Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 761403, 14 pages
http://dx.doi.org/10.1155/2014/761403

Research Article

A Novel Adaptive Elite-Based Particle Swarm Optimization Applied to VAR Optimization in Electric Power Systems

Ying-Yi Hong,[1] Faa-Jeng Lin,[2] Syuan-Yi Chen,[3] Yu-Chun Lin,[2] and Fu-Yuan Hsu[1]

[1] *Department of Electrical Engineering, Chung Yuan Christian University, Chung Li City 320, Taiwan*
[2] *Department of Electrical Engineering, National Central University, Chung Li City 320, Taiwan*
[3] *Department of Applied Electronics Technology, National Taiwan Normal University, Taipei 320, Taiwan*

Correspondence should be addressed to Ying-Yi Hong; yyh10632@yahoo.com.tw

Received 5 February 2014; Accepted 30 March 2014; Published 22 May 2014

Academic Editor: Ker-Wei Yu

Particle swarm optimization (PSO) has been successfully applied to solve many practical engineering problems. However, more efficient strategies are needed to coordinate global and local searches in the solution space when the studied problem is extremely nonlinear and highly dimensional. This work proposes a novel adaptive elite-based PSO approach. The adaptive elite strategies involve the following two tasks: (1) appending the mean search to the original approach and (2) pruning/cloning particles. The mean search, leading to stable convergence, helps the iterative process coordinate between the global and local searches. The mean of the particles and standard deviation of the distances between pairs of particles are utilized to prune distant particles. The best particle is cloned and it replaces the pruned distant particles in the elite strategy. To evaluate the performance and generality of the proposed method, four benchmark functions were tested by traditional PSO, chaotic PSO, differential evolution, and genetic algorithm. Finally, a realistic loss minimization problem in an electric power system is studied to show the robustness of the proposed method.

1. Introduction

Particle swarm optimization (PSO) is a population-based optimization method [1]. PSO attempts to mimic the goal-searching behavior of biological swarms. A possible solution of the optimization problem is represented by a particle. The PSO algorithm requires iterative computation, which consists of updating the individual velocities of all particles at their corresponding positions. The PSO algorithm has some merits: (1) it does not need the crossover and mutation operations used in genetic algorithms; (2) it has a memory so that the elite solutions can be retained by all particles (solutions); and (3) the computer code of PSO is easy to develop. The PSO has been successfully applied to solve many problems, for example, reconfiguration of shipboard power system [2], economic dispatch [3], harmonic estimation [4], harmonic cancellation [5], and energy-saving load regulation [6].

PSO is designed to conduct global search (exploration) and local search (exploitation) in the solution space. The former explores different extreme points of the search space by jumping among extreme particles. In contrast, the latter searches for the extreme particle in a local region. The particles ultimately converge to the same extreme position. However, when a problem involves a great number of unknowns, the PSO generally requires a large number of particles in order to gain a global optimal solution (position). Consequently, achieving the coordination between the global search and the local search becomes more difficult.

To overcome the limitation described above, van den Bergh and Engelbrecht presented the cooperative PSO (COPSO) based on the dimension partition [7]. Nickabadi et al. focused on several well-known inertia weighting strategies and gave a first insight into various velocity control approaches [8]. The adaptive PSO (APSO) presented by Zhan et al. utilized the information on population distribution to identify the search status of particles; the learning strategy tries to find an elitist and then search the global best position in an iterative step [9]. Caponetto et al. presented a chaos algorithm to compute the inertia weight of preceding position of a particle and a self-tuning method to compute learning factors [10]. Their method introduces a chaos parameter, which is determined by a logistic map. The learning factors

are adaptive according to the fitness value of the best solution during the iterations. To enhance the scalability of PSO, José and Enrique presented two mechanisms [11]. First, a velocity modulation is designed to guide the particles to the region of interest. Second, a restarting modulation redirects the particles to promising areas in the search space. Deng et al. presented a two-stage hybrid swarm intelligence optimization algorithm incorporating with the genetic algorithm, PSO, and ant colony optimization to conduct rough searching and detailed searching by making use of the advantages of the parallel and positive feedback [12]. Li et al. presented a cooperative quantum-behaved PSO, which produces several particles using Monte Carlo method first and then these particles cooperate with one another to converge to the global optimum [13]. Recently, Arasomwan and Adewumi tried many chaotic maps, in additional to the traditional logistic map, which attempted to enhance the performances of PSO [14]. Yin et al. incorporated the crossover and mutation operators in PSO to enhance diversity in the population; besides, a local search strategy based on constraint domination was proposed and incorporated into the proposed algorithm [15].

The variants of PSO generally have the following limitations.

(1) The solutions are likely to be trapped in the local optima and undesirable premature situations because it relies heavily on the learning factors and inertia weight.

(2) Compared with the traditional PSO, variants of PSO require longer CPU times to compute the global optimum due to the extra treatment in exploration and exploitation.

(3) The iterative process is unstable or even fails to converge because of the ergodic and dynamic properties, for example, chaos sequence.

Based on the above literature review, a novel PSO is proposed in this paper. This novel adaptive elite-based PSO employs the mean of all particles and standard deviation of distances between any two particles. In addition to global and local searches, a new mean search is introduced as an alternative method for finding the global optimum. The standard deviation of distances among all particles is utilized to prune the distant particles. The best particle is cloned to replace the discarded particles. This treatment ensures a stable iterative process. The increase in the computational burden of this enhancement is trivial.

The rest of this paper is organized as follows. Section 2 provides the general PSO algorithm. Section 3 proposes the novel elite-based PSO. In Section 4, four benchmark functions are utilized to validate the proposed method. Comparative studies among different optimization methods (traditional PSO, chaotic PSO [16], differential evolution [17], and genetic algorithm) are given. The results of its application to the real problem of loss minimization in an electric power system are presented. Section 5 draws conclusions.

2. General PSO Algorithm

PSO, which is an evolutionary optimization method for minimizing an objective $f(X)$, reflects the behavior of flocking birds or schooling fish. PSO comprises a population of particles iteratively updating the empirical information about a search space. The population consists of many individuals that represent possible solutions and are modeled as particles moving in a ψ-dimensional search space. Let the superscript t be the iteration index. The position and velocity of each particle p are updated as follows [1]:

$$V_p^{t+1} = \omega^t \times V_p^t + C_1 \times r_1^t \times \left(p_{\text{best}}^t - X_p^t\right) + C_2 \times r_2^t \times \left(g_{\text{best}}^t - X_p^t\right), \quad (1)$$

$$X_p^{t+1} = X_p^t + V_p^{t+1}, \quad (2)$$

where vectors X_p and V_p are the ψ-dimensional position and velocity of particle p, $p = 1, 2, \ldots, P$ (number of population size), respectively. The inertia weight ω is designed to copy the previously updated features to the next iteration. If $\omega = 1$, then the preceding V_p^t has a strong impact on V_p^{t+1}. p_{best} and g_{best} are the best position of a particle and the best known position found by any particle in the swarm so far, respectively. The random numbers r_1 and r_2 are between 0 and 1. Learning factors C_1 and C_2 are positive constants constrained within the range 0 and 2 such that $C_1 + C_2 \leq 4$. The products $C_1 \times r_1$ and $C_2 \times r_2$ thus stochastically control the overall velocity of a particle. In this paper, X_p^{t+1} denotes the pth vector (particle) at the $(t+1)$th iteration and V_p^{t+1} can be regarded as the $(t+1)$th updated value (ΔX_p^t) that is added to X_p^t.

The second term in (1) is designed to conduct the global search (exploration) and the third term in (1) describes the local search (exploitation) in the solution space, as described in Section 1. The global and local searches should be coordinated in order to gain the global optimum and avoid premature results. The inertia weight ω decreases linearly throughout the iterations and governs the momentum of the particle by weighting the contribution of the particle's previous velocity to its current velocity. A large ω is designed for global searches, whereas a small ω is used for local searches. Consequently, in the earlier stages of the search process, a large ω is preferred, while a small value is preferred in the later stages.

3. Proposed Adaptive Elite-Based PSO

The proposed novel PSO has the following three features. (1) An extra mean search is introduced to coordinate exploration and exploitation; (2) distant particles are pruned and the best particle is cloned to ensure that the iterative process is stable; and (3) complicated computation is avoided and CPU time is thus reduced. The proposed adaptive elite-based PSO, thus, performs 2 main tasks: mean search and particle pruning/cloning. The effects of these two tasks decline as the number of iterations increases. The limitations

of the traditional variants of PSO described in Section 1 can therefore be eliminated.

