmission network which can be described as follows:

$$f_Q = \min P_L = \sum_{i}^{N_B} V_i \sum_{j \in h} V_j (G_{ij} \cos \delta_{ij} + B_{ij} \sin \delta_{ij}) \quad (1)$$

where $i \in N_B$, $j \in N_i$ .The minimization of the above function is subject to a number of constraints.

$$P_{Gi} - P_{Di} = V_i \sum_{j=1}^{N} V_j (G_{ij} \cos \theta_{ij} + B_{ij} \sin \theta_{ij}) \quad (2)$$

$$Q_{Gi} - Q_{Di} = V_i \sum_{j=1}^{N} V_j (G_{ij} \sin \theta_{ij} - B_{ij} \cos \theta_{ij}) \quad (3)$$

$$V_{i\min} \le V_i \le V_{i\max} \quad (4)$$

$$Q_{Gi,\min} \le Q_{Gi} \le Q_{Gi,\max} \quad (5)$$

$$C_{i\min} \le C_i \le C_{i\max} \quad i \in S_C \quad (6)$$

$$T_{ik\min} \le T_{ik} \le T_{ik\max} \quad (i,k) \in S_T \quad (7)$$

The constraints of equations, which are power balance restrictions of nodes, are defined as (2)-(3). The inequation restrictions of control variable and state variable are defined as (4)-(7). P_i , Q_i , V_i are the active power, reactive power and voltage in node i; Y_{ij} , δ_{ij} are the conductance and phase angle between node i and j.

In the most of the nonlinear optimization problems, the constraints are considered by generalizing the objective function using penalty terms. In the reactive power optimization, the generator bus voltages, V_g , the tap position of transformer, T_{ik} , the amount of reactive power source installations C_i , are control variables which are self constrained. Voltages of PQ -buses, V_i , and injected reactive power of PV -buses, Q_{Gi} , are the state variables. By adding the state variables constraints to the objective function (equation (1)) as penalty terms, the above problem can be written in the following form:

$$F = \min \left\{ P_L + \lambda_1 \sum_{i=1}^{n} \left(\frac{V_i - V_{i\lim}}{V_{i\max} - V_{i\min}}\right)^2 + \lambda_2 \sum_{i=1}^{n} \left(\frac{Q_i^g - Q_{i\lim}^g}{Q_{i\max}^g - Q_{i\min}^g}\right)^2 \right\} \quad (8)$$

$$Q_{i\lim}^g = \begin{cases} Q_{i,\max}^g & (Q_i^g > Q_{i,\max}^g) \\ Q_i^g & (Q_{i,\min}^g \le Q_i^g \le Q_{i,\max}^g) \\ Q_{i,\min}^g & (Q_i^g \le Q_{i,\min}^g) \end{cases} \quad (9)$$

$$V_{i\lim} = \begin{cases} V_{i,\max} & (V_i > V_{i,\max}) \\ V_i & (V_{i,\min} \le V_i \le V_{i,\max}) \\ V_{i,\min} & (V_i < V_{i,\min}) \end{cases} \quad (10)$$

where P_L : function of power loss

λ_1 : penalty coefficient of exceeding voltage

λ_2 : penalty coefficient of exceeding reactive power limit

V_i : voltage of node i

Q_i : reactive power of generator i

G_{ij} : imaginary part of network admittance matrix

B_{ij} : real part of network admittance matrix

The definitions of $Q_{i\lim}^g$ and $V_i^{\lim}$ are defined as (9) and (10). Voltage of main generators, ratio of transformers and shunt-wound compensators are adjusted in order to meet the reactive power demand, improve the voltage quality, and decrease the line loss in the power system.

3. Genetic Simulated Annealing Algorithm

GA is a part of evolutionary algorithms which are a branch of artificial intelligence. The method is based on Darwin's survival of the fittest hypothesis. It uses the technology of community searches to make selection, crossover and mutation on current population, which will create new generation, and gradually causes the population to evolve to contain or approaches the optimal solution[7]. Furthermore, being different to the traditional search techniques, which use characteristics of the problem to determine the sampling point, GA can be used to solve multi-modal, discontinuous and none differentiable functions. But it will spend much computation time to iterations to find the global optimum solution although theoretically, SGA can converge to the solution with probability one, provided that there are enough time, iterations and computations.

