A NOVEL PARTICLE SWARM OPTIMIZATION APPROACH FOR OPTIMAL REACTIVE POWER DISPATCH

Bo Zhao, Quanyuan Jiang, Chuangxin Guo, Yijia Cao
Department of Electrical Engineering of Zhejiang University
Zhejiang, China
zhaobozju@zju.edu.cn

Abstract –In this paper, a solution to reactive power dispatch problem with a novel particle swarm optimization approach based on multi-agent systems (MAPSO) is presented. The method integrates multi-agent system (MAS) and particle swarm optimization algorithm (PSO). An agent in MAPSO represents a particle to PSO and a candidate solution to the optimization problem. All agents live in a lattice-like environment, with each agent fixed on a lattice-point. In order to quickly obtain optimal solution, each agent competes and cooperates with its neighbors, and it can also learn by using its knowledge. Making use of these agent-agent interactions and evolution mechanism of PSO, MAPSO realizes the purpose of optimizing the value of objective function. MAPSO applied for optimal reactive power dispatch is evaluated on an IEEE 30-bus power system and a practical 118-bus power system. Simulation results show that the proposed approach converges to better solutions much faster than the earlier reported approaches. The optimization strategy is general and can be used to solve other power system optimization problems as well.

Keywords: Reactive power dispatch, Particle swarm optimization, Multi-agent system

1 INTRODUCTION

Reactive power optimization is a sub-problem of the optimal power-flow (OPF) calculation, which determines all kinds of controllable variables, such as reactive-power outputs of generators and tap ratios of transformers, outputs of shunt capacitors/reactors, etc., and minimizes transmission losses or other appropriate objective functions, while satisfying a given set of physical and operating constraints. Up to now, a number of techniques ranging from classical techniques like gradient-based optimization algorithms to various mathematical programming techniques have been applied to solve this problem [1]. Recently, due to the basic efficiency of interior-point methods, which offer fast convergence and convenience in handling inequality constraints in comparison with other methods, interior-point linear programming, quadratic programming, and nonlinear programming [2,3] methods have been widely used to solve the OPF problem of large-scale power systems. However, these techniques have severe limitations in handling non-linear, discontinuous functions and constraints, and function having multiple local minima. Unfortunately, the original reactive power problem does have these properties.

In the last decade, many new stochastic search methods have been developed for the global optimization problems, such as simulated annealing, genetic algorithms and particle swarm optimization. Particle swarm optimization (PSO) is one of the evolutionary computation techniques [4]. It was developed through simulation of a simplified social system, and has been found to be robust in solving continuous nonlinear optimization problems. Although the PSO seems to be sensitive to the tuning of some weights or parameters, many researches are still in progress for proving its potential in solving complex power system problems [5]. Generally, PSO has a more global searching ability at the beginning of the run and a local search near the end of the run. Therefore, while solving problems with more local optima, there are more possibilities for the PSO to explore local optima at the end of run. However, the reactive power optimization problem does have these properties in itself. For these reasons, a reliable global approach to power system optimization problems would be of considerable value to power engineering community.

Recently, agent-based computation has been studied in the field of distributed artificial intelligence and computer science [6]. Enlightened by multi-agent system and PSO, this paper integrates multi-agent system and PSO to form a novel multi-agent based particle swarm optimization approach (MAPSO), for solving the reactive power optimization problem. In MAPSO, an agent represents a particle to PSO and a candidate solution to the optimization problem. All agents live in a lattice-like environment, with each agent fixed on a lattice-point. In order to quickly obtain optimal solution, they compete and cooperate with their neighbors, and they can also use knowledge. Making use of these agent-agent interactions and evolution mechanism of PSO, the proposed method can find high quality solutions reliably with the faster convergence characteristics in a reasonably good computation time. MAPSO applied for optimal reactive power is evaluated on an IEEE 30-bus power system and a practical 118-bus power system. Simulation results show that the proposed approach converges to better solutions much faster than the earlier reported approaches.

2 PROBLEM FORMULATION

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

$$f_Q = \sum_{k \in N_E} g_k \left(V_i^2 + V_j^2 - 2V_iV_j \cos\theta_{ij} \right) \quad (1)$$

where $k = (i, j)$; $i \in N_B$; $j \in N_i$. The symbols of the above equation and in the following context are given in the Appendix. The minimization of the above function is subject to a number of constraints:

$$0 = P_{Gi} - P_{Di} - V_i \sum_{j \in N_i} V_j \left(G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij} \right) \quad i \in N_0 \quad (2)$$

$$0 = Q_{Gi} - Q_{Di} - V_i \sum_{j \in N_i} V_j \left(G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij} \right) \quad i \in N_{PQ} \quad (3)$$

