Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 293087, 10 pages
http://dx.doi.org/10.1155/2014/293087

Research Article

A Comparison of Selected Modifications of the Particle Swarm Optimization Algorithm

Michala Jakubcová, Petr Máca, and Pavel Pech

Department of Water Resources and Environmental Modeling, Faculty of Environmental Sciences, Czech University of Life Sciences Prague, Kamýcká 1176, Prague 6, 165 21 Suchdol, Czech Republic

Correspondence should be addressed to Michala Jakubcová; jakubcovam@fzp.czu.cz

Received 10 October 2013; Accepted 24 March 2014; Published 15 June 2014

Academic Editor: Nan-Jing Huang

We compare 27 modifications of the original particle swarm optimization (PSO) algorithm. The analysis evaluated nine basic PSO types, which differ according to the swarm evolution as controlled by various inertia weights and constriction factor. Each of the basic PSO modifications was analyzed using three different distributed strategies. In the first strategy, the entire swarm population is considered as one unit (OC-PSO), the second strategy periodically partitions the population into equally large complexes according to the particle's functional value (SCE-PSO), and the final strategy periodically splits the swarm population into complexes using random permutation (SCERand-PSO). All variants are tested using 11 benchmark functions that were prepared for the special session on real-parameter optimization of CEC 2005. It was found that the best modification of the PSO algorithm is a variant with adaptive inertia weight. The best distribution strategy is SCE-PSO, which gives better results than do OC-PSO and SCERand-PSO for seven functions. The sphere function showed no significant difference between SCE-PSO and SCERand-PSO. It follows that a shuffling mechanism improves the optimization process.

1. Introduction

Particle swarm optimization (PSO) is a stochastic, metaheuristic computational technique for searching the optimal regions from multidimensional space. It is an optimization method inspired by social behaviour of organisms and was established by Kennedy and Eberhart in 1995 [1]. The technique is based on iterative work with a population. PSO is an evolutionary computation (EC) method within the group of techniques known as swarm intelligence (SI) [2, 3]. PSO mimics the movement of flock of birds or school of fish using simple rules for adjusting the particle location, which is adjusted by means of its velocity information.

PSO's main benefits are that there are few parameters to adjust and the method is easy to implement. Another advantage of PSO over derivative based local search methods is that there is no need for the gradient information during the iterative search when solving complicated optimization problems [4–6].

While it has been successfully applied to solve many tests and real-life optimization problems [7–9], the PSO method often suffers from premature convergence and, as a result, from the optimization process's finding a merely local optimum. In order to achieve better algorithm performance, the original PSO algorithm has been modified by adding the parameter inertia weight or constriction factor [10–13].

Another important strategy for improving EC algorithms relies on division of the original population into subswarms or complexes which simultaneously search across the parametric space and exchange information according to some prescribed rule. Periodic shuffling is a typical example [9, 14, 15].

In order to explore the interaction of modifications in particle velocities together with the different types of distributed PSO versions, this paper analyzes 27 different PSO variants. Nine modifications of the PSO algorithm, in which the original particle velocities are altered using different approaches for setting the inertia weights and constriction

factor [13], are combined with three strategies for swarm distribution. The population is either considered as one unit (OC-PSO) or the swarm is divided into several complexes either according to the functional value (SCE-PSO) [14] or randomly (SCERand-PSO).

The remainder of this paper is organized as follows. Section 2 describes the particle swarm optimization method. The original equations and modifications of PSO algorithm are included. Section 3 describes different strategies for distribution of PSO. The experiment and obtained results are compared in Section 4. Conclusions are discussed in Section 5.

2. Particle Swarm Optimization

Particle swarm optimization is a global optimization method applied to find the optimal solution $\mathbf{X}^{\text{opt}}$ of objective function f. The sought optimum is most generally a minimum value. There exists a population of particles $i = 1, \ldots, S$, where S is the total number of individuals. All particles search through the problem space of dimension $d = 1, \ldots, \text{Dim}$, where Dim is the total number of dimensions. Each particle stores information about its position and velocity. The vector of the ith particle's position is $\mathbf{X}_i = (\mathbf{x}_1^i, \mathbf{x}_2^i, \ldots, \mathbf{x}_{\text{Dim}}^i)$ and the vector of the ith particle's velocity is $\mathbf{V}_i = (\mathbf{v}_1^i, \mathbf{v}_2^i, \ldots, \mathbf{v}_{\text{Dim}}^i)$. Each particle maintains a memory of its previous best position which is represented as $\mathbf{P}_i = (\mathbf{p}_1^i, \mathbf{p}_2^i, \ldots, \mathbf{p}_{\text{Dim}}^i)$. The best position among all particles from the swarm is represented as $\mathbf{G} = (\mathbf{g}_1, \mathbf{g}_1, \ldots, \mathbf{g}_{\text{Dim}})$. Equations (1) and (2) are the original PSO equations for computing a new velocity and new position. Consider