3.1. Mean Search. The proposed method is based on the mean of the particles and standard deviation of the distances between any two particles in the tth iterations. Equation (1) is herein modified to the following:

$$V_p^{t+1} = \omega^t \times V_p^t + C_1^t \times r_1^t \times \left(p_{\text{best}}^t - X_p^t\right) + C_2^t \times r_2^t \times \left(g_{\text{best}}^t - X_p^t\right) + C_3^t \times r_3^t \times \left(g_{\text{best}}^t - X_m^t\right), \tag{3}$$

where the inertia weight ω^t is decreased linearly from 0.5 in the first iteration to 0.3 in the final iteration. In addition to r_1 and r_2, the term r_3 is also a random number between 0 and 1. X_m^t is a ψ-dimensional vector of the mean values of all particles. The last term introduced in (3) is used to coordinate the global and local searches as well as the particle pruning/cloning task later.

Because $C_1 + C_2 \leq 4$ [18], the new learning factor is defined by

$$C_3^t = 4 - C_1^t - C_2^t, \tag{4}$$

where

$$C_1^t = C_2^t = 1 + \frac{1}{1 + \exp\left(\alpha / F\left(g_{\text{best}}^t\right)\right)} \tag{5}$$

and $\alpha = F(g_{\text{best}}^1)$ [10]. Equation (4) ensures that C_3^t decreases gradually to zero and (3) becomes (1) when the iterative process converges.

3.2. Particle Pruning/Cloning. X_m^t introduced in Section 3.1 is used to develop the second task of the adaptive elite strategy: pruning the distant particles. The standard deviation σ^t of all distances among particles in the tth iteration is evaluated. Suppose that all distances follow a Gaussian distribution. Then $X_m^t \pm 3\sigma^t$ covers 99.7% of all particles. Let $\eta^t = 3 - 2C_3^t$. Because C_3^t in (4) decreases approximately from 1.48 to zero, the values of η^t will be increased from 0.04 to 3. Hence, the second task in the adaptive elite-based PSO is initially to prune distant particles and finally cover all particles. Restated, the particles outside the range of $X_m^t \pm \eta^t \sigma^t$ are pruned. The reduced number of particles are substituted by cloning g_{best}^t. That is,

$$X_p^{t+1} = g_{\text{best}}^t + V_p^{t+1}. \tag{6}$$

As shown in Figure 1, X_m^t is a virtual particle which includes the mean of all particles. Consider the range $X_m^t \pm \sigma^t$. Only seven particles are inside this range and three are outside it. Consider X_p^t which is outside the range and is pruned. X_p^t is hence substituted by g_{best}^t, which is denoted by $X_p'^t$.

3.3. Algorithmic Steps. The proposed method can be implemented using the following algorithmic steps.

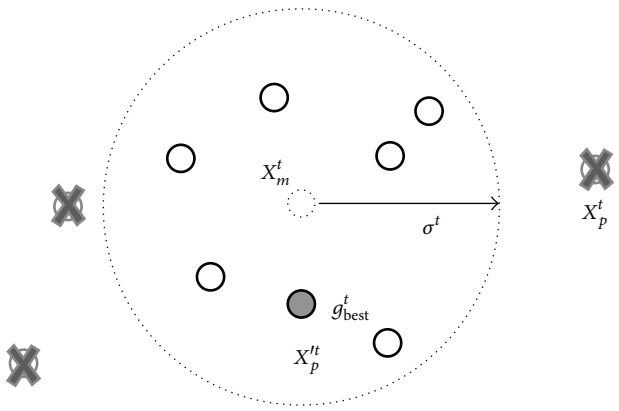

Figure 1: Pruning and cloning particles.

Step 1. Input the objective, unknowns, and the number of population size.

Step 2. Give randomly the particles (vectors of unknowns). Let $t = 1$.

Step 3. Calculate all objective values for all particles.

Step 4. If the iterative process meets the convergence criteria, then stop.

Step 5. Find the p_{best}^t and g_{best}^t from all particles according to the objective values of all particles.

Step 6. Calculate C_1^t and C_2^t using (5). Compute C_3^t using (4).

Step 7. Evaluate X_m^t, σ^t, and η^t.

Step 8. If a particle lies outside the range $X_m^t \pm \eta^t \sigma^t$, then it is pruned and $X_p^t = X_p'^t = g_{\text{best}}^t$.

Step 9. Calculate the velocities of all particles using (3).

Step 10. Update all particles using (2).

Step 11. Let $t = t + 1$ and go to Step 3.

3.4. Proof of Convergence of the Proposed Method. Let ω be a real number. Equations (2) and (3) are rewritten in the discrete-time matrix form as follows [19, 20]:

$$Y(t+1) = A(t)\,Y(t) + B(t)\,U(t), \tag{7}$$

where

$$Y(t) = \begin{bmatrix} X_p^t \\ V_p^t \end{bmatrix} = \begin{bmatrix} X_p(t) \\ V_p(t) \end{bmatrix};$$

$$A(t) = \begin{bmatrix} -\left(-1 + C_1^t r_1^t + C_2^t r_2^t + \dfrac{1}{P} C_3^t r_3^t\right) & \omega^t \\ -\left(C_1^t r_1^t + C_2^t r_2^t + \dfrac{1}{P} C_3^t r_3^t\right) & \omega^t \end{bmatrix};$$

$$B(t) = \begin{bmatrix} C_1^t r_1^t & C_2^t r_2^t + C_3^t r_3^t & -\frac{1}{P} C_3^t r_3^t \\ C_1^t r_1^t & C_2^t r_2^t + C_3^t r_3^t & -\frac{1}{P} C_3^t r_3^t \end{bmatrix};$$

$$U(t) = \begin{bmatrix} p_{\text{best}}^t \\ g_{\text{best}}^t \\ \sum_{k=1, k\neq p}^{P} X_k^t \end{bmatrix}. \tag{8}$$

Adding a constant vector $-[\delta_1 \quad \delta_2]^T$ into both sides of (7), one can yield

$$\begin{aligned} Y(t+1) - [\delta_1 \quad \delta_2]^T &= A(t) Y(t) + B(t) U(t) - [\delta_1 \quad \delta_2]^T \\ &= A(t) \left[Y(t) - [\delta_1 \quad \delta_2]^T \right] \\ &\quad + A(t) [\delta_1 \quad \delta_2]^T + B(t) U(t) \\ &\quad - [\delta_1 \quad \delta_2]^T. \end{aligned} \tag{9}$$

Define

$$\begin{aligned} \widehat{Y}(t+1) &\equiv Y(t+1) - \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} X_p(t+1) - \delta_1 \\ V_p(t+1) - \delta_2 \end{bmatrix}, \\ \widehat{Y}(t) &\equiv Y(t) - \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} X_p(t) - \delta_1 \\ V_p(t) - \delta_2 \end{bmatrix}. \end{aligned} \tag{10}$$

Thus, (9) can be rewritten as follows:

$$\begin{aligned} \widehat{Y}(t+1) \equiv A(t) \widehat{Y}(t) &+ A(t) [\delta_1 \quad \delta_2]^T + B(t) U(t) \\ &- [\delta_1 \quad \delta_2]^T. \end{aligned} \tag{11}$$

If the condition $A(t)[\delta_1 \quad \delta_2]^T + B(t)U(t) - [\delta_1 \quad \delta_2]^T = 0$ holds, (11) becomes a linear system as

$$\widehat{Y}(t+1) \equiv A(t) \widehat{Y}(t). \tag{12}$$

The equilibrium point $\widehat{Y}_{\text{eq}}(t) = 0$ must occur at $\widehat{Y}(t+1) = \widehat{Y}(t)$ as $t \to \infty$ for any $A(t)$. According to the condition $A(t)[\delta_1 \quad \delta_2]^T + B(t)U(t) - [\delta_1 \quad \delta_2]^T = 0$, one can obtain

$$\begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} \dfrac{C_1^t r_1^t p_{\text{best}}^t + \left(C_2^t r_2^t + C_3^t r_3^t\right) g_{\text{best}}^t - \left(C_3^t r_3^t / P\right) \sum_{k=1, k\neq p}^{P} X_k^t}{C_1^t r_1^t + C_2^t r_2^t + \left(C_3^t r_3^t / P\right)} \\ 0 \end{bmatrix}. \tag{13}$$