and

$$V_i^{\min} \le V_i \le V_i^{\max} \qquad i \in N_B \quad (4)$$

$$T_k^{\min} \le T_k \le T_k^{\max} \qquad k \in N_T \quad (5)$$

$$Q_{Gi}^{\min} \le Q_{Gi} \le Q_{Gi}^{\max} \qquad i \in N_G \quad (6)$$

$$Q_{Ci}^{\min} \le Q_{Ci} \le Q_{Ci}^{\max} \qquad i \in N_C \quad (7)$$

where power flow equations are used as equality constraints, reactive power source installation restrictions, reactive generation restrictions, transformer tap-setting restrictions and bus voltage restrictions are used as inequality constraints.

In the most of the nonlinear optimization problems, the constraints are considered by generalizing the objective function using penalty terms. In the reactive power dispatch problem, the generator bus voltages, V_{PV} and V_s, the tap position of transformer, T, the amount of the reactive power source installation Q_C, are control variables which 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 (equation (1)). The above problem is generalized as follows:

$$F_Q = f_Q + \sum_{i \in N_V^{\lim}} \lambda_{Vi} \left(V_i - V_i^{\lim} \right)^2 + \sum_{i \in N_Q^{\lim}} \lambda_{Gi} \left(Q_{Gi} - Q_{Gi}^{\lim} \right)^2 \quad (8)$$

where λ_{Vi} and λ_{Gi} are the penalty factors, and both penalty factors are large positive constants; $V_i^{\lim}$ and $Q_{Gi}^{\lim}$ are defined as

$$V_i^{\lim} = \begin{cases} V_i^{\max}; V_i > V_i^{\max} \\ V_i^{\min}; V_i < V_i^{\min} \end{cases}, \qquad Q_{Gi}^{\lim} = \begin{cases} Q_{Gi}^{\max}; Q_{Gi} > Q_{Gi}^{\max} \\ Q_{Gi}^{\min}; Q_{Gi} < Q_{Gi}^{\min} \end{cases}$$

3 A NOVEL PARTICLE SWARM OPTIMIZATION APPROACH

3.1 Particle Swarm Optimization

Particle swarm optimization (PSO) is a novel optimization method developed by Eberhart, *et al* [4,7]. It uses a number of particles that constitute a swarm. Each particle traverses the search space looking for the global minimum (or maximum). In a PSO system, particles fly around in a multidimensional search space. During flight, each particle adjusts its position according to its own experience, and the experience of neighboring particles, making use of the best position encountered by itself and its neighbors.

Let x and v denote a particle coordinates and its corresponding flight speed in a search space, respectively. The best previous position of a particle is recorded and represented as *pBest*. The index of the best particle among all the particles is represented as *gBest*. To ensure the convergence of PSO, Eberhart indicate that use of a constriction function may be necessary [6]. At last, the velocity and position of each particle can be calculated as shown in the following formulas:

$$v_{d+1} = k * (w * v_d + \varphi_1 \cdot rand() * (pBest - x_d) + \varphi_2 * rand() * (gBest - x_d)) \quad (9)$$

$$x_{d+1} = x_d + v_{d+1} \quad (10)$$

where d is pointer of generations, x_d is current position of particle at the dth generation, v_d is velocity of particle at the dth generation, w is inertia weight factor, φ_1 and φ_2 are acceleration constant, *rand()* is uniform random value in the range [0,1], k is constriction factor which is a function of φ_1 and φ_2 as reflected in (11).

$$k = \frac{2}{\left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|} \quad (11)$$

where $\varphi = \varphi_1 + \varphi_2$ and $\varphi > 4$.

3.2 Multi-agent System

Agent-based computation has been studied for some years in the field of artificial intelligence. Multi-agent systems (MAS) are computational system in which several agents interact or work together in order to achieve goals. According to [8], an agent is a physical or virtual entity that essentially has the following properties:

(1) Agents live and act in a given environment.
(2) Agents are able to sense its local environment, and to interact with other agents in its local environment.
(3) Agents attempt to achieve particular goals or perform particular tasks.
(4) Agents are able to respond in a timely fashion to changes that occur in it based on their learning ability.

MAS technology provides an opportunity to compute and optimize many complicated problems. Hence, this paper combines PSO and MAS to form a novel optimal algorithm. In general, the following four elements should be defined when MAS are used to solve problems.

(1) The meaning and the purpose of each agent.
(2) An environment where all agents live.
(3) The definition of a local environment.
(4) A set of behavioral rules, governing the interaction between the agents and their environment.

In the following, the definitions of the above four elements and the implementation of MAPSO are described in detail.

