AN ADAPTIVE PSO ALGORITHM FOR REACTIVE POWER OPTIMIZATION

Wen Zhang[1] Yutian Liu[1] Maurice Clerc[2]
(1. School of Electrical Engineering, Shangdong University, Jinan 250061, China;)
(2. France Télécom Recherche & Développement, 90000, Belfort, France)

ABSTRACT

The particle swarm optimization (PSO) algorithm is a new evolutionary computation method. It has been proved to be powerful but needs parameters predefined for a given problem by users. An adaptive particle swarm optimization (APSO) algorithm is proposed in this paper. It can adjust parameters automatically in optimization process to find the global optimum. Two power systems, the IEEE 30-bus system and a practical 125-bus power system with 64 control variables, have been tested. Simulation results of the proposed approach, compared with PSO algorithm, show that the APSO algorithm is more efficient in searching global optimization solution.

Index Terms—particle swarm optimization, evolutionary computation, reactive power optimization, power systems.

1. INTRODUCTION

Reactive power optimization is of value both in theory and application. The adjustment of generator voltages, transformer taps and shunt capacitors and inductors will control the reactive power distribution and affect the system voltage profile and system losses. The objective is to find proper adjustments of these variables that would maintain acceptable voltage levels throughout the system and minimize system losses. Because of its goal, reactive power optimization has commonly been formulated as a complicated constrained optimization problem with non-differentiable nonlinear objective functions. The conventional optimization algorithms[1], such as linear programming, nonlinear programming and interior point methods, have been widely used to solve this problem for decades. However, these conventional methods need many mathematical assumptions, such as differential properties of the objective functions and unique minima existing in problem domains and often trap in local optimal solutions.

In recent years, some artificial intelligence methods[2], such as genetic algorithm, simulated annealing, tabu search and evolutionary programming, have been applied to reactive optimization problem to lead to the global optimum. A new evolutionary computation method, called particle swarm optimization[3](PSO), has been proposed recently and proved to be a competitor in the field of optimization. PSO, inspired by social behavior of bird flocking or fish schooling, is a population based optimization tool. It is an algorithm for finding optimal regions of complex search spaces through the interaction of individuals in a population of particles. PSO has been applied to some power system problems[4-8], such as dynamic security border identification, state estimation, optimal power flow and reactive power and voltage control, and has been shown to perform well.

It should be noted that most of the above-mentioned papers are applications of simple PSO and the parameters of the algorithm should be selected carefully for efficient performance of PSO. In order to find a "good" set of parameters, the simple PSO has to be run several times with different parameter sets. To avoid this drawback, some attempts have been made to define an adaptive PSO[9-11], for example by using fuzzy rule or selection to modify a coefficient, but all of them do need some predefined parameters, such as the swarm size and weight coefficient. The choice of the PSO algorithm's parameters seems to be of utmost importance for the speed and efficiency of the algorithm. Despite recent research efforts, the selection of the algorithm parameters remains empirical to a large extent.

This paper proposes an adaptive particle swarm optimization (APSO) for reactive power optimization of power systems. It can adjust parameters automatically in optimization process. The effectiveness of the proposed algorithm has been showed by the simulation results of IEEE-30 bus system and a practical power system.

2. ADAPTIVE PARTICLE SWARM OPTIMIZATION

2.1 The Simple PSO

The PSO is initialized with a population of random, called particles, and searches for optima by following the current optimum particles. Each particle is, in fact, represents a potential solution to a problem, and is treated as a point in a D-dimensional space. For a given particle P_i, its position and velocity are represented as $x_i(t) = (x_{i,1}(t),...,x_{i,d}(t),...,x_{i,D}(t))$ and $v_i(t) = (v_{i,1}(t),...,v_{i,d}(t),...,v_{i,D}(t))$ respectively. The best previous position (the position giving the best fitness value) found so far by particle P_i is recorded as $p_i = (p_{i,1},...,p_{i,d},...,p_{i,D})$. The best previous position among the neighborhood (or all the particles in the population) is represented as $g_i = (g_{i,1},...,g_{i,d},...,g_{i,D})$. The particle's velocity and position are constantly adjusted according to the particle's experience and their companions' experience. Mathematically, the particles are manipulated according to the following equations[12]:

$$\begin{cases} v_{i,d}(t+1) = \chi(v_{i,d}(t) \\ \qquad + rand(0,\frac{\varphi}{2})(p_{i,d}(t) - x_{i,d}(t)) \\ \qquad + rand(0,\frac{\varphi}{2})(g_{i,d}(t) - x_{i,d}(t))) \\ x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1) \end{cases} \quad (1)$$

$$\begin{cases} \varphi > 4 \\ \chi = \dfrac{2}{\varphi - 2 + \sqrt{\varphi^2 - 4\varphi}} \end{cases} \quad (2)$$

where χ is called constriction coefficient, $rand(0,\frac{\varphi}{2})$ stands for a random number in $[0,\frac{\varphi}{2}]$.

2.2 The Adaptive PSO(APSO)

The PSO algorithm includes some tuning parameters that greatly influence the algorithm performance. The parameter values are usually determined through experimentation. The key idea of the APSO is to adapt three parameters automatically, the swarm size N, the coefficient φ_i and the neighborhood size h_i of each particle, based on the fitness values of particles during the process.

2.2.1 Some definitions

a) Comparing particles.

If the fitness value of P_i is smaller than that of P_j, we think that P_i is better than P_j.

$$f(p_i) \le f(p_j) \Leftrightarrow P_i \text{ better than } P_j \quad (3)$$

b) Best particle, worst particle

The best particle $P_{best,i}$ in the neighborhood of the particle P_i is "better than" any other in this neighborhood. More precisely,

$$\begin{cases} P_{best,i} = P_k \\ P_k \in \{P_k \in h_i, \forall P_l \in h_i, f(P_k) \le f(P_l)\} \end{cases} \quad (4)$$

There may be several particles like that and the best particle is chosen at random among them. For the worst particle $P_{worst,i}$, the similar definition is,

$$\begin{cases} P_{worst,i} = P_k \\ P_k \in \{P_k \in h_i, \forall P_l \in h_i, f(P_k) \ge f(P_l)\} \end{cases} \quad (5)$$

c) Improvement

The improvement for a given particle P_i is defined by

$$\delta(P_i) = \frac{f(P_i(t_0)) - f(P_i(t))}{f(P_i(t_0))} \quad (6)$$

d) Improvement threshold

The threshold "enough improvement" is first computed as follows. After the random swarm initialization, the worst fitness value and the best fitness value for the whole swarm are found. Then

the initial threshold is defined by

$$\Delta = 1 - \frac{f_{best}}{f_{worst}} \tag{7}$$

After that, during the process, it is updated by Eq.(8) when a particle is removed and by Eq.(9) when a particle is generated.

$$\Delta = \Delta\left(2 - \frac{1}{e^N}\right) \tag{8}$$

$$\Delta = \frac{\Delta}{\left(2 - \frac{1}{e^N}\right)} \tag{9}$$

The underlying idea is that when there has been enough improvement, this threshold can be increased, and vice versa. Also, the smaller the swarm, the easier it is to increase the improvement threshold.

2.2.2 Adaptive strategies

1) Swarm size *N*.

Swarm size(the number of particles in the swarm) affects the performance of PSO significantly. Too few particles will cause the algorithm to become stuck in a local minimum, while too many particles will slow down the algorithm, thus a balance between variety and speed must be sought. In simple PSO, it is constant. Some authors use 20, others 30. Some studies have been done showing the influence of the swarm size[13],[14], but there is no rule saying a given size is better than another for a given kind of problem. So it seems simpler to let the algorithm modify this size, from time to time, by removing some particles and adding new ones.

If a particle has an enough local improvement but is the worst local particle, remove the particle.

$$\begin{cases} P_i = P_{worst,i} \\ \delta\left(P_{best,i}\right) \geq \Delta \end{cases} \Rightarrow \text{remove } P_i \tag{10}$$

If a particle hasn't an enough local improvement but is the best local particle, generate a new particle.

$$\begin{cases} P_i = P_{best,i} \\ \delta\left(P_{best,i}\right) < \Delta \end{cases} \Rightarrow \text{add a particle} \tag{11}$$

2) Coefficient φ_i

The values of φ_i determine how much the particles are attracted by the best points found previously by itself and by its neighborhood. That is the convergence characteristic of the system can be controlled by φ_i . The constriction coefficient χ (calculated by Eq.(2)) prevents a buildup of velocity because of the particle inertia. Without χ, particles with buildup velocities might explore the search space, but loose the ability to fine-tune a result. Constricting the particle speed too much might, however, cripple the search space exploration. The setting for these parameters thus also determines the global versus local abilities of the PSO.