Since $\widehat{Y}(t) \equiv Y(t) - [\delta_1 \quad \delta_2]^T$ and the equilibrium point $\widehat{Y}_{\text{eq}}(t) \equiv Y_{\text{eq}}(t) - [\delta_1 \quad \delta_2]^T = 0$, this implies

$$\begin{aligned} Y_{\text{eq}}(t) &= \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} \\ &= \begin{bmatrix} \dfrac{C_1^t r_1^t p_{\text{best}}^t + \left(C_2^t r_2^t + C_3^t r_3^t\right) g_{\text{best}}^t - \left(C_3^t r_3^t / P\right) \sum_{k=1, k\neq p}^{P} X_k^t}{C_1^t r_1^t + C_2^t r_2^t + \left(C_3^t r_3^t / P\right)} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} g_{\text{best}}^t \\ 0 \end{bmatrix} \end{aligned} \tag{14}$$

which is true only when $p_{\text{best}}^t = g_{\text{best}}^t = X_k^t$ as $t \to \infty$. Hence, the equilibrium point is

$$Y_{\text{eq}}(t) = \begin{bmatrix} g_{\text{best}}^t \\ 0 \end{bmatrix}. \tag{15}$$

In order to guarantee $Y_{\text{eq}}(t)$ of the discrete-time linear system as shown in (12) to be asymptotically stable, the following necessary and sufficient conditions are obtained using the classical Jury criterion:

$$\begin{aligned} C_1^t r_1^t + C_2^t r_2^t + \frac{C_3^t r_3^t}{P} &> 0, \\ 1 + \omega^t - \frac{C_1^t r_1^t + C_2^t r_2^t + \left(C_3^t r_3^t / P\right)}{2} &> 0 \\ \left|\omega^t\right| &< 1. \end{aligned} \tag{16}$$

Since random numbers r_1^t, r_2^t, and r_3^t are between 0 and 1, the above stability conditions are equivalent to the following set of parameter selection heuristics, which guarantee convergence for the proposed adaptive elite-based PSO algorithm:

$$\begin{aligned} 0 < C_1^t + C_2^t + \frac{C_3^t}{P} &< 4, \\ \frac{C_1^t + C_2^t + \left(C_3^t / P\right)}{2} - 1 < \omega^t &< 1. \end{aligned} \tag{17}$$

According to (17), the convergence of the proposed adaptive elite-based PSO can be guaranteed.

4. Simulation Results

The performance of the proposed adaptive elite-based PSO is verified by four benchmark test functions in this section. Traditional PSO, chaotic PSO (CPSO), differential evolution (DE), and genetic algorithm (GA) are used to investigate the benchmark functions for comparative studies [21]. The number of dimensions of each benchmark function is 20 (meaning that the number of unknowns $n = 20$). The number of particles (population size) is also 20 in all methods. The mutation rate and crossover rate in DE are 0.8 and 0.5, respectively; while those in GA are 0.01 and 1.0, respectively. Ten simulations are conducted by each method to verify the optimality and robustness of the algorithms. Two convergence criteria are adopted: convergence tolerance and fixed maximum number of iterations. Five manuscript codes are developed using MATLAB 7.12 to minimize the benchmark functions. The CPU time for studying these four test functions is evaluated using a PC with Intel (R) Core i7 (CPU@3.4 GHz, RAM 4 GB). Finally, the problem of loss minimization in an electric power system is used to validate the proposed method.

4.1. Benchmark Testing: Sphere Function. First, the sphere function is tested as follow:

$$f_1(\vec{x}) = \sum_{i=1}^{n} x_i^2 \quad -100 < x_i < 100, \tag{18}$$

Table 1: Performance of different methods used to minimize f_1 based on 10 simulation results (convergence tolerance = 0.001).

		Proposed method	PSO	CPSO	DE	GA
Shortest time	f_1	$7.74e-4$	$9.16e-4$	$8.93e-4$	$7.16e-4$	—
	Iterations	203	218	203	294	—
	Time (s)	0.0343	0.0129	0.0146	0.1854	—
Mean	f_1	$8.80e-4$	$9.40e-4$	$9.06e-4$	$8.95e-4$	—
	Iterations	250.1	239.9	241	267.8	—
	Time (s)	0.0391	0.0141	0.0168	0.1935	—
Longest time	f_1	$8.68e-4$	$9.76e-4$	$9.72e-4$	$9.87e-4$	—
	Iterations	302	263	208	247	—
	Time (s)	0.0439	0.0149	0.0245	0.2062	—

Table 2: Performance of different methods used to minimize f_1 based on 10 simulation results (number of iterations = 500).

		Proposed method	PSO	CPSO	DE	GA
Best f_1	f_1	$3.30e-12$	$2.65e-11$	$1.02e-11$	$2.03e-09$	7.1996
	Time (s)	0.0616	0.0232	0.0271	0.6265	1.3834
Mean	f_1	$1.41e-10$	$1.70e-09$	$4.09e-07$	$1.22e-08$	18.4275
	Time (s)	0.0619	0.0240	0.0269	0.6209	1.3143
Worst f_1	f_1	$1.00e-09$	$8.20e-09$	$1.54e-06$	$4.10e-08$	45.4336
	Time (s)	0.0620	0.0234	0.0262	0.6140	1.3328

where $n = 20$ and $\widehat{x} = (x_1, x_2, \ldots, x_{20}) \cdot f_1(\widehat{x})$ is a unimodal test function whose optimal solution is $f_1(\widehat{x}) = 0$ and $x_1 = x_2 = \cdots = x_{20} = 0$. First, the convergence tolerance is set to 0.001. Table 1 shows the shortest, mean and the longest CPU times obtained from 10 simulations by the five methods. As illustrated in Table 1, all methods can approach similar solutions except for GA (which fails to converge). DE requires the longest CPU times. In the conditions of mean and the longest CPU times, the proposed method yields smaller values of $f_1(\widehat{x})$ than the other methods. Figure 2 shows the iteration excursions of the different methods with their shortest CPU times.

Table 2 shows the best, mean and the worst $f_1(\widehat{x})$ obtained by running 500 iterations using all methods. It can be found that the proposed method always yields the smallest $f_1(\widehat{x})$ among the five methods. Figures 3 and 4 present the parameter C_3 and the standard deviation σ obtained using the proposed method in the case with the best $f_1(\widehat{x})$, respectively. The values of C_3 decrease quickly in the first 30 iterations and the standard deviation oscillates while decreasing to zero near the 200th iteration.

Generally, the CPU time required by the proposed method is longer than those taken by PSO and CPSO but shorter than those taken by DE and GA.

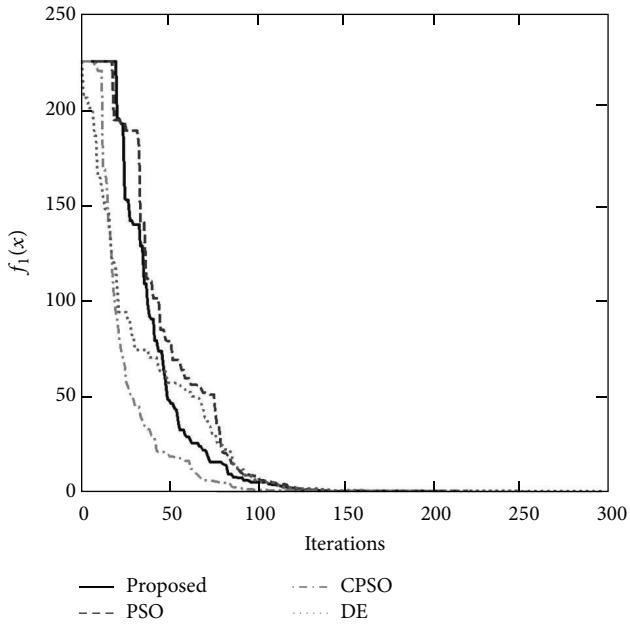

Figure 2: Iteration excursions with the shortest CPU times for different methods used to minimize f_1.

4.2. Benchmark Testing: Rosenbrock Function. This subsection employs the Rosenbrock function for testing as follows:

$$f_2(\widehat{x}) = \sum_{i=1}^{n-1} 100 \times \left(x_{i+1} - x_i^2\right)^2 + \left(1 - x_i\right)^2 \quad -2.048 < x_i < 2.048, \tag{19}$$

where $n = 20$ and $\widehat{x} = (x_1, x_2, \ldots, x_{20})$. Each contour of the Rosenbrock function looks roughly parabolic. Its global minimum is located in the valley of the parabola (banana valley). Since the function changes little in the proximity of the global minimum, finding the global minimum is considerably difficult. $f_2(\widehat{x})$ is also a unimodal test function whose optimal solution is $f_2(\widehat{x}) = 0$ and $x_1 = x_2 = \cdots = x_{20} = 0$. First, the convergence tolerance is set to 0.001. Table 3

Table 3: Performance of different methods used to minimize f_2 based on 10 simulation results (convergence tolerance = 0.001).