3.3 Multi-Agent Based Particle Swarm Optimization

In MAPSO, an agent represents not only a candidate solution to the optimization problem but also a particle to PSO. Firstly, a lattice-like environment is constructed, with each agent fixed on a lattice-point. In order to quickly obtain optimal solution, each agent competes and cooperates with their neighbors, and they can also use knowledge to obtain high-quality optimal

solution by self-learning. Making use of evolution mechanism of PSO, it can speed up the transfer of information among agents, and MAPSO can realize the purpose of optimizing the value of objective function.

3.3.1 The purpose of each agent

In MAPSO, an agent *a* represents a candidate solution to the optimization problem in hand. Hence, agent *a* has a fitness value to the optimization problem. For solving reactive power problem, its fitness value is the value of the active power loss, i.e., equation (1),

$$f(\alpha) = \sum_{k \in N_E} g_k \left(V_i^2 + V_j^2 - 2V_i V_j \cos\theta_{ij} \right) \qquad (12)$$

The purpose of *a* is to minimize the real power transmission losses and keep all the voltages within the limits as much as possible.

3.3.2 The definition of an environment

An environment is organized as a lattice-like structure in Figure 1. In the environment *L*, each agent is fixed on a lattice-point and each circle represents an agent, the data in circle represents its position in the environment *L*. Due to representing a particle in PSO, each agent in its database contains two data, i.e., particle's current velocity and its coordinates in the search space. The size of L is $L_{size} \times L_{size}$, where L_{size} is an integer.

1,1	1,2	.....	1, Lsize
2,1	2,2	.....	2, Lsize
.....	.....	.....	.....
Lsize, 1	Lsize, 2	.....	Lsize, Lsize

Figure 1: Structure of the environment

3.3.3 The definition of the local environment

Since each agent can only sense its local environment in MAS, the definition of the local environment is very important. In this paper, suppose that the agent α located at *(i,j)* is represented as $a_{i,j}$, $i,j=1,2,\ldots,L_{size}$, then the neighbors of $a_{i,j}$, $N_{i,j}$ are defined as follows:

$$N_{i,j} = \left\{ \alpha_{i^1,j}, \alpha_{i,j^1}, \alpha_{i^2,j}, \alpha_{i,j^2} \right\} \qquad (13)$$

where $i^1 = \begin{cases} i-1 & i \neq 1 \\ L_{size} & i = 1 \end{cases}$, $j^1 = \begin{cases} j-1 & j \neq 1 \\ L_{size} & j = 1 \end{cases}$,

$i^2 = \begin{cases} i+1 & i \neq L_{size} \\ 1 & i = L_{size} \end{cases}$ $j^2 = \begin{cases} j+1 & j \neq L_{size} \\ 1 & j = L_{size} \end{cases}$.

From equation (13), each agent has four neighbors. They form a little local environment in which agent can only sense.

3.3.4 The behavioral strategies for agents

To achieve its purposes, each agent has some behaviors. In MAPSO, each agent competes and cooperates with its neighbors to diffuse its useful information to the whole environment, and it can also use evolution mechanism of PSO and its knowledge. On the basis of such behaviors, three operators are designed for the agents.

Competition and cooperation operator: Suppose that the operator is performed on the agent *a* located at *(i,j)*, and $\alpha_{i,j} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, represented a real-valued vector in the search space. Suppose that $m = MinN_{i,j} = (m_1, m_2, \ldots, m_n)$ is the agent with minimum fitness value among its neighbors of N_{ij}, namely, $m \in N_{i,j}$, and $\forall \varepsilon \in N_{i,j}$, then $f(\varepsilon) \geq f(m)$. If agent a_{ij} satisfies equation (14), it is a winner; otherwise it is a loser.

$$f(\alpha_{ij}) \leq f(m) \qquad (14)$$

If α_{ij} is a winner, it can still live in the agent lattice, and its location in the search space will not change. If it is a loser, it must die, and its lattice –point will be occupied by a new agent New_{ij}. The new agent $New_{ij} = (\alpha_1', \alpha_2', \ldots, \alpha_n')$ is determined by the following two strategies in MAPSO.

Strategy 1: New_{ij} is determined by,

$$\alpha_k' = m_k + rand(0,1) \cdot (m_k - \alpha_k) \quad \text{k=1,2,…,n} \qquad (15)$$

where *rand(0,1)* is uniform random value in the range [0,1]. Suppose that $X_{\min} = (x_{\min 1}, x_{\min 2}, \ldots, x_{\min n})$ and $X_{\max} = (x_{\max 1}, x_{\max 2}, \ldots, x_{\max n})$ are vector of the lower and upper bound of the search space, respectively. If $\alpha_k' \geq x_{\max k}$, then $\alpha_k' = x_{\max k}$. If $\alpha_k' \leq x_{\min k}$, then $\alpha_k' = x_{\min k}$. From equation (15), the strategy 1, a kind of heuristic crossover is in favor of reserving some useful information of a loser.