The genetic algorithm has the ability to avoid becoming trapped in a "local optimum" solution, and is designed to locate the "global optimum" solution. Genetic algorithms have been shown to solve linear and nonlinear problems by exploring all regions of the state space and exponentially exploiting promising areas through mutation, crossover, and selection operations applied to individuals

in the population[8][9].

Simulated annealing (SA) algorithm is a Monte-Carlo global minimization technique [10]. The algorithm is based upon that of Metropolis et al. [11], which was originally proposed as a means of finding the equilibrium configuration of a collection of atoms at a given temperature. Kirkpatrick et al. [12] who proposed the connection between this algorithm and mathematical minimization, forms the basis of an optimization technique for combinatorial (and other) problems.

This paper uses a hybrid genetic algorithm, which is called adaptive genetic simulated annealing (AGSA) algorithm. The simulated annealing algorithm is adopted as individual substitution strategy to avoid local optimal solution. The basic idea of AGSA is a hybrid optimal algorithm between GA and SA. GA has the merit of searching global solution while the local searching is inferior. SA has the capability of local searching and avoiding be trapped into local optimal solution.

The selection principle of function fitness is to eliminate the individuals of small fitness at the beginning period, and to preserve the excellent characteristics of individuals to the utmost at the later period of optimization. At the same time, it needs to improve precision and avoid falling into local optimal value. The objective of reactive power optimization is to achieve the minimal values of power loss; however the algorithm of GA is to get the maximal fitness of individuals. So, we can extrude the fitness according to the ideology of SA. The function of fitness can be defined as follows:

$$\mathrm{f}(x)=\exp\left(\frac{1}{K*F(x)}\right) \tag{11}$$

$$\mathrm{K}=\begin{cases}\mathrm{T}_0 & k<\dfrac{N_{\max}}{2}\\ \mathrm{T}_0*0.99^{k} & k>\dfrac{N_{\max}}{2}\end{cases} \tag{12}$$

where $F(x)$ is the function value, $\mathrm{f}(x)$ is the fitness of individual, T_0 is the initial temperature of SA, $N_{\max}$ is the maximal iteration number, k is the current iteration.

The reactive power optimization is a mixed integer non-linear programming problem. Because the transformer ratio and capacitance are discrete variable, the voltages of generator are always chosen as discrete value in control center, the control variables are adopted as integer coding.

The chromosome is formed as shown in Fig. 1. There are three chromosome regions (one for each set of control variables), namely, 1) T_{ik} ; 2) C_i ; 3) V_i^g . Encoding is performed using different gene-lengths for each set of control variables, depending on the desired accuracy.

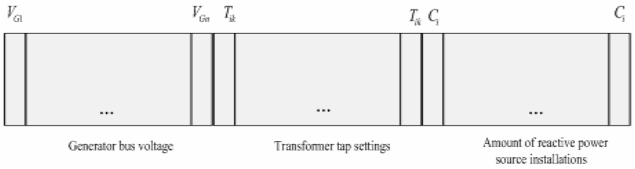


Fig. 1 Control variable of reactive power optimization

$$X=[x_1...x_n]=[N_1^T...N_{n1}^T\,|\,N_1^C...N_{n2}^C\,|\,N_1^{V_g}...N_{n3}^{V_g}] \tag{13}$$

where N_i^T is the integer number of tap position of transformer, N_i^C is the number of reactive power source installations and $N_i^{V_g}$ is the number of generator bus voltage; $n1$ is the number of OLTC transformer, $n2$ is the number of reactive power source installation, $n3$ is the number of generator.

The initial value of N_i^T , N_i^C , $N_i^{V_g}$ can be generated as follows:

$$X_i=\mathrm{int}(rand\times(X_{i\max}-X_{i\min}+1))+X_{i\min} \tag{14}$$

where *rand* is the random number, between 0 and 1, and int( ) is the function of getting integer.