$$\mathbf{v}_d^i(t+1) = \mathbf{v}_d^i(t) + c_1 \cdot r_1 \cdot \left(\mathbf{p}_d^i(t) - \mathbf{x}_d^i(t)\right) + c_2 \cdot r_2 \cdot \left(\mathbf{g}_d(t) - \mathbf{x}_d^i(t)\right), \tag{1}$$

$$\mathbf{x}_d^i(t+1) = \mathbf{x}_d^i(t) + \mathbf{v}_d^i(t+1) \tag{2}$$

for all $i \in 1 \ldots S, d \in 1 \ldots \text{Dim}$, where t is a time step, c_1 and c_2 are acceleration constants predefined by the user, and r_1 and r_2 are random numbers uniformly distributed in $[0, 1]$. The component with $\mathbf{P}_i$ in (1) is known as the cognition part and it tells us about the individual experience of the particle. The component with $\mathbf{G}$ is called the social part and it represents the cooperation among particles within the swarm [16].

The simplified original PSO algorithm is shown in Algorithm 1. Initialization of a particle's position is randomly distributed in the range of $[x_{\min}, x_{\max}]$ as shown in

$$\mathbf{X} = x_{\min} + (x_{\max} - x_{\min}) \cdot \text{rand}(), \tag{3}$$

where rand() is a random number uniformly distributed in $[0, 1]$, while $x_{\min}$ and $x_{\max}$ are boundaries of the search space and their values depend on the benchmark function [17]. In this paper, initialization of particles is through Latin hypercube sampling (LHS).

A particle's initial velocity could be randomly distributed in the range of $[-v_{\max}, v_{\max}]$ or, alternatively, the velocities could be initialized to 0, since the starting positions are already randomized [18]. In initial experiments, the value of $v_{\max}$ was set to 100 000, and in subsequent experiments and applications it was found that a better approach is to limit $v_{\max}$ to $x_{\max}$ [19]. Other authors [20, 21] have set the value of maximum velocity as $v_{\max} = k \cdot x_{\max}$, where $0.1 < k < 1$. A larger value of $v_{\max}$ facilitates global exploration, whereas a smaller value of $v_{\max}$ encourages local exploitation [22]. In this paper, $v_{\max} = x_{\max}$ was applied for initial particle velocity.

```
(1)  initialize the position and velocity of all particles
(2)  repeat
(3)     for each particle i = 1 to S do
(4)        if (f(X_i) < f(P_i)) then
(5)           P_i = X_i
(6)        end if
(7)        G = min{P_0, P_1, ..., P_S}
(8)        for each dimension d = 1 to Dim do
(9)           v_d^i = v_d^i + c_1 · r_1 · (p_d^i − x_d^i) + c_2 · r_2 · (g_d − x_d^i)
(10)          x_d^i = x_d^i + v_d^i
(11)       end for
(12)    end for
(13) until termination criteria is met
```

Algorithm 1: Original PSO algorithm.

Table 1: Summary of PSO modifications.

Label	Equation
AdaptW	$w = (w_{\max} - w_{\min}) \cdot P_s + w_{\min}$
ChaoticRandW	$w(\text{iter}) = 0.5 \cdot \text{rand}() + 0.5 \cdot z$
ChaoticW	$w(\text{iter}) = (w_{\max} - w_{\min}) \cdot \frac{\text{iter}_{\max} - \text{iter}}{\text{iter}_{\max}} + w_{\min} \cdot z$
ConstantW	$w = c$
ConstrFactor	$K = \frac{2}{\left\lvert 2 - \varphi - \sqrt{\varphi^2 - 4 \cdot \varphi} \right\rvert}$
LinTimeVaryingW	$w(\text{iter}) = \frac{\text{iter}_{\max} - \text{iter}}{\text{iter}_{\max}} \cdot (w_{\max} - w_{\min}) + w_{\min}$
NonlinTimeConstW	$w(\text{iter}) = w_{\text{ini}} \cdot u^{\text{iter}}$
NonlinTimeW	$w(\text{iter}) = \left(\frac{2}{\text{iter}}\right)^{0.3}$
RandomW	$w = 0.5 + \frac{\text{rand}()}{2}$

2.1. Modifications of the PSO Algorithm. The original PSO equation was modified to improve the ability for optimization. The first group of modifications consists in incorporating the parameter of *inertia weight* w and the second in using the parameter of *constriction factor* K. In the present study, nine variants of PSO algorithm were used and tested (Table 1), including eight modifications using w and one modification with K.