Strategy 2: Enlightened by the inversion operation in evolutionary algorithms, $MinN_{ij}$ is firstly mapped on [0, 1] according to,

$$m_k' = \frac{(m_k - x_{k\min})}{(x_{k\max} - x_{k\min})}, \quad k = 1,2,\ldots,n \qquad (16)$$

Then, $New_{ij}' = (\beta_1', \beta_2', \ldots, \beta_n')$ is determined by

$$New_{i,j}' = (m_1', m_2', \ldots, m_{i1-1}', m_{i2}', m_{i2-1,}' \ldots, m_{i1+1,}' m_{i1,}' m_{i2+1,}' m_{i2+2,}' \ldots, m_n') \qquad (17)$$

where $1 < i_1 < n$, $1 < i_2 < n$, and $i_1 < i_2$. Finally, New_{ij} is obtained by mapping New_{ij}' back to $[X_{\min}, X_{\max}]$ according to

$$\alpha_k' = x_{\min k} + \beta_k' \cdot (x_{\max k} - x_{\min k}), \quad \text{k=1,2,…,n} \qquad (18)$$

From strategy 2, it has a function of random search and is better than random search in that it makes use of the information of a winner.

According to the above analysis, strategy 1 puts emphasis on exploitation while strategy 2 puts emphasis on exploration. Generally, strategy 2 is employed in the first iterative process in order to explore the global search space. With the process of iteration, MAPSO makes use of the strategy 1 to further exploit the local search space.

PSO operator: After competition and cooperation operator, MAPSO integrates evolution mechanism of PSO to quickly obtain the optimal solution. Since each agent can only sense its local environment, its behaviors of competition and cooperation can only take place between the agent and its neighbors. An agent interacts with its neighbors so that information is transferred. Owing to the ability to interact with its neighbors, the information is slowly diffused from the local agent lattice to the whole agent lattice. In order to quickly diffuse information to the whole agent lattice and enhance computational efficiency, MAPSO assimilates evolution mechanism of PSO. Hence, each agent adjusts its position in the search space according to its own experience, and the experience of the best agent among all the agents in the environment. Each agent renews its position by equations (9) and (10), and sets its position at the bound of the search space if having any violation.

Self-learning operator: Each agent is able to learn by using its knowledge in order to further enhance its ability for solving the problems. Enlightened by the reference [9], this paper proposes the self-learning operator which uses a small scale MAPSO to realize the behavior of using knowledge.

In self-learning operator of agent $\alpha_{i,j} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, first of all, a lattice-like environment, *sL*, is constructed. The size of *sL* is $sL_{size} \times sL_{size}$, where sL_{size} is an integer, and all agents, $sa_{i'j'}$, $i',j'=1,2,\ldots,sL_{size}$, are generated according to

$$s\alpha_{i,j'} = \begin{cases} \alpha_{i,j} & i' = 1,\ j' = 1 \\ New\alpha_{i,j'} & \text{otherwise} \end{cases} \tag{19}$$

where $New\alpha_{i,j'} = \left(e_{i,j',1}, e_{i,j',2}, \ldots, e_{i,j',n}\right)$ is determined by,

$$e_{i,j',k} = \begin{cases} x_{k\min} & \alpha_k * rand(1-sR, 1+sR) < x_{k\min} \\ x_{k\max} & \alpha_k * rand(1-sR, 1+sR) > x_{k\max} \\ \alpha_k * rand(1-sR, 1+sR) & \text{otherwise} \end{cases} \tag{20}$$

$k = 1, 2, \ldots, n$

where $sR \in [0,1]$ represents the search radius.

Next, the neighborhood competition and cooperation operator with evolution mechanism of PSO is iteratively performed on *sL*. Finally, a_{ij} is replaced by the agent with minimum fitness value found during the above process. Figure 2 describes the details of self-learning operator. In Figure 2, *sGen* is the number of generations, and *sBest(r)* is the agent with the minimum fitness value in the *r*th generation.

3.4 Mixed-Variable handing methods

In its basic form, the proposed MAPSO method can only handle continuous variables. To handle mixed variables, simply truncating the real values to discrete variables to calculate fitness value will not effect the search performance significantly. The truncation is only performed in evaluating the fitness function. That is, the swarm will 'fly' in a continuous search space regardless of the variable type.