The more a particle improves itself, the smaller can be the part of the search space the particle explores. That is to increase the value of φ_i. As φ_i increases above 4, both χ and $\chi\varphi_i$ are indeed decreasing and so does velocity.

$$\begin{cases} m_i = \delta(P_i) - \Delta \\ m_i \geq 0 \Rightarrow \delta\varphi = (\varphi_{\max} - \varphi_i) m_i \\ \varphi_i = \varphi_i + \delta\varphi \end{cases} \tag{12}$$

The less a particle improves itself, the bigger should be the part of the search space the particle explores. That is to decrease φ_i.

$$\begin{cases} m_i = \delta(P_i) - \Delta \\ m_i < 0 \Rightarrow \begin{cases} \lambda = (\varphi_{\max} - \varphi_i)(\varphi_i - \varphi_{\min}) \\ \delta\varphi = (\varphi_i - \varphi_{\min})((1 - m_i)^{-\lambda} - 1) \end{cases} \\ \varphi_i = \varphi_i + \delta\varphi \end{cases} \tag{13}$$

3) Neighborhood size h_i

There are different neighborhood topologies, no clear and constant effects has been found[14]. The simple circular one is used. For example, if the particles are numbered(0,1,2,3), the neighborhood of size 3 for the particle3 is {2, 3, 0}.The lower value of h_i, the more slowly the best found positions are communicated between particles as particles only exchange information with neighboring particles in the topology. As h_i increases, the algorithm converges faster but might be trapped into local optimum for some problems.

If a particle is the local best and has improved itself enough, the particle doesn't need to ask so many

neighbors for more information; reduce its neighborhood size.

If a particle is the local best but hasn't improved itself enough, the particle need more information; increase its neighborhood size.

$$\begin{cases} P_i = P_{best,i} \\ \delta(P_i) \geq \Delta \Rightarrow \delta(h_i) = \delta(h_i) + \dfrac{\delta(h_i) - 1}{N - 1} \\ \delta(P_i) < \Delta \Rightarrow \delta(h_i) = \delta(h_i) + \dfrac{N - \delta(h_i)}{N - 1} \end{cases} \quad (14)$$

The modifications have a real effect only when their absolute values of "accumulation" are greater than or equal to 1. That is

$$\begin{aligned} \delta(h_i) \leq -1 &\Rightarrow \begin{cases} h_i = h_i - 1 \\ \delta(h_i) = 0 \end{cases} \\ \delta(h_i) \geq 1 &\Rightarrow \begin{cases} h_i = h_i + 1 \\ \delta(h_i) = 0 \end{cases} \end{aligned} \quad (15)$$

3. REACTIVE POWER OPTIMIZATION BASED ON APSO

3.1 Problem Formulation

The objective function of reactive power optimization is the network real power loss,

$$\begin{aligned} &\min P_{loss}(x_1, x_2) \\ &s.t.\ h(x_1, x_2) = 0 \\ &\quad x_{1\min} \leq x_1 \leq x_{1\max} \\ &\quad x_{2\min} \leq x_2 \leq x_{2\max} \end{aligned} \quad (16)$$

where $P_{loss}(x_1,x_2)$ is the network real power loss; the power flow equation, $h(x_1,x_2)=0$, is used as equality constraint; x_1 and x_2 are used as inequality constraints. $x_1=[V_G\ K_T\ Q_C]^T$ includes generator voltages(V_G), transformer taps(K_T), and shunt capacitors(Q_C) which are control variables and self-constrained; $x_2=[V_L\ Q_G]^T$ includes voltages of load-buses(V_L) and injected reactive powers of generators(Q_G) which are constrained by adding them as penalty terms to the objective function. The above objective function (1) is generalized as follows:

$$\min f = P_l + \lambda_V \sum_{\alpha} \Delta V_L^2 + \lambda_Q \sum_{\beta} \Delta Q_G^2 \quad (17)$$

where λ_V and λ_Q are the penalty factors. ΔV_L and ΔQ_G are the violation of load-bus voltage and generator reactive power. α and β are sets of buses which voltage and reactive power generation violate their constraints respectively.