The use of the inertia weight parameter (4) was developed by Shi and Eberhart [11] and it has provided for improved performance:

$$\mathbf{v}_d^i = w \cdot \mathbf{v}_d^i + c_1 \cdot r_1 \cdot \left(\mathbf{p}_d^i - \mathbf{x}_d^i\right) + c_2 \cdot r_2 \cdot \left(\mathbf{g}_d - \mathbf{x}_d^i\right). \quad (4)$$

There are many methods of computing the inertia weight value. Nickabadi et al. [13] divided those techniques into three classes which are applied in this paper: (1) constant and random inertia weight, (2) time varying inertia weight strategies, and (3) adaptive inertia weight. They compared all modifications employing benchmark functions and proposed a PSO algorithm using adaptive inertia weight.

The constant ("ConstantW") and random ("RandomW") inertia weights are used, where no input information is required. Bansal et al. [23] discussed their work with Shi and Eberhart [10] and set the constant inertia weight to be equal to 0.7. Gimmler et al. [24] proposed using constant inertia weight for hybrid particle swarm optimization. The best constant w was 0.2. We set the constant inertia weight for the "ConstantW" modification to 0.7 for this study, because our experiment is more similar to that of Bansal et al. [23]. Eberhart and Shi [25] had proposed random inertia weight, where w is a variable with uniform distribution within $[0.5, 1]$.

Time varying inertia weight is defined as a function of time or number of iterations and this method may be linear or nonlinear. In linear decreasing w ("LinTimeVaryingW") developed by Shi and Eberhart [11], inertia weight decreases linearly from $w_{\max} = 0.9$ to $w_{\min} = 0.4$. This method of determining the inertia weight value is very common [26, 27]. Eberhart and Shi [19] compared linearly decreasing w with constriction factor and found that better performance was achieved when constriction factor was used. The chaotic model ("ChaoticW") and chaotic random model ("ChaoticRandW") of inertia weight were proposed by Feng et al. [28], where $z = 4 \cdot z \cdot (1 - z)$ and the initial value of z is uniformly distributed in $[0, 1]$. Two modifications of nonlinear time varying inertia weight are used. In the "NonlinTimeW" and the "NonlinTimeConstW", where parameter u is set to 1.0002, w_{ini} is the initial value for inertia weight uniformly distributed in [0, 1] and iter is the actual number of functional evaluations [13].

One modification of adaptive inertia weight proposed by Nickabadi et al. [13] is used ("AdaptW"), because it had demonstrated the best performance in the original paper. The w value is adapted based on one feedback parameter. The value S shows the success of particles and is defined as in (5) and P_s is the success percentage of the swarm and it is computed as in (6), where n is the size of the population. The range of inertia weights $[w_{\min}, w_{\max}]$ is $[0, 1]$. Consider

$$S(i,t) = \begin{cases} 1 & \text{if fit}\left(\mathbf{p}_d^i(t)\right) < \text{fit}\left(\mathbf{p}_d^i(t-1)\right), \\ 0 & \text{if fit}\left(\mathbf{p}_d^i(t)\right) = \text{fit}\left(\mathbf{p}_d^i(t-1)\right), \end{cases} \quad (5)$$

$$P_s(t) = \frac{\sum_{i=1}^{n} S(i,t)}{n}. \quad (6)$$

Beyond variants using inertia weight, the next modification of the original PSO algorithm consists in incorporating the parameter of constriction factor ((7) and (8)). This strategy was first used by Clerc [29] and it increases convergence of the algorithm. We have named the method "ConstrFactor." Another approach to constriction factor is that of Bui et al. [12]. They proposed a time-dependent strategy, where they used nonlinear decay rules to adapt K. Their results are not better than those obtained when using the setting in accordance with Clerc [29], and we therefore calculate the K value using

$$\mathbf{v}_d^i = K \cdot \left(\mathbf{v}_d^i + c_1 \cdot r_1 \cdot \left(\mathbf{p}_d^i - \mathbf{x}_d^i\right) + c_2 \cdot r_2 \cdot \left(\mathbf{g}_d - \mathbf{x}_d^i\right)\right), \quad (7)$$

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4 \cdot \varphi}\right|}, \quad (8)$$

where $\varphi = c_1 + c_2$ and $\varphi > 4$.