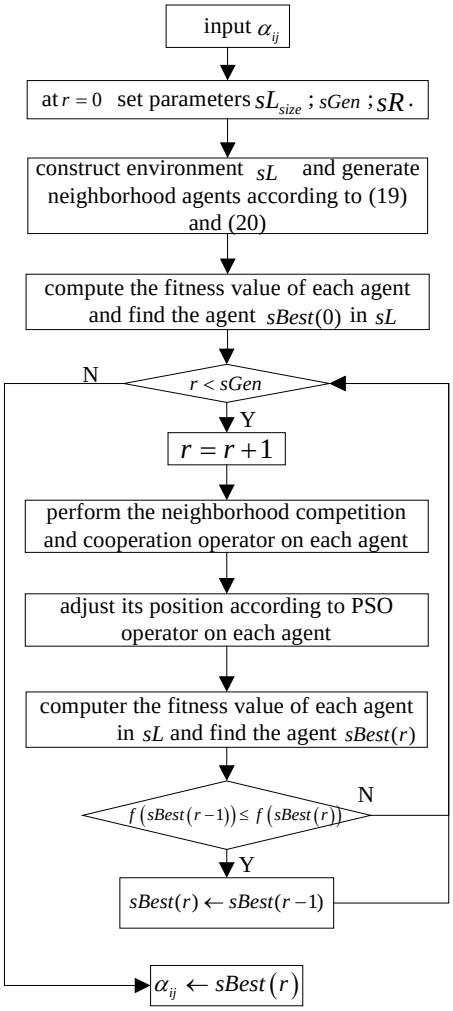

Figure 2: Flow chart of self-learning operator

For discrete variables of the *i*th particle x_i, the most straightforward way is to use the indices of the set of discrete variables with n_D elements:

$$x_i^D = [x_{i,1}^D, x_{i,2}^D, \ldots, x_{i,n_D}^D] \tag{21}$$

Let x_i^C denote the continuous variables with n_C elements:

$$x_i^C = [x_{i,1}^C, x_{i,2}^C, \ldots, x_{i,n_C}^C] \tag{22}$$

then particle *i* is denoted by $x_i = [x_i^C, x_i^D]$. For particle *i*, the index value *j* of the discrete variable $x_{i,j}^D$ is then optimized instead of the discrete value of the variable directly. In the population, the indices of the discrete variables of the *i*th particle should be the float point variables before truncation. That is, $j \in [1, n_{D+1})$, n_D is the number of discrete variables. Hence, the objective function of the *i*th particle x_i can be expressed as follows:

$$f(x_i) \quad i = 1, 2, \ldots, M \tag{23}$$

where

$$x_i = \begin{cases} x_{i,j} & x_{i,j} \in x_i^C \quad j = 1, \ldots, n_c \\ x_{i,INT(j)} & x_{i,INT(j)} \in x_i^D \quad j \in [1, n_D + 1) \end{cases} \tag{24}$$

where $x_i^C \in \mathbb{R}^{n_C}$ and $x_i^D \in \mathbb{R}^{n_D}$ denote the feasible subsets of continuous and discrete variables of particle x_i, respectively. INT*(x)* denotes the greatest integer less than the real value *x*.

3.5 Implementation of MAPSO for Reactive Power Optimal Dispatch

In order to reduce the computational cost, the self-learning operator is only performed on the agent with the minimum fitness value in each generation, but it has an important effect on the performance of MAPSO. The details of the overall algorithm are as follows.

Step 1: Input parameters of system, and specify the lower and upper boundaries of each variable.

Step 2:Generate a lattice-like environment *L*, and initialize randomly each agent.

Step 3: Evaluate the fitness value of each agent based on the Newton-Raphson power flow analysis results and the proposed mixed-variable handing methods..

Step 4: Update the time counter *t=t+1*.

Step 5:Perform the neighborhood competition and cooperation operator on each agent.

Step 6: Execute the PSO operator and further adjust its position on each agent according to (9) and (10).

Step 7: Evaluate the fitness value of each agent based on the Newton-Raphson power flow analysis results and the proposed mixed-variable handing methods..

Step 8: Find the best agent with the minimum fitness value, and then perform the self-learning operator.

Step 9: If one of the stopping criteria is satisfied then go to **Step 10**. Otherwise, go to **Step 4**.

Step 10: Output the agent with the minimum fitness value in the last generation.

4 NUMERICAL RESULTS

To verify the effectiveness and efficiency of the proposed MAPSO based reactive power optimization approach, two power systems are used as the test systems. The MAPSO has been implemented in Matlab 6.5 programming language and numerical tests are carried on a Pentium IV 2.0G computer.