3.2 Reactive power Optimization Using APSO

In the APSO algorithm, the fitness function is the generalized objective function (17). The population has N particles and each particle is a D-dimensional vector, where D is the number of control variables including generator voltages(V_G), transformer taps(K_T), and shunt capacitors (Q_C). The proposed algorithm can be expressed as follows:

Step 1: Reading the original parameters. Including the power system data and the APSO data;

Step 2: Initialization. Set the time counter t=0, place the particles randomly and uniformly distributed in the searching space and assign a random and uniformly chosen velocity for each particle. Moreover, we set $p_i(0) = g_i(0) = x_i(0)$;

Step 3: Time updating. t=t+1;

Step 4: Updating the parameter N, φ_i and h_i as described in 2.2.2;

Step 5: Velocity and position updating. Calculate $v_i(t+1)$ and $x_i(t+1)$ using equation(1);

Step 6: Calculating the fitness value. Calculate the fitness value for the new particle position $f(x_i(t+1))$ by power flow calculation;

Step 7: Updating p_i and g_i. Update p_i (t+1) with $x(t+1)$ if $f(x(t+1))$ better than $f(x(t))$; update $g_i(t+1)$ with best $p_i(t+1)$ in the neighborhood;

Step 8: Stop criterion. Loop to step 3 until a criterion is met, usually a sufficiently good fitness or a maximum number of function evaluations.

4. SIMULATION RESULTS

The proposed APSO method of reactive power optimization has been evaluated on IEEE 30 bus and

a practical 125 bus systems and has been compared with the simple PSO. The programs were written in C language and executed on a PC with a 128 MHz Pentium IV 1.6G CPU. The adopted parameters in the algorithms are given in Table 1. Typically, when the simple PSO is used, the swarm size and the neighborhood size are constant. Here N and h_i equal to 20 and 3 respectively. Coefficient φ_i is 4.1 and the constriction coefficient χ is thus 0.729 according to Eq.(2). The APSO parameters include the minimum, maximum and the initial value of N_i, φ_i and h_i. The stop criterion is 2000 function evaluations for both APSO and simple PSO.

TABLE 1: Parameter values for APSO

Min	3	4.0	3
Max	999	4.2	999
Initial	3	$(\varphi_{\min}+\varphi_{\max})/2$	1

4.1 IEEE 30-bus Power System

The IEEE-30 bus system consists of 41 branches, six generator buses and 22 load-buses. Four branches (4,12), (6,9), (6,10) and (27,28) are under load tap setting transformer branches. The possible reactive power source installation buses are 10, 15, 19 and 24. The variable limits are given in TABLE 2.

TABLE 2: Control Variable limits of IEEE 30

categories	Sites	Limits(p.u.)
K_T	Tap(4-12), Tap(6-9), Tap(6-10), Tap(27-28)	0.95-1.05 (5 taps)
Q_C	10, 15, 19, 24	0-0.5 (10 sets)
V_G	1, 2, 5, 8, 11, 13	1-1.1

All results presented in this paper are averages of 50 repeated runs. Table 3 shows the results of our experiments. The table lists the average network real power loss Ploss(p.u.), computation time Time(s) and the save percent of real power P_{save}%. The original real power loss is 0.05362(p.u.).

TABLE 3: Average results of 30 bus system

	Ploss(p.u.)	Time(s)	P_{save}%
APSO	0.04952	1.82	7.65
PSO	0.04993	2.27	6.88

The results in Tables 3 show that the active power loss by the APSO is lower than that by the PSO, which confirms that the APSO is well capable of determining the global or near-global optimum. In addition, the results summarized in Table 3 show that the proposed APSO is faster than PSO in speed. The reason is that the average number of particles in APSO 17 is smaller than the constant swarm size of PSO 20.