The decoding of a chromosome to the problem physical variable is performed as follows:

$$\begin{aligned}&T_{ik}=T_{ik0}+T_{ik}^{step}\times N_i^T && (i,k)\in S_T\\ &C_i=C_{i0}+C_i^{step}\times N_i^C && i\in S_C\\ &V_i^g=V_0^g+V_i^{step}\times N_i^{V_g}\end{aligned} \tag{15}$$

where T_{ik} 、 C_i 、 V_i^g are the ratio of transformer, reactance value and voltage value of generator respectively. T_{ik0} 、 C_{i0} 、 V_0^g correspond to the initial value and T_{ik}^{step} , C_i^{step} , V_i^{step} correspond to the step value respectively.

In SGA, P_c and P_m are const values, which is low efficiency for sophisticated multi-variable problem. This paper adopts adaptive crossover and mutation operation. P_c and P_m can be changed by the fitness of function of individuals. When the population is to falling into local optimal point, P_c and P_m need to be increased. If the population is to disperse, P_c and P_m need to be reduced. The values of P_c and P_m should be varied depending on the average value of f .

$$P_c=\begin{cases}\dfrac{k_1(f_{\max}-f)}{f_{\max}-f_{avg}} & f\ge f_{avg}\\ k_3 & f<f_{avg}\end{cases} \tag{16}$$

$$P_m = \begin{cases} \dfrac{k_2(f_{\max} - f')}{f_{\max} - f_{avg}} & f' \geq f_{avg} \\ k_4 & f' < f_{avg} \end{cases} \quad (17)$$

where k_1, k_2, k_3 and k_4 need to be less than 1.0 to constraint P_c and P_m to the range 0.0-1.0. $f_{\max}$ is the maximum fitness value, and f_{avg} is the average fitness value of the population. f is the larger of fitness values of the individuals selected for crossover and f' is the fitness of the i -th chromosome to which the mutation with probability P_m is applied.

Through selection, crossover and mutation, a new set of population created, and then two chromosome of every individual are selected randomly to be turbulent point. If the fitness of new individual increases, then this new individual can be accepted; if the fitness decreases, this new individual can be accepted by probability p .

$$p(T_{k+1}) = \begin{cases} 1 & f_{k+1} > f_k \\ \exp(-\dfrac{f_{k+1} - f_k}{T_{k+1}}) & f_{k+1} \leq f_k \end{cases} \quad (18)$$

$$T_{k+1} = \alpha * T_k$$

where f_{k+1} and f_k are the fitness of new individual and old one respectively, $p(T_{k+1})$ is the acceptable probability at the temperature of T_{k+1}. α is the decline coefficient of temperature.

The condition for this algorithm to stop is as follows: if the iteration number of algorithm does not exceed the maximal iteration number and the optimal individual is feasible.

The genetic algorithm and simulated annealing algorithm are two heuristic algorithms that can be paralleled. So the proposed algorithm above can be applied in distributed parallel environment, especially for a large scale optimization.

4. Proposed DPGA Based on Load Balance of Multi Population

4.1 Load Balance of Multi Population

Many researchers use a pool of processors to speed up the execution of a sequential algorithm, just because independent runs can be made more rapidly by using several processors than by using a single one. In this case, the usages of every processor are different. The load of every node and communication are different. In consideration of parallel computation efficiency, we introduce the dynamic load balance. The main idea is that computation waiting time between different processors can be eliminated through recombining genetic population.

The population number of every slave process is equal at initial stage. After one immigration interval, slave process begins to migrate individuals. At the same time, the slave process submits a computing time to master process on two communication intervals. Master process evaluates the computation capability and judge the population number of every slave process at next interval. The computation capability of every slave process usually use computation time(CPU time) [13] and computation efficiency[14]. Computation efficiency is used in this paper to evaluation.

For sub population A, we assume the interval of two communication is T_A, and the total number of individual between two communication is N_A. The computation efficiency and new group size of sub population P_A can be expressed as:

$$\eta_A = \frac{N_A}{T_A} \quad (19)$$

$$P_A = N * \frac{\eta_A}{\eta_A + \eta_B + ... + \eta_n} \quad (20)$$

In equation (19) and (20), N is the size of total population, and $\eta_A, \eta_B ..., \eta_n$ are computation efficiency of sub population respectively. If the size of new sub population is larger than its primary sub population, the difference will be randomly added to primary sub population. If the new sub population is less than primary sub population, the inferior difference individuals will be discarded.