		Proposed method	PSO	CPSO	DE	GA
Shortest time	f_2	$9.99e-04$	$9.99e-04$	—	$9.73e-04$	—
	Iterations	6543	6931	—	1167	—
	Time (s)	0.6641	0.3037	—	0.7361	—
Mean	f_2	$9.97e-04$	$9.99e-04$	—	$9.13e-04$	—
	Iterations	7681.3	8532.3	—	1274.7	—
	Time (s)	0.7787	0.4415	—	1.0411	—
Longest time	f_2	$9.99e-04$	$9.98e-04$	—	$8.68e-04$	—
	Iterations	8759	9099	—	1680	—
	Time (s)	0.8879	0.6239	—	1.4807	—

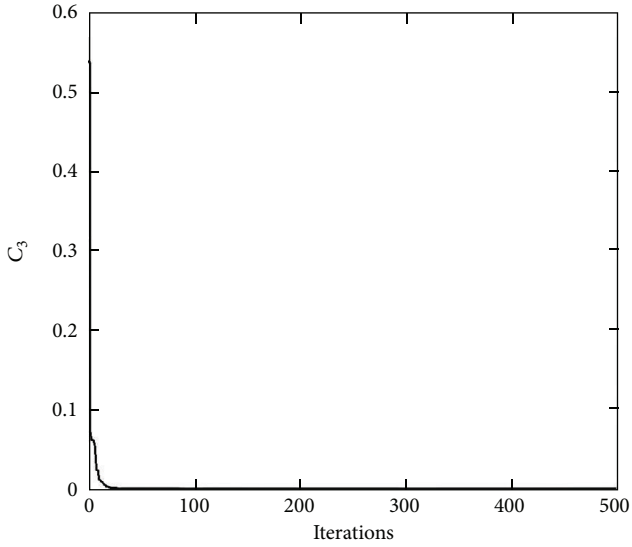

Figure 3: Iterative performance in terms of C_3 (sphere function).

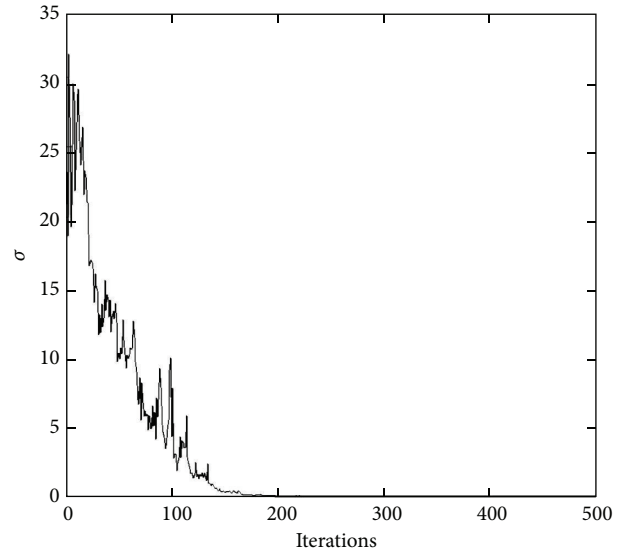

Figure 4: Iterative performance in terms of standard deviation (sphere function).

shows the shortest, mean and the longest CPU times obtained by performing 10 simulations using the five methods. As illustrated in Table 3, all methods yield similar solutions except for CPSO and GA (which fail to converge). The CPU times required by the proposed method are between those taken by PSO and DE. It could be found that DE requires fewer iterations but longer CPU times in all studies. All solutions (optimality), from the viewpoints of the shortest, mean and the longest CPU times, are similar. Figure 5 shows the iteration excursions of the different methods (the cases with their shortest CPU times).

Table 4 shows the best, mean and the worst $f_2(\widehat{x})$ obtained by running 1000 iterations using all methods. Since the proposed AEPSO and PSO require 6500~8000 iterations to find the global optimum of $f_2(\widehat{x})$, they only reach a near optimal solution in the 1000th iterations. Figures 6 and 7 display the parameter C_3 and standard deviation σ of the proposed method in the case with the best $f_2(\widehat{x})$, respectively. The values of C_3 decrease quickly in the first 40 iterations. However, the standard deviation decreases to a very small positive value near the 220th iteration and then oscillates continuously.

4.3. Benchmark Testing: Griewank Function. In this subsection, the Griewank function is tested as follows:

$$f_3(\widehat{x}) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad -600 < x_i < 600, \tag{20}$$

where $n = 20$ and $\widehat{x} = (x_1, x_2, \ldots, x_{20})$. If $n = 1$, then the function $f_3(x)$ has 191 minima, with the global minimum at $x = 0$ and local minima at $\pm x$ for $x \cong 6.28005$, 12.5601, 18.8401, and 25.1202,.... First, the convergence tolerance is set to 0.01. Table 5 shows the shortest, mean and the longest CPU times achieved using the five methods in 10 simulations. As illustrated in Table 5, CPSO, DE, and GA fail to converge. The optimality of the proposed method and PSO, as given by the shortest, mean and the longest CPU times, is similar. Figure 8 shows the iteration excursions of the methods with the shortest CPU times.

Table 6 shows the best, mean and the worst $f_3(\widehat{x})$ obtained at the 1000th iteration by all methods. The proposed method still attains the best optimality compared with other methods.

Table 4: Performance of different methods used to minimize f_2 based on 10 simulation results (number of iterations = 1000).

		Proposed method	PSO	CPSO	DE	GA
Best f_2	f_2	0.0466	0.0258	65.8326	$4.19e-04$	1.9769
	Time (s)	0.1152	0.0488	0.0545	0.6335	2.0670
Mean	f_2	0.0633	0.0717	70.1498	0.0037	6.1595
	Time (s)	0.1162	0.0493	0.0535	0.6406	2.1172
Worst f_2	f_2	0.0816	0.1474	77.7252	0.0072	9.0414
	Time (s)	0.1157	0.0500	0.0543	0.6358	2.1768

Table 5: Performance of different methods used to minimize f_3 based on 10 simulation results (convergence tolerance = 0.01).

		Proposed method	PSO	CPSO	DE	GA
Min (Time)	f_3	0.0082	0.0098	—	—	—
	Generation	850	1416	—	—	—
	Time (s)	0.1349	0.1158	—	—	—
Mean (Time)	f_3	0.0093	0.0095	—	—	—
	Generation	5435	10347	—	—	—
	Time (s)	0.8773	0.8910	—	—	—
Max (Time)	f_3	0.0098	0.0100	—	—	—
	Generation	11553	30425	—	—	—
	Time (s)	1.6046	2.4117	—	—	—

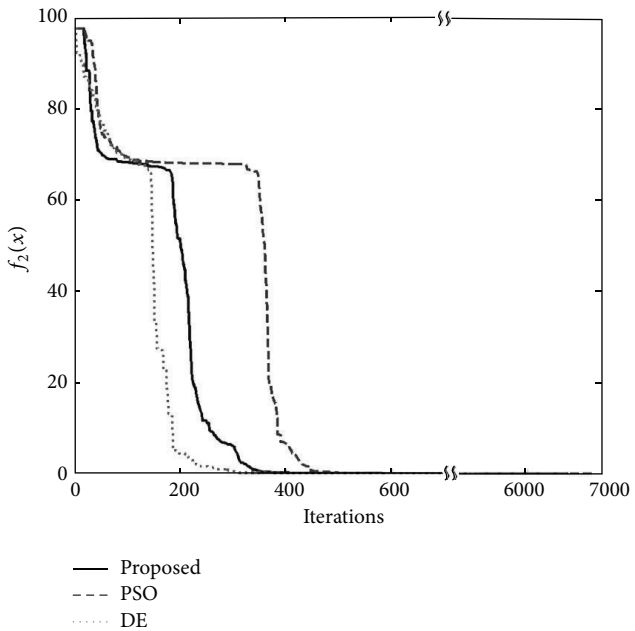

Figure 5: Iteration excursions with the shortest CPU times for different methods used to minimize f_2.