Some parameters must be assigned before MAPSO is used to solve reactive power optimization dispatch. $L_{size} \times L_{size}$ is equivalent to the population size in traditional PSO, so L_{size} can be chosen from 5 to 10. In self-learning operator, with consideration of the computational cost, it is better to let sL_{size} be smaller than 5 and *sGen* range from 5 to 10. In all study systems, the following MAPSO parameters are used: L_{size} =6, i.e. the members of a particle are 36, *generations=300*, sL_{size}*=3*, *sR=0.25*, *sGen =10*, inertia weight factor *w=0.7298*, acceleration constant $\varphi_1 = \varphi_2 = 2.05$, both penalty factors in equation (8) are chosen, $\lambda_{Vi} = \lambda_{Gi} = 500$.

4.1 IEEE 30-Bus Power System

The system data and the variable limits are given in the reference [10]. The network consists of 48 branches, 6 generator-buses and 22 load-buses. 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 V θ -buses as follows: PV-buses: bus 2, 5, 8, 11,13, V θ -bus: bus 1. The transformer taps and the reactive power source installation are discrete variables with the changes step of 0.01 p.u..

To demonstrate the superiority of MAPSO, optimal results have been compared with various techniques available in literature, namely, standard genetic algorithm (SGA), adaptive GA (AGA) in Ref. [10], EP method in Ref. [11], Broyden's non-linear programming method [12] and PSO method. The initial conditions for all the methods are same and are given as follows:

$P_{load} = 2.834$ p.u. $\quad Q_{load} = 1.262$ p.u.

The initial generator bus voltages and transformer taps are set to 1.0. The total generations and power losses are obtained as follows:

$\sum P_G = 2.893857$ p.u. $\quad \sum Q_G = 0.980199$ p.u.

$P_{loss} = 0.059879$ p.u. $\quad Q_{loss} = -0.064327$ p.u.

The voltages outside the limits on three PQ-buses are given as follows:

$V_{26} = 0.932$; $V_{29} = 0.940$; $V_{30} = 0.928$

Table 1 summarizes the results of the optimal settings as obtained by different methods. These results show that the optimal dispatch solutions determined by the MAPSO lead to lower active power loss than that found by other methods, which confirms that the MAPSO is well capable of determining the global or near-global optimum dispatch solution. Moreover, these results show that maximum saving is obtained by the MAPSO method. At the same time, the MAPSO succeeds in keeping the dependent variables within their limits.

	$\sum P_G$	$\sum Q_G$	P_{loss}
Broyden	2.88986	0.93896	0.055860
SGA	2.88380	1.02774	0.049800
AGA	2.88326	0.66049	0.049260
EP	2.88362	0.87346	0.049630
PSO	2.88330	0.82500	0.049262
MAPSO	2.88270	0.81950	0.048747
	Q_{loss}	P_{SAVE}	$\%P_{SAVE}$
Broyden	-0.32304	0.00402	6.71000
SGA	-0.23426	0.01008	16.8400
AGA	-0.60151	0.01062	17.7400
EP	-0.38527	0.01025	17.1200
PSO	-0.22920	0.01062	17.6200
MAPSO	-0.22836	0.01113	18.5900

Table 1: Comparison of optimal transmission loss for different methods (p.u.)

Owing to the randomness in MAPSO, SGA and PSO, these algorithms are executed 50 times when applied to the test system. The best and worst reactive power dispatch solutions together with the associated power loss found by the three methods are tabulated in Table 2. MAPSO shows good consistency by keeping the difference between the best and worst solutions within

1%. In addition, the average execution times summarized in Table 2 show that MAPSO is faster than SGA and PSO in speed. Table 3 lists the best control variables found by the above three methods in the 50 run times. The optimization search procedures by SGA, PSO and MAPSO are shown in Figure 3. It can be seen that, by using the MAPSO method, system losses are drastically reduced at the early iterations and the total iterations for convergence can be reduced greatly.

Compared item	SGA	PSO	MAPSO
Best P_{loss}	0.049800	0.049262	0.048747
Worst P_{loss}	0.052140	0.050769	0.048759
Average P_{loss}	0.050810	0.049973	0.048751
Average time (sec)	156.34	59.21	41.93

Table 2: Comparison of simulation results in the IEEE 30-bus System (p.u)

	Bus	SGA	PSO	MAPSO
V_1	1	1.0751	1.0725	1.0780
V_2	2	1.0646	1.0633	1.0689
V_5	5	1.0422	1.0410	1.0468
V_8	8	1.0454	1.0410	1.0468
V_{11}	11	1.0337	1.0648	1.0728
V_{13}	13	1.0548	1.0597	1.0642
T_1	6~9	0.94	1.03	1.04
T_2	6~10	1.04	0.95	0.95
T_3	4~12	1.04	0.99	0.99
T_4	28~27	1.02	0.97	0.97
Q_3	3	0.00	0.00	0.00
Q_{10}	10	0.37	0.16	0.16
Q_{24}	24	0.06	0.12	0.12