4.2 The Flow of Proposed Distributed parallel Genetic Algorithm

The coarse-grain master-slave framework of proposed distributed parallel genetic algorithm is shown in Fig.2. We describe the master and slave algorithm respectively. The procedure of master process is as follows:

Step 1: The master process fork np-1 slave processes, and every slave process is used to implement sequential GA;

Step 2: The master process waits for the message of slave process in block communication style;

Step 3: The master process computes efficiency of every slave processor, $\eta_A, \eta_B ..., \eta_n$, after slave process submitted the info of load balance. The master process sends the message, adding or discarding number of sub population, to every slave process;

Step 4: The master process waits for message of slave process in block communication style. If the received message is the terminative signal, the procedure goto step 5; otherwise the procedure circulates step 2;

Step 5: The master process send signal to slave process, to terminate every slave process;

Step 6: Comparing the received optimal value of every slave process, the master process gets the global optimal value, the algorithm ends.

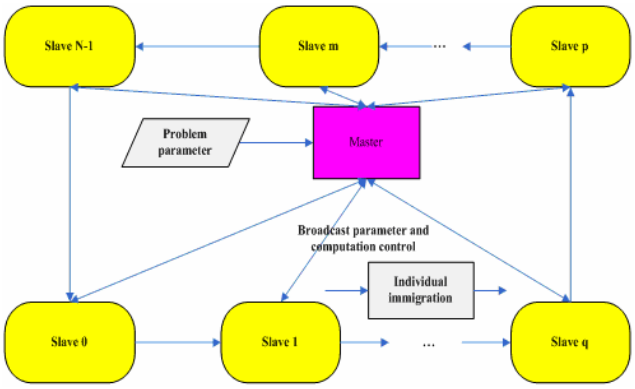


Fig. 2. Framework of distributed parallel genetic algorithm

The procedure of slave process algorithm is as follows:

Step 7: Optimal searching of genetic simulated annealing algorithm on every slave process independently;

Step 8: Submit the individual number of total generation and time interval between two communications to master process, and wait for the message of master adjusting sub population;

Step 9: Slave process received the adjusting info, and revises the number of sub population according to Equation (20);

Step 10: Slave process judges whether it is satisfied to immigrate. If it is satisfied, individuals in sub population are immigrated to other population according to immigration topology, and immigrated from other population are received in sub population and substituted. The substitution strategy is to substitute the inferior with the immigrated individual;

Step 11: Slave process judges whether it is satisfied the convergence condition. If it is satisfied the condition, slave process sends the optimal info and convergence flag; otherwise slave process goto step 7;

Step 12: Waiting the terminative signal from master process, and terminate.

The MPI (Message Passing Interface) is a library of message-passing routines used in parallel environment[15]. The MPI functions support process-to-process communication, group communication, setting up and managing communication groups, and interacting with the environment. This paper uses MPICH to build the high performance distributed parallel environment linking the existing PC cluster. The above algorithm can be run in multi processors of distributed parallel style or in single processor of sequential style.

5. Common Mathematical Simulation Test

The convergence speed is the main norm of algorithm optimization performance. For singular function optimization problem, there are demands of quick convergence and getting the global optimum value. We take a singular function to test the proposed algorithm above. The function is shown in (21). From the projection of Fig 3, it can be found that the global optimal solution is (1, 1). The function is hard to get the solution by traditional mathematical methods.

$$\begin{cases} f_2(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 \\ -2.048 \le x_i \le 2.048 \end{cases} \quad (21)$$

The algorithm is implemented on six HP high performance workstations through 100Mbps LAN. Every machine is configured with MPICH 1.2.5[16] to form a distributed parallel platform. In following test, the immigration interval is 5, and every 2 processes are allocated in one CPU. Using the proposed dynamic DPGA, we can get the optimal solution (0.9998, 0.9998), which is very close to the global solution (1, 1).