4.4. Benchmark Testing: Ackley Function. The Ackley function is an n-dimensional highly multimodal function that has a great number of local minima, which look like noise but only one global minimum at $f_4(\widehat{x}) = 0$ and $\widehat{x} = (x_1, x_2, \ldots) = (0, 0, \ldots)$:

$$f_4(\widehat{x}) = -20e^{-0.2\times\sqrt{(1/n)\sum_{i=1}^{n} x_i^2}} - e^{(1/n)\sum_{i=1}^{n}\cos(2\pi\times x_i)} + 20 + e^1 \quad -32 < x_i < 32, \tag{21}$$

where $n = 20$ and $\widehat{x} = (x_1, x_2, \ldots, x_{20})$ herein.

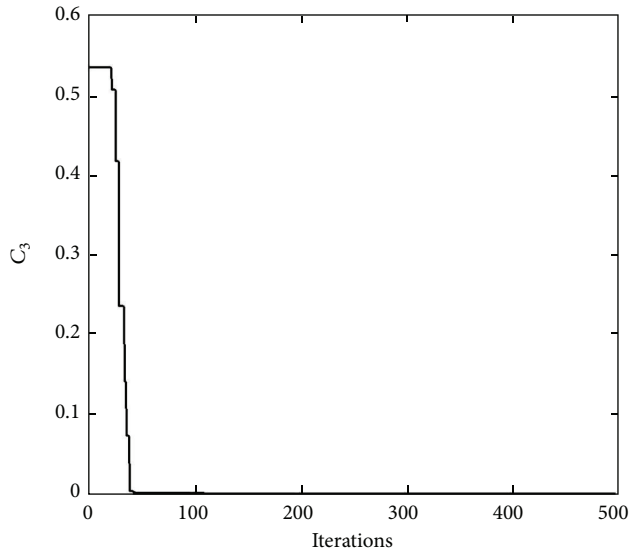

Figure 6: Iterative performance in terms of C_3 (Rosenbrock function).

First, the convergence tolerance is set to 0.001. Table 7 shows the shortest, mean and the longest CPU times of 10 simulations obtained by the five methods. As illustrated in Table 7, all methods are able to find the global optimum, except for CPSO (which fails to converge). The CPU times required by the proposed method are between those required by PSO and DE, while GA needs very long CPU times. All solutions (optimality) are similar from the viewpoints of the shortest, mean and the longest CPU times. Figure 11 shows the iteration excursions of the various methods with the shortest CPU times. The proposed method converges the fastest.

Table 8 shows the best, mean and the worst $f_4(\widehat{x})$ obtained at the 500th iteration by all methods. The proposed method and DE can gain the global optimum but PSO, CPSO,

Table 6: Performance of different methods used to minimize f_3 based on 10 simulation results (number of iterations = 1000).

		Proposed method	PSO	CPSO	DE	GA
Best f_3	f_3	0.0004	0.0195	$3.53e-02$	0.6151	0.3880
	Time (s)	0.1941	0.0820	0.1221	0.8560	2.7651
Mean	f_3	0.0129	0.0779	0.1293	0.6953	0.7267
	Time (s)	0.1927	0.1031	0.1157	0.8649	2.7097
Worst f_3	f_3	0.0472	0.1687	$1.87e-01$	0.8114	0.9426
	Time (s)	0.1922	0.1033	0.1144	0.8706	2.5987

Table 7: Performance of different methods used to minimize f_4 based on 10 simulation results (convergence tolerance = 0.001).

		Proposed method	PSO	CPSO	DE	GA
Shortest time	f_4	0.000926	0.000865	—	0.000963	0.000732
	Iterations	329	410	—	467	5604
	Time (s)	0.0698	0.0354	—	0.4140	313.2481
Mean	f_4	0.000956	0.000946	—	0.000944	0.000915
	Iterations	380.9	424.5	—	490.2	6222.1
	Time (s)	0.0775	0.0368	—	0.4371	322.7549
Longest time	f_4	0.000928	0.000959	—	0.000933	0.000972
	Iterations	398	431	—	503	7987
	Time (s)	0.0875	0.0397	—	0.4616	333.9159

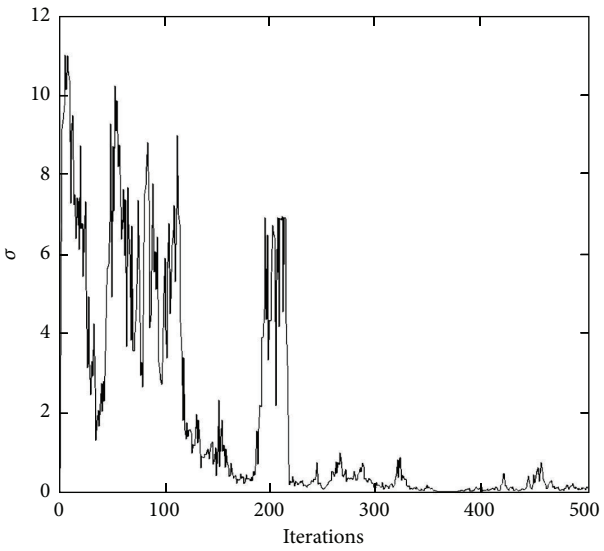

Figure 7: Iterative performance in terms of standard deviation (Rosenbrock function).

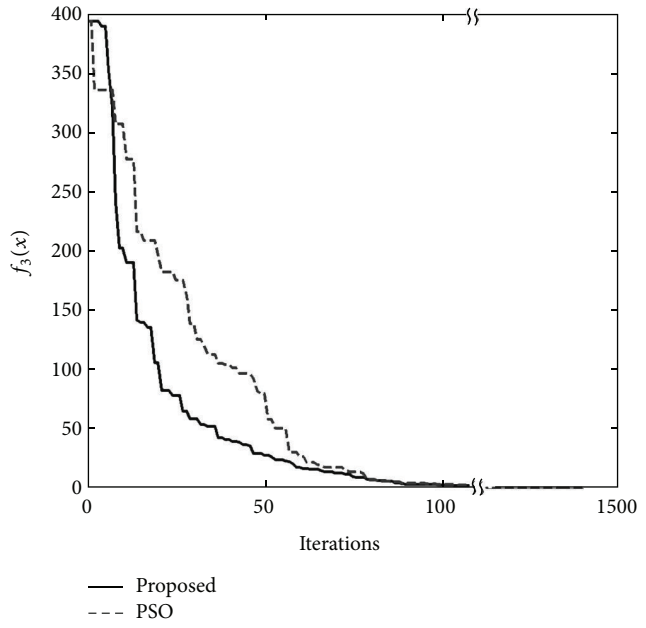

Figure 8: Iteration excursions with the shortest CPU times for different methods used to minimize f_3.

and GA cannot. The proposed method always gains the smallest values of $f_4(\hat{x})$ by inspecting the best, mean and the worst values of the $f_4(\hat{x})$. The proposed method also requires the shortest CPU times to converge to solutions. Figures 12 and 13 illustrate parameter C_3 and standard deviation σ, respectively, obtained using the proposed method in the case with the best $f_4(\hat{x})$. The values of C_3 decrease to small positive values close to the 220th iteration. The standard deviation decreases to very small positive values near the 400th iteration and thereafter oscillates (see Figures 9 and 10).

4.5. Studies on Loss Minimization in a Power System. Active power loss in the electric power system is caused by the resistance in the transmission/distribution lines. The losses can be evaluated as $\sum$(each line current)^2 × (each line resistance) or simplified as (total active power generations − total active power demands). If the active power demands are constant and voltage magnitudes are increased, then the line currents will be reduced, leading to reduction of active power losses.

Table 8: Performance of different methods used to minimize f_4 based on 10 simulation results (number of iterations = 500).

		Proposed method	PSO	CPSO	DE	GA
Best f_4	f_4	$8.29e-06$	$1.93e-04$	1.423543	$4.46e-04$	0.780793
	Time (s)	0.0917	0.0394	0.0472	0.4336	1.8306
Mean	f_4	$7.83e-05$	0.000463	2.379854	0.000836	1.517645
	Time (s)	0.0921	0.0405	0.0475	0.4346	1.7197
Worst f_4	f_4	$3.35e-04$	$2.58e-03$	$4.10e+00$	$8.12e-04$	2.621082
	Time (s)	0.0919	0.0406	0.0475	0.4347	1.6464

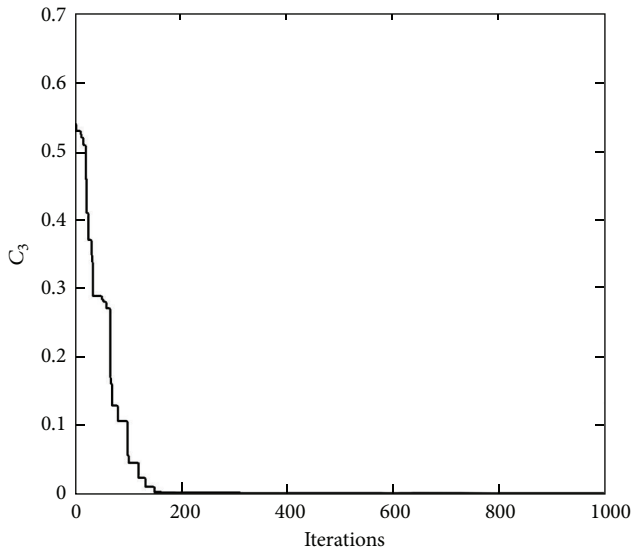

Figure 9: Iterative performance in terms of C_3 (Griewank function).