4.2 A Practical Power System

The proposed method is applied to a practical area system with 125 buses and 136 branches in China. The control variables include 35 on-load tap changers of transformer and 29 shunt capacitors. Because of the small reactive power reserve of generator dispatched by the area control center, the generator voltage is not considered as the control variable. The original real power loss is 0.3584.

TABLE 4: Average results of a practical system

	Ploss(p.u.)	Time(s)	P_{save}%
APSO	0.3096	27.5	13.6
PSO	0.3219	14.1	10.1

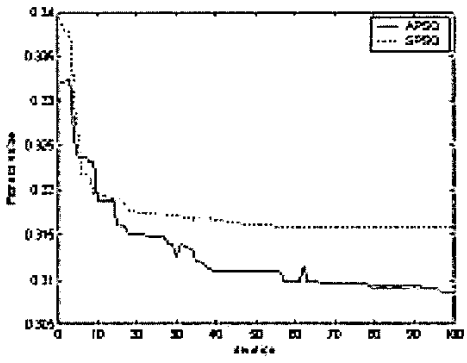

Fig.1 Convergent characteristics of algorithms

The average loss value by the proposed method is smaller than that by PSO as shown in Table 4. Typical convergence characteristics of APSO and PSO in Fig.1 showed that the APSO kept on optimizing towards a better fitness, whereas the PSO algorithm stagnated and flattened out with no further improvement on a local minimum.

5. CONCLUSION

This paper proposed the APSO algorithm for reactive power optimization, in which three parameters, the swarm size N, the coefficient φ_i and the neighborhood size h_i of each particle, were adapted automatically based on the fitness values of particles during the process. The simulation results showed that the APSO provided more reliable optimization in comparison with PSO method. The proposed method makes it possible for APSO to find new and better areas of the search space and to prevent premature convergence.

6. REFERENCES

[1] Momoh JA, El-Hawary ME, Adapa R. A review of selected optimal power flow literature to 1993, Part I: Nonlinear and quadratic programming approaches, Part II: Newton, linear programming and interior point methods. IEEE Trans.on Power Systems, vol. 14, pp. 96-111, 1999.

[2] Alves da Silva AP, Abrao PJ. Application of evolutionary computation in electric power systems, Proceedings of International Congress on Evolutionary Computation, Hawaii, USA, 2002, pp.1057-1062.

[3] Kennedy J, Eberhart RC. Particle swarm optimization, Proceedings of International Conference on Neural Networks, Australia, 1995, pp.1942-1948.

[4] Kassabalidis I. N., El-Sharkawi M.A.,et al. Dynamic security border identification using enhanced particle swarm optimization. IEEE Trans. on Power Systems, vol.17, pp. 723-729, 2002.

[5] Shigenori Naka, Takamu Genji. A hybrid particle swarm optimization for distribution state estimation. IEEE Trans. on Power Systems, vol.18, pp. 60-68, 2003.

[6] Hirotaka Yoshida, *et al.* A particle swarm optimization for reactive power and voltage control considering voltage security assessment. IEEE Trans. on Power Systems, vol.15, pp.1232-1239, 2000.

[7] Abido MA. Optimal power flow using particle swarm optimization. Electric Power and Energy Systems, vol.24, pp.563-571, 2002.

[8] Abido MA. Optimal design of power-system stabilizers using particle swarm optimization. IEEE Trans. on Energy Conversion, vol.17, pp. 406-413, 2002.

[9] Shi Y, Eberhart RC. Fuzzy adaptive particle swarm optimization. Proceedings of International Congress on Evolutionary Computation, Korea, 2001, pp.101-106.

[10] Angeline PJ. Using selection to improve particle swarm optimization. Proceedings of International Congress on Evolutionary Computation, Alaska, USA, 1998, pp.84-89.

[11] M. Clerc, "The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization," presented at Congress on Evolutionary Computation, Washington DC, 1999.

[12] Clerc M., Kennedy J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation, vol.6, pp.58-73, 2002.

[13] F. Van den Bergh and A. P. Engelbrecht, "Effects of Swarm Size on Cooperative Particle Swarm Optimisers," GECCO 2001, San Francisco, USA, 2001.

[14] J. Kennedy, "Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance," Congress on Evolutionary Computation, Washington D.C., 1999.