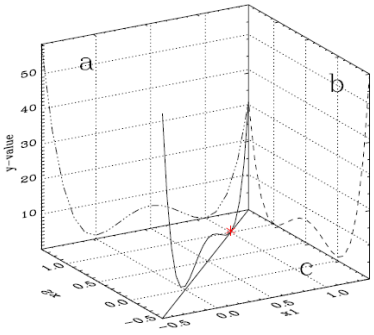


Fig. 3. Plots of singular function

6. Application

In this section, the proposed DPGA is evaluated to solve reactive power optimization in power system application using the IEEE 14-bus power test system. It has a total of 11 control variables as follows: six generator-bus voltage magnitudes, forty-one branch, four adjustable transformer, and two bus-shunt admittances.

The control variables are encoded as decimal integer coding in every single process. The adaptive crossover

ratio is 0.6~0.9 and the adaptive mutation ratio is 0.01~0.06. The scale of immigration is taken as 1, 2 and 3 respectively. In the algorithm, the master process doesn't

participate in computing. It only takes charge of collect data info and control the computation efficiency. Slave process does sub population evolution independently, and interacts with other slave processes and master process.

Fig 4. Comparison of search performances of DPGA with different migration scale

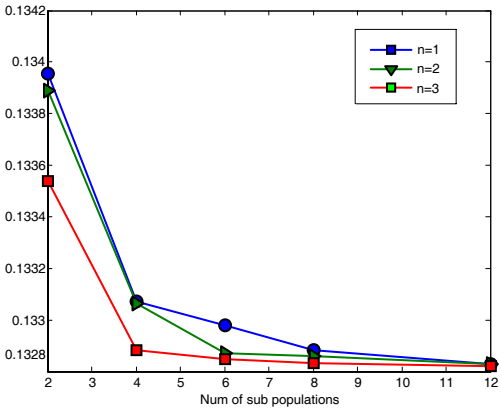


Table 1. Comparison of optimal value with different process number (n=1)

Population	Best value	Worst value	Average value
60	0.13282	0.13326	0.13295
80	0.13280	0.13307	0.13294
120	0.13280	0.13298	0.13288
160	0.13280	0.13282	0.13282

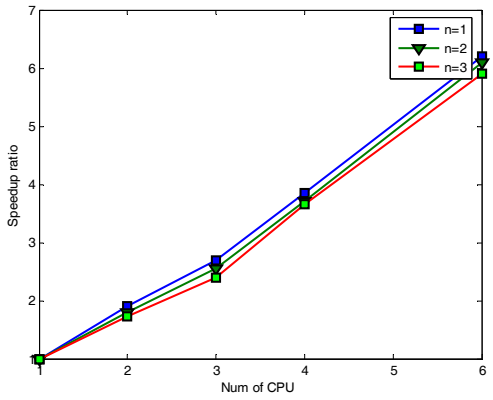
(a. without dynamic load balance) (b. dynamic load balance)

With different immigration scale, we can get the performance comparison on n=1, 2, 3, which is shown in Fig.4. The comparison of speedup ratio is shown in Fig.5. From Fig.5, it can be seen that the speedup ratio can be linear with the scale n=1. Because of the intrinsically heuristic random nature, every test case is tested for 8 times to get the best value, the worst value and average value. The comparison of multi population dynamic load balance with different process is shown in Tab 1. With the increase of process, the searching quality becomes better. From the value of Tab 1, it denotes that adopting computation style of multi population dynamic load balance can get better searching quality and speedup ratio. Comparison of optimal value of different population size is shown in Tab.2.

Table 2. Comparison of optimal value of different popsize (np-1=12)

CPU	Best value	Worst value	Average value	Speedup
1	0.13282	0.13326	0.13295	1
2	0.13280	0.13307	0.13294	1.85
4	0.13280	0.13298	0.13288	3.83
6a	0.13280	0.13282	0.13282	5.91
6b	0.13280	0.13282	0.13282	6.23

Fig 5. Comparison of Speedup Ratio with different migration styles



6. Conclusion

In this paper, a DPGA was proposed based on PC cluster and MPI. The algorithm uses the individual migration strategy to collaboratively optimize every process. The dynamic sub populations are adopted to balance the CPU load. The singular function and reactive power optimization problem in power system are tested. The results show that the proposed algorithm should have a bright future in the application. Using dynamic population method can keep load balance without improving the communication; the algorithm has the superiority of global searching and has a stable searching quality; the algorithm not only can promote the theory of distributed genetic algorithm, but also has a direct role on dynamic reactive power optimization.