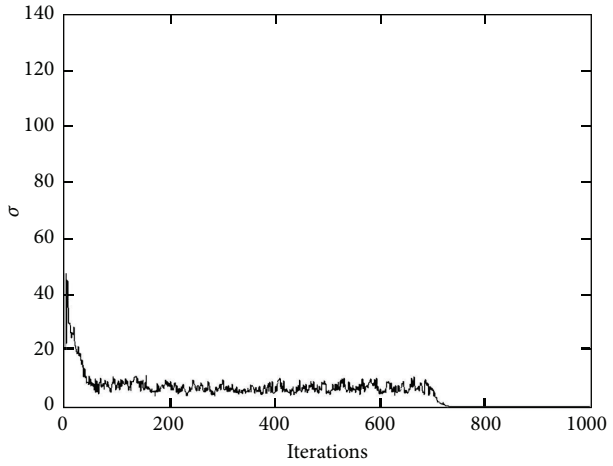

Figure 10: Iterative performance in terms of standard deviation (Griewank function).

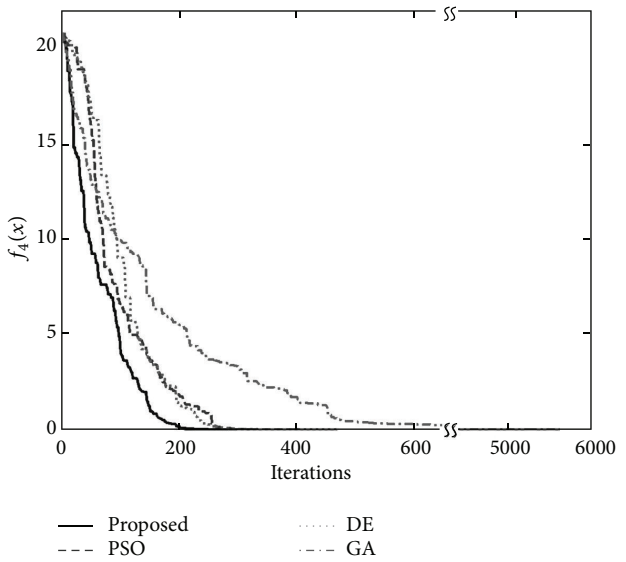

Figure 11: Iteration excursions with the shortest CPU times for different methods used to minimize f_4.

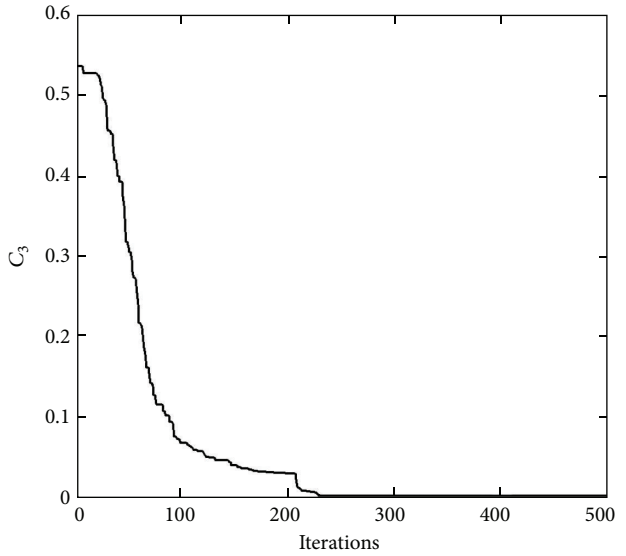

Figure 12: Iterative performance in terms of C_3 (Ackley function).

Therefore, engineers who work on electric power systems tend to utilize voltage controllers to increase the voltage profile in order to reduce the active power losses [22]. The voltage controllers include generator voltages, transformer taps, and shunt capacitors. The problem formulation of active power loss minimization is given in the appendix.

Figure 14 illustrates a 25-busbar power system with 3 wind farms at busbars 13, 14, and 25. The wind power generations in these three busbars are 2.4, 2.4, and 5.1 MW, respectively. The diesel generators are located at busbars 1~11. The shunt capacitors are at busbars 21, 22, and 23. Seven

Table 9: Optimality and CPU time required by different methods (10 runs).

		AEPSO	PSO	CPSO	DE	GA
Best objective	MW loss	1.783	1.865	1.829	1.857	1.862
	Time (s)	33.9461	43.4231	48.3487	42.0286	70.3614
Mean	MW loss	1.840	1.875	1.8654	1.915	1.886
	Time (s)	34.2277	46.8953	48.3953	44.9488	72.0624
Worst objective	MW loss	1.886	1.952	1.902	1.961	1.946
	Time (s)	34.6663	48.4029	48.7938	47.7389	75.1891

Table 10: Optimal solutions obtained by different methods (best solution).

Control variables	Proposed	PSO	CPSO	ED	GA
V_1–V_3	1.01	1.02	0.96	0.95	1.04
V_4–V_{11}	1.04	1.04	1.05	1.04	1.04
V_{13}	1.03	1.04	1.03	1.03	1.00
V_{14}	0.99	0.99	0.99	0.99	1.01
V_{25}	1.04	1.04	1.04	1.04	1.00
$\text{Tap}_{1–12}$	0.95	0.96	0.95	0.95	0.97
$\text{Tap}_{4–12}$	1.05	1.05	1.03	1.05	0.99
$\text{Tap}_{12–17}$	0.95	1.01	0.98	1.01	1.04
$\text{Tap}_{12–18}$	0.98	0.98	0.98	0.98	0.99
$\text{Tap}_{12–19}$	1.02	1.03	1.02	1.03	1.02
$\text{Tap}_{17–15}$	1.02	1.03	1.02	1.02	0.96
$\text{Tap}_{14–16}$	1.02	1.02	1.02	1.02	0.98
$\text{Tap}_{20\text{-}21}$	1.01	0.96	1.01	1.01	1.01
$\text{Tap}_{20–22}$	1.04	1.04	0.96	1.04	1.01
$\text{Tap}_{20–23}$	1.02	1.04	0.98	1.01	1.03
SC_{21}	0.05	0.05	0.1	0.1	0.09
SC_{22}	0.1	0.1	0.1	0.1	0.06
SC_{23}	0.15	0.15	0.1	0.1	0.09

demands can be seen at busbars 17~19 and 21~24. Consequently, the number of control variables is 39 (14 generator voltage magnitudes, 22 taps, and 3 shunt capaciotrs). The number of state variables is 36 (voltage magnitudes and phase angles at busbars 12, 15~19, and 20~24, phase angles and reactive power generations at generator busbars, and active power generation at reference busbar 1).

The proposed method, PSO, CPSO, DE, and GA are applied to the find the optimal 39 control variables in order to minimize the active power (MW) losses. Ten simulations, each with fixed 500 iterations are conducted using each method. Figure 15 illustrates the iteration performance of these five methods. It can be found the iteration process of the proposed method converges slowly at first but quickly after the 220th iteration. Table 9 reveals that the proposed method is able to converge the fastest and requires the shortest CPU times for the conditions of the best active power loss, mean values, and the worst active power loss in the 10 simulations. As shown in Table 9, GA still needs much more CPU time to solve the realistic problem and DE yields the worst, mean values over 10 simulation runs. Table 10 gives the results of optimal controls obtained using the different methods (best solution). Figure 16 displays the optimal voltage profile, which is obtained by the proposed method, of the electric power system.