Table 3: The values of control variables after optimization by SGA, PSO and MAPSO (p.u)

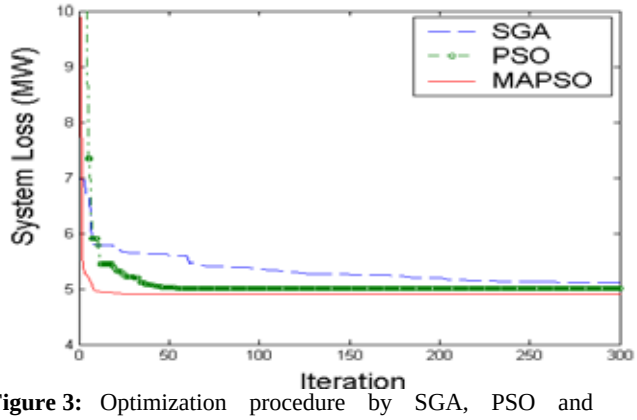

Figure 3: Optimization procedure by SGA, PSO and MAPSO for IEEE 30-bus system

4.2 Practical 118 Bus Power System

The proposed MAPSO method is applied to a practical 118-bus power system. The power system has 181 transmission elements, 17 generators for AVR control, 9 transformers with 9 to 25 positions and 14 reactive power source installation buses. At initial operating condition, system losses are 141.84 MW and represents about 2.72% of the total real-power generation in the system. There exist 11 deviations at the initial operating point. SGA, PSO and MAPSO are compared in 300 searching iterations.

To avoid any hazardous interpretation of optimization results, related to the choice of particular initial agents, we performed the simulation 50 times, starting from different agents randomly generated in the search space. Table 4 shows the best and worst loss values and the computational time. From Table 4, it can be seen that MAPSO can generate better solution with the bigger possibility than SGA and PSO. The average loss value by MAPSO is smaller than the best results by SGA and PSO. Figure 4 shows convergence characteristic for the 118-bus system by the three methods. It is clear for the figure that the solution obtained by MAPSO converges to high quality solutions at the early iterations (about 30 iterations). The average iteration to the best result by the MAPSO is about 35. However, the average iterations by SGA and PSO are 72 and 288, respectively. The average execution time by MAPSO is about 3 times faster than that by SGA. At the same time, the average execution time of MAPSO is 21% less than that of PSO. Considering together the characteristics of MAS and PSO, MAPSO performs better than PSO, both in the quality of the solution discovered and in the velocity of convergence, and simulation results show that MAPSO outperforms SGA and PSO, and is competent for the practical reactive power optimization problems.

Compared item	SGA	PSO	MAPSO
Best P_{loss}	1.332694	1.310471	1.26513
Worst P_{loss}	1.414267	1.348792	1.30147
Average P_{loss}	1.375215	1.321843	1.28215
Average time (sec)	335.54	144.46	119.35

Table 4: Comparison of simulation results in the practical 118-bus system (p.u.)

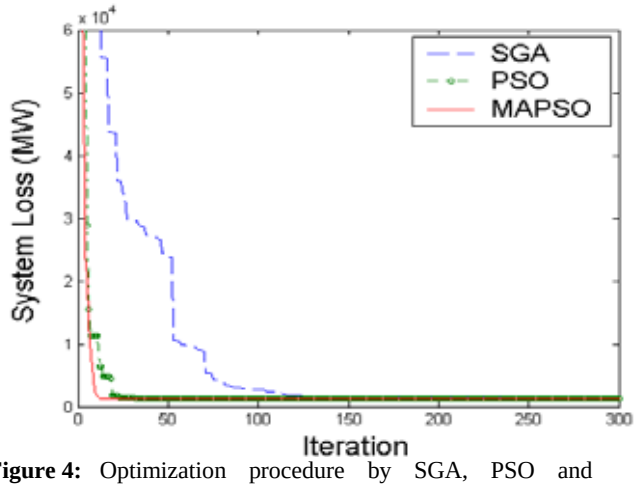

Figure 4: Optimization procedure by SGA, PSO and MAPSO for the practical 118-bus system

5 CONCLUSION

A MAPSO method has been developed for determination of the global or near-global optimum solution for optimal reactive power dispatch. The proposed method benefits mainly from the environment of the agent lattice and the behaviors of agents. In the environment, each agent can compete and cooperate with its neighbors, and further adjust its position in the search space according to PSO. Thus, each agent can quickly transfer its useful information to the global environment, and all agents can share the information after a process of diffusion. Owing to the three operators, the advantage of the MAPSO method is its ability in finding high quality solutions reliably with the faster convergence properties The performance of the proposed method demonstrated through its evaluation on the IEEE 30-bus power system and a practical 118-bus power system shows that MAPSO is able to undertake global search with a fast convergence rate and a feature of robust computation. From the simulation study, it has been found that MAPSO converges to the global optimum. The optimization strategy is general and can be used to other power system optimization problems as well.