Our next task is to use our proposed algorithm to solve large scale computation of power grid based on existing PC cluster environment.

References

[1] L. Davis, The handbook of genetic algorithms[M]. New York: Van Nostrand Reingold, 1991.

[2] E. Alba,M. Tomassini, Parallelism and Evolutionary Algorithms[J], IEEE trans on Evolutionary computation, 2002,6(5): 443-462.

[3] G. A. Sena, D. Megherbi, G. Isern, Implementation of a parallel genetic algorithm on a cluster of workstations:

Traveling Salesman Problem, a case study[J], Future Generation Computer Systems, 2001,17(4): 477-488.
[4] Cao Yijia, Application parallel genetic algorithms to economic dispatch—Effects of migration strategy on algorithms' performance[J], Automation of Electric Power Systems, 2002,26(13): 20-24.
[5] Xiong shengwu, Wang guan, A distributed genetic algorithm based on dynamic demes[J], Journal of WUT(information & Management Engineering), 2002,24(5): 9-12.
[6] N. U. Ahmed, X. Lu, L. O. Barbosa, An efficent parallel optimization algorithm for the Token Bucket control mechanism[J], Computer Communication, 2006,29(12): 2281-2293.
[7] D. E. Goldberg, Genetic algorithms in search, optimization and machine learning[M]. Boston: Addison Wesley Publishing Company, 1989.
[8] Z. Michalewicz, Genetic Algorithm + Data Structures = Evolution Programs[M]. New York: Springer-Verlag, 1994.
[9] J. Joines and C. Houck, On the use of non-stationary penalty functions to solve constrainted optimization problems with genetic algorithms[C], IEEE International symposium Evolutionary computation, Orlando, 1994.
[10] A. Corana, M. Marchesi, C. Martini, and S. Ridella, Minimizing multimodal functions of continuous variables with the simulated annealing algorithm[J], ACM Transactions on Mathematical Software, vol. 13, pp. 262-280, 1987.
[11] N. Metropolis, Rosenbluth, A.W., Rosenbluth, M. N., Teller, A.H. and Teller, E., Equations of State Calculations by Fast Computing Machines[J], J. Chem. Phys, vol. 21, pp. 1087-1092, 1958.
[12] S. Kirkpatrick, Gerlatt, C. D. Jr., and Vecchi, M.P., Optimization by Simulated Annealing[J], Science, vol. 220, pp. 671-680, 1983.
[13] Xiong shengwu,Li chengjun, Distributed evolutionary algorithms to TSP based on cluster of PC[J], Mini-Micro systems, 2003,24(6): 959-961.
[14] Chen qian, Li xing, Parallel genetic algorithm on Load-unbalanced parallel computer[J], Computer Engineering and applications, 2000,36(9): 55-57.
[15] Du zhihui, parallel programming of High performance computation-MPI parallel design[M], Beijing, Tsinghua university press, 2002.
[16] http://www-unix.mcs.anl.gov/mpi/mpich/.

Keyan Liu is a PhD candidate in school of automation science and electrical engineering of Beijing University of Aeronautics &Astronautics, China. He was born in 1978 in HeNan province, China. His research interests are computer modeling and simulation, distributed computation and intelligent application in mechanical & electrical system and power electric system.

Wanxing Sheng is a professor in China Electric Power Research Institute(CEPRI). He was born in 1965 in HeNan province, China. He got his Ph.D. in 1995 from Xi'an Jiaotong University, China. His research interests include distributed computation and intelligent application in mechanical & electrical system and power electric system.

Yunhua Li is a professor in school of automation science and electrical engineering of Beijing University of Aeronautics &Astronautics, China. He was born in 1963 in HeBei Province, China. He got his Ph.D. in 1994 from Xi'an Jiaotong University, China. His research interests include mechatronics, control of mechatronic and hydraulic control of the aircraft, control theory and its application, vehicle electronic and hydraulic control, etc.