4.6. Comparison of Time, Space Complexities, and Performance. Compared with the traditional PSO, the proposed

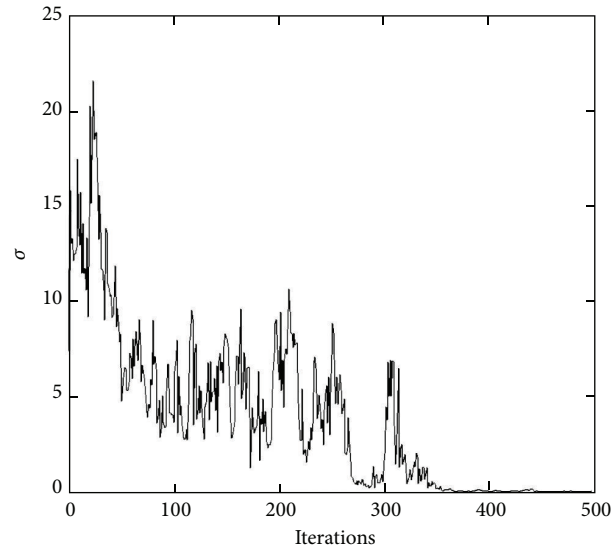

Figure 13: Iterative performance in terms of standard deviation (Ackley function).

novel mean search and particle pruning/cloning manners have been added to achieve stable iterative process and they avoid ineffective searches, respectively. Although the space complexity is increased a little bit due to these two tasks as shown in (3)–(6), the number of iterations can be mitigated

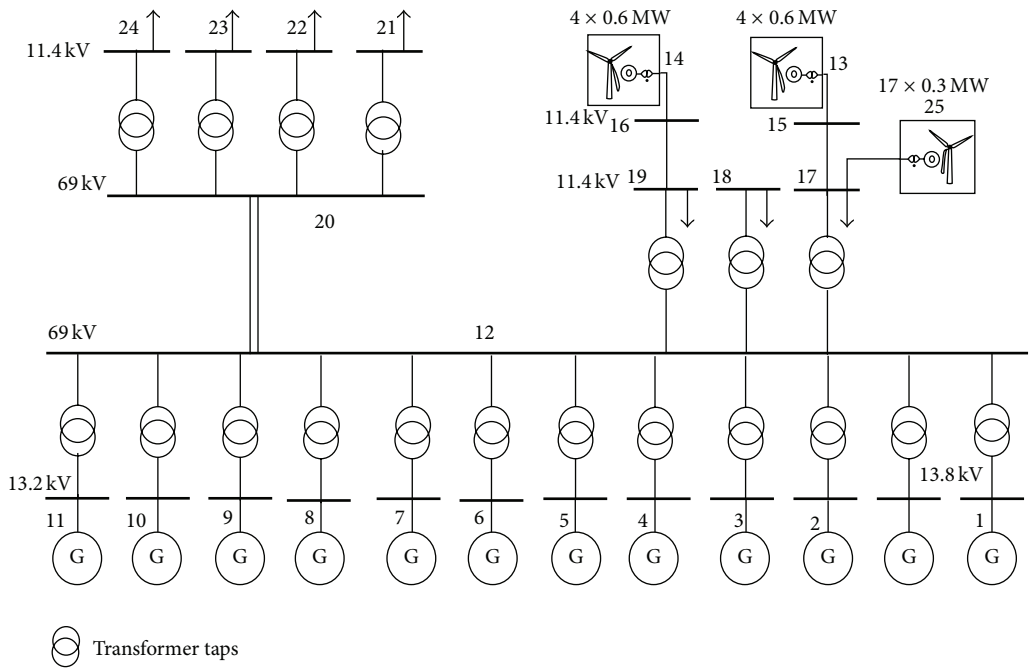

Figure 14: Oneline diagram of 25-busbar system.

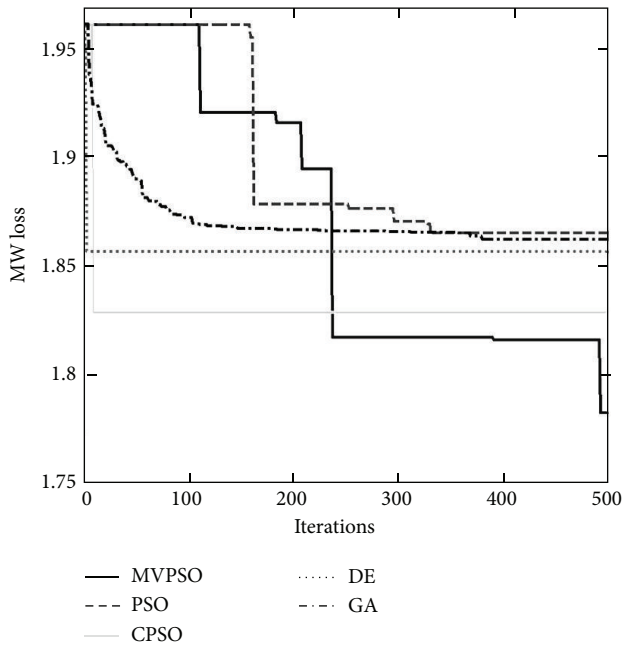

Figure 15: Iteration performance of five methods.

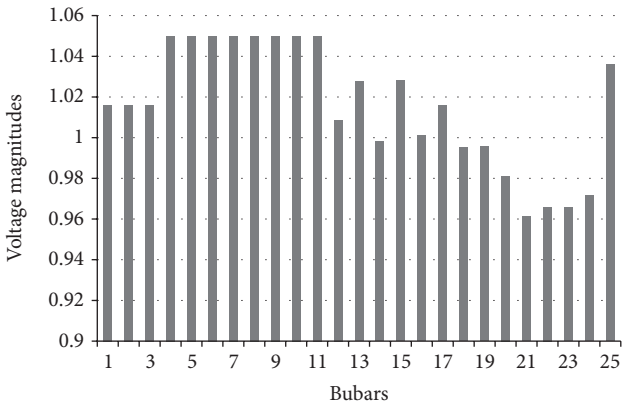

Figure 16: Voltage magnitudes in 25-busbar system (the proposed method).

and the computational burden can be reduced effectively. Thus, the time complexity is similar to that of the traditional PSO.

The performance comparisons among the proposed method, PSO, CPSO, DE, and GA are implemented using a set of four benchmark functions and one realistic loss minimization problem in an electric power system. The simulation results, as shown in Tables 1–10, imply that the CPSO and DE sometimes are unable to find the optimal solution or even diverge. Furthermore, both GA and DE always require long CPU times for computation. Additionally, the computational effort required by the proposed method to obtain high quality solutions is less than the effort required to obtain the same high quality solutions by the GA and DE. Therefore, the proposed method has the most stable convergence performance to find the optimums and has the best computational efficiency compared with the other methods considering the time complexity and space complexity.

5. Conclusions

In this paper, to provide a better tradeoff between global and local search in the iterative process than existing methods,

a novel elite-based PSO approach is proposed for studying highly dimensional and extraordinarily nonlinear optimization problems. Two tasks have been developed: mean search and particle pruning/cloning. The mean of all particles and the standard deviation of the distances between pairs of particles are evaluated in each iteration. The mean search ensures that the iterative process is stable. Particle pruning/cloning avoids ineffective searches. All parameters are adaptively self-tuned during the iterations. The impacts of these two tasks are gradually mitigated in the iteration process. When the proposed process is near convergence, its computational burden is similar to that of the traditional PSO.

Based on the studies herein of benchmark functions, the CPSO and DE sometimes are unable to find the optimal solution or even diverge. GA is often unable to find the optimum if the benchmark functions are very nonlinear. In terms of optimality, the proposed method is generally better than PSO. In the loss minimization problem, the best, mean and the worst objective values obtained by the proposed method in 10 simulations are always the best among those obtained by all methods. The smallest mean loss value implies that the numerical process of proposed method is the more stable than all of the others. The proposed method also requires the least CPU time to solve the loss minimization problem.

Appendix

A.

The active power loss minimization problem can be formulated as follows. Let the number of transmission lines, the set of generator busbars, and the set of total busbars be T, K, and I, respectively. The objective is to minimize the line losses in all lines:

$$\min \sum_{t=1}^{T} P_{\text{loss}}^{t}. \tag{A.1}$$

Let busbar 1 be the reference with a phase-angle of zero. The loss can be expressed as

$$V_1 \sum_{j=1}^{I} V_j \left(G_{1j} \cos\theta_{1j} + B_{1j} \sin\theta_{1j}\right) + \left(\sum_{k\in K, k\neq 1} PG_k\right) - \left(\sum_{i\in I} PD_i\right), \tag{A.2}$$

where P_{Gk} and P_{Di} are the active generation and demand at the kth generator and busbar i, respectively. The terms V_i and V_j represent the voltages of busbars i and j, respectively. $G_{ij} + jB_{ij}$ is the element at location (i, j) of the system admittance matrix. The term θ_{ij} represents the voltage phase-angle difference between busbars i and j.