6 APPENDIX

Nomenclature:

θ_{ij} voltage angle difference between buses i and j (rad)

B_{ij} transfer susceptance between bus i and j (p.u.)

f_Q active power loss in network (p.u.)

G_{ij} transfer conductance between bus i and j (p.u.)

g_k conductance of branch k (p.u.)

N_0 set of numbers of total buses excluding slack bus

N_B set of numbers of total buses

N_C set of numbers of possible reactive power source installation buses

N_D set of numbers of power demand buses

N_E set of numbers of network branches

N_G set of numbers of generator buses

N_i set of numbers of buses adjacent to bus i, including bus i

N_{PQ} set of numbers of PQ buses

N_{PV} set of numbers of PV buses

N_Q^{lim} set of numbers of buses on which injected reactive power outside limits

N_T set of numbers of transformer branches

N_V^{lim} set of numbers of buses on which voltages outside limits

P_{Di} demanded active power at bus i (p.u.)

P_{gi} injected active power at bus i (p.u.)

Q_{Ci} reactive power source installation at bus i (p.u.)

Q_{Di} demanded reactive power at bus i (p.u.)

Q_{Gi} injected reactive power at bus i (p.u.)

REFERENCES

[1] K. Y. Lee, Y. M. Park, and J. L. Ortiz, “A united approach to optimal real and reactive power dispatch,” IEEE Trans. Power Systems, vol. 104, pp. 1147-1153, May 1985.

[2] Y. C. Wu, A. S. Debs, and R. E. Marsten, “A direct nonlinear predictor-corrector primal-dual interior point algorithm for optimal power flows,” IEEE Trans. Power Systems, vol. 9, pp. 876-883, May 1994.

[3] J.A. Momoh, S. X. Guo, E. C. Ogbuobiri, and R. Adapa, “The quadratic interior point method solving power system optimization problems,” IEEE Trans. Power Systems, vol. 9, pp. 1327-1336, Aug. 1994.

[4] J. Kennedy, R. Eberhart, “Particle swarm optimization,” IEEE International Conference on Neural Networks, vol. 4, pp. 1942-1948, Nov. 1995.

[5] Z. L. Gaing, “Particle swarm optimization to solving the economic dispatch considering the generator constraints,” IEEE Trans. Power Systems, vol. 18, pp. 1187-1195, Mar. 2003.

[6] J. Liu, Y. Y. Tang, and Y. C. Cao, “An evolutionary autonomous agents approach to image feature extraction,” IEEE Trans. Evolutionary Computation, vol. 1, pp.141-158, Feb. 1997.

[7] R.C. Eberhart, Y. Shi, “Comparing inertia weights and constriction factors in particle swarm optimization,” In Proceedings of the Congress on Evolutionary Computing, pp. 84-88, 2000.

[8] W. Z. Zhong, J. Liu, M. Z. Xue, et al, “A Multiagent genetic algorithm for global numerical optimization,” IEEE Trans. System, Man and Cybernetics-Part B: Cybernetics, vol. 34, pp. 1128-1141, Apr. 2004.

[9] S. A. Kazarlis, S. E. Papadakis, J. B. Theocharis, and V. Petridis, “Microgenetic algorithms as generalized hill-climbing operators for GA optimization,” IEEE Trans. Evolutionary Computation, vol. 5, pp. 204–217, Jun. 2001.

[10] Q. H. Wu, Y. J. Cao, J. Y. Wen, “Optimal reactive power dispatch using an adaptive genetic algorithm,” Int. J. Electr Power & Energy Syst, vol. 20, pp. 563-569, Aug. 1998.

[11] L. L. Lai and J. T. Ma, “Application of evolutionary programming to reactive power planning – comparison with nonlinear programming approach,” IEEE Trans. Power Systems, vol. 12, pp. 198-204, Feb. 1997.

[12] D. B. Das and C. Patvardhan, “Reactive power dispatch with a hybrid stochastic search technique,” Int. J. of Electr Power & Energy Syst, vol. 24, pp. 731-736, Sep. 2002