The objective is subject to the following equality and inequality constraints.

A.1. Equality Constraints. The equality constraints covering the power flow equations are shown in (A.3) as follows:

$$\begin{aligned} P_{Gi} - P_{Di} - P_i &= 0, \quad i = 1, \ldots, I \\ Q_{Gi} - Q_{Ci} - Q_{Di} - Q_i &= 0, \quad i = 1, \ldots, I, \end{aligned} \tag{A.3}$$

where P_{Gi} and Q_{Gi} are the active and reactive power generation at busbar i, respectively. P_{Di} and Q_{Di} are the active and reactive power demands at busbar i, respectively. P_i and Q_i are the active and reactive power flow injections at busbar i, respectively. Q_{Ci} is the injected reactive power at busbar i where a capacitor is installed. P_i and Q_i can be represented as follows:

$$\begin{aligned} P_i &= V_i \sum_{j=1}^{I} V_j \left(G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}\right) \\ Q_i &= V_i \sum_{j=1}^{I} V_j \left(G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij}\right). \end{aligned} \tag{A.4}$$

A.2. Inequality Constraints. In the following, the subscripts "max" and "min" denote the upper and lower limits.

Operational Limits of Voltage Magnitudes at Generator Busbars. Consider

$$V_{Gk,\min} \leq V_{Gk}^{t} \leq V_{Gk,\max}, \quad k \in K, \tag{A.5}$$

where V_{Gk} represents the voltage magnitude at generator busbar k. The symbol K denotes the set of generator busbars.

Limitations on Reactive Power at Generators. Consider

$$Q_{Gk,\min} \leq Q_{Gk}^{t} \leq Q_{Gk,\max}, \quad k \in K, \tag{A.6}$$

where Q_{Gk} represents the reactive power output at generator k.

Operational Limits at Transformers. Consider

$$T_{l,\min} \leq T_{l}^{t} \leq T_{l,\max}, \quad l \in L, \tag{A.7}$$

where T_l represents the tap at transformer l. The symbol L represents the set of transformers.

Operational Limits of Capacitors. Consider

$$Q_{Cm,\min} \leq Q_{Cm} \leq Q_{Cm,\max}, \quad m \in M, \tag{A.8}$$

where Q_{Cm} represents the injected reactive power at busbar m. The symbol M represents the set of capacitors.

Operational Limits of Voltage Magnitudes at Demand Busbars. Consider

$$V_{Dn,\min} \leq V_{Dn} \leq V_{Dn,\max}, \quad n \in N, \tag{A.9}$$

where V_{Dn} represents the voltage magnitude at busbar n. The symbol N indicates the set of demands.

The active/reactive power demands, active power generation, and line impedances ($R_{ij} + jX_{ij}$) of the 25-busbar system are given in Tables 11(a), 11(b), and 11(c), respectively. Please note that $G_{ij} + jB_{ij} = (R_{ij} + jX_{ij})^{-1}$.

Table 11: (a) Active and reactive demands. (b) Power generation (MW). (c) Line impedances of 25-busbar system.

(a)

Busbar	Active demand (MW)	Reactive demand (MVAR)
17	13.58	2.78
18	15.85	3.31
19	13.76	2.79
21	23.24	4.69
22	18.63	3.83
23	18.81	3.84
24	11.94	2.43

(b)

Busbar	Active power
2	10
3	10
4	11
5	11
6	11
7	11
8	11
9	11
10	11
11	11
13	1.55
14	1.59
25	1.52

(c)

Line impedance	R_{ij}	X_{ij}
L_{1-12}	0.0470	0.980
L_{2-12}	0.0470	0.980
L_{3-12}	0.0470	0.9840
L_{4-12}	0.0585	0.9820
L_{5-12}	0.0585	0.9820
L_{6-12}	0.0585	0.9800
L_{7-12}	0.0585	0.9750
L_{8-12}	0.0585	0.9820
L_{9-12}	0.0585	0.9870
L_{10-12}	0.0585	0.9900
L_{11-12}	0.0585	0.9820
L_{12-17}	$8e-05$	0.3548
L_{12-18}	$8e-05$	0.3544
L_{12-19}	$8e-05$	0.3544
L_{12-20}	0.0239	0.0514
L_{13-15}	0	0.4000
L_{14-16}	0	0.4000
L_{15-17}	0.9540	1.4070
L_{16-19}	0.9540	1.4070
L_{20-21}	$8e-05$	0.3440
L_{20-22}	$8e-05$	0.3444
L_{20-23}	$8e-05$	0.3444
L_{20-24}	$8e-05$	0.3444
L_{25-17}	0	0.4000

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank Professor C. R. Chen at NTUT (Taiwan) for his comments about this work. The authors are also grateful for financial support from the National Science Council (Taiwan) under Grant NSC 102-3113-P-008-001.

References

[1] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science (MHS '95)*, pp. 39–43, Nagoya, Japan, October 1995.

[2] C. Wang, Y. C. Liu, and Y. T. Zhao, "Application of dynamic neighborhood small population particles warm optimization for reconfiguration of shipboard power system," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 4, pp. 1255–1262, 2013.

[3] M. Neyestani, M. M. Farsangi, and H. Nezamabadi-Pour, "A modified particle swarm optimization for economic dispatch with non-smooth cost functions," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 7, pp. 1121–1126, 2010.

[4] B. Vasumathi and S. Moorthi, "Implementation of hybrid ANNPSO algorithm on FPGA for harmonic estimation," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 3, pp. 476–483, 2012.

[5] W. X. Liu, I.-Y. Chung, L. Liu, S. Y. Leng, and D. A. Cartes, "Real-time particle swarm optimization based current harmonic cancellation," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 132–141, 2011.

[6] R. J. Wai, Y. C. Huang, Y. C. Chen, and Y. W. Lin, "Performance comparisons of intelligent load forecasting structures and its application to energy-saving load regulation," *Soft Computing*, vol. 17, no. 10, pp. 1797–1815, 2013.

[7] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[8] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Applied Soft Computing*, vol. 11, no. 4, pp. 3658–3670, 2011.

[9] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics B: Cybernetics*, vol. 39, no. 6, pp. 1362–1381, 2009.

[10] R. Caponetto, L. Fortuna, S. Fazzino, and M. G. Xibilia, "Chaotic sequences to improve the performance of evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, pp. 289–304, 2003.

[11] G. N. José and A. Enrique, "Restart particle swarm optimization with velocity modulation: a scalability test," *Soft Computing*, vol. 15, no. 11, pp. 2221–2232, 2011.

[12] W. Deng, R. Chen, B. He, Y. Liu, L. Yin, and J. Guo, "A novel two-stage hybrid swarm intelligence optimization algorithm and application," *Soft Computing*, vol. 16, no. 10, pp. 1707–1722, 2012.

[13] Y. Y. Li, R. R. Xiang, L. C. Jiao, and R. C. Liu, "An improved cooperative quantum-behaved particle swarm optimization," *Soft Computing*, vol. 16, no. 6, pp. 1061–1069, 2012.

[14] A. M. Arasomwan and A. O. Adewumi, "An investigation into the performance of particle swarm optimization with various chaotic maps," *Mathematical Problems in Engineering*, vol. 2014, Article ID 178959, 17 pages, 2014.

[15] H. Yin, C. Zhang, B. Zhang, Y. Guo, and T. Liu, "A hybrid multiobjective discrete particle swarm optimization algorithm

for a SLA-aware service composition problem," *Mathematical Problems in Engineering*, vol. 2014, Article ID 252934, 14 pages, 2014.

[16] B. Liu, L. Wang, Y.-H. Jin, F. Tang, and D.-X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons & Fractals*, vol. 25, no. 5, pp. 1261–1271, 2005.

[17] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[18] J. Kennedy and R. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, San Francisco, Calif, USA, 2001.

[19] S. Bouallègue, J. Haggège, M. Ayadi, and M. Benrejeb, "PID-type fuzzy logic controller tuning based on particle swarm optimization," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 3, pp. 484–493, 2012.

[20] N. J. Li, W. J. Wang, C. C. J. Hsu, W. Chang, H. G. Chou, and J. W. Changa, "Enhanced particle swarm optimizer incorporating a weighted particle," *Neurocomputing*, vol. 124, pp. 218–227, 2014.

[21] G. A. Ortiz, "Evolution Strategies (ES)," Mathwork, 2012.

[22] A. J. Wood and B. F. Wollenberg, *Power Generation, Operation and Control*, John Wiley & Son, New York, NY, USA, 2nd edition, 1996.