3. Distribution of PSO

All nine modifications are used with three strategies of swarm distribution (SD). We observed changes in behaviour of the population for each modification and strategy. The first distributed strategy considered the whole population as one unit and we called it OC-PSO. In the next SD, the population was divided into several complexes according to the particle's functional value (SCE-PSO) or through random permutation (SCERand-PSO).

3.1. One Complex Strategy (OC-PSO). In the OC-PSO method, the entire population is considered as a single unit. All particles participate in the PSO algorithm and share information about the best position achieved so far.

3.2. Shuffled Complex Evolution (SCE-PSO). During the optimization process a premature convergence to a local optimum could appear instead of the global optimum. Many researchers in this field are devoted avoiding this premature convergence [30–32]. To address this, Duan et al. [14] proposed shuffled complex evolution (SCE). Yan et al. [33] combined SCE with the particle swarm optimization algorithm (SCE-PSO).

The SCE-PSO method is described simply below and is shown in Algorithm 2. After the first initialization, the entire population is divided into subswarms according to the functional values of the individuals. All particles are sorted in increasing order and then each ith complex receives the parent individuals $\mathbf{X}_i, \mathbf{X}_{i+NC}, \mathbf{X}_{i+2NC}, \ldots$, where NC is the number of complexes [14, 34]. The PSO algorithm is applied at each complex. After running a predefined number of iterations in all complexes, all particles return to the swarm and the shuffling and redistribution of complexes according to the functional value are again made. This is repeated until the termination criteria are satisfied.

```
Require S, NC, N_comp, max_eval, fitness_lim
(1)  initialize population X
(2)  while (number_eval ≤ max_eval) || (fitness_best ≥ fitness_lim) do
(3)    E ← sorted X in increasing order according to the functional value
(4)    for i = 1 to NC do
(5)      A_i ← divided E into NC complexes
(6)      run PSO
(7)      for j = 1 to N_comp do
(8)        if (f(X_j) < f(P_j)) then
(9)          P_j = X_j
(10)       end if
(11)       if (f(X_j) < f(G)) then
(12)         G = X_j
(13)       end if
(14)     end for
(15)   end for
(16) end while
```

Algorithm 2: Algorithm SCE-PSO.

The shuffling mechanism preserves the population diversity and helps to prevent premature convergence. For this study, the shuffling was performed after the running of a predefined number of generations in each complex [34]. Another approach allows the shuffling to occur randomly with some associated probability [35].

In the original SCE-PSO method [33], only a predefined number of particles from each complex are chosen to participate in the PSO algorithm. In this paper, the number of participating individuals is equal to the number of particles in the complex. This means that all particles from the complex are inputs to the PSO algorithm.

3.3. Random Shuffled Complex Evolution (SCERand-PSO). Random shuffled complex evolution differs from SCE-PSO in that the entire population is divided into complexes according to random permutation. There is no sorting by functional value. The algorithm for computing random permutation is according to Durstenfeld [36].

Algorithm 2 is applied for the SCERand-PSO, except that at line 3 the following substitution is made: **E** ← sorted **X** according to random permutation.

4. Experiment and Results

4.1. Experimental Setup. After running several tests with different parameter settings, the following setup was found to be the best.

The position of individuals is initialized randomly between lower and upper bounds of the search space through Latin hypercube sampling (LHS). The range of the problem space depends on the benchmark function (Table 2). LHS is a type of stratified Monte Carlo sampling first described by McKay et al. in 1979 for the analysis of output from a computer code [37]. The range, which is in PSO optimization defined by lower and upper bounds of the search space, is portioned into n intervals of equal probability $1/n$. The n value is in PSO which is equal to the population size. LHS then randomly selects one value from each interval [38]. Due to this selection, particles are uniformly distributed in the search space.

In accordance with Eberhart and Shi [19], the maximum value of velocity is set to $x_{\max}$. The value of acceleration constants c_1 and c_2 in variants with inertia weight is set to 2. In modifications with constriction factor, the K value is set to 0.729 and $c_1 = c_2 = 1.49445$ [19].

In accordance with Eberhart and Shi [39], the population size is set to 25. In the OC-PSO method, all particles are solved together. In the SCE-PSO and SCERand-PSO methods, individuals are uniformly divided into 6 complexes, where each complex contains 25 particles. The number of shuffling is set to 5. The maximum number of function evaluations is $10\,000 \cdot \mathrm{Dim}$ and the dimension of the solution is set to 30. For analyzing the results, the total number of optimization runs is set to 25. Each run stops when the maximum number of evaluations is achieved.

4.2. Benchmark Problems. For comparison purposes, 11 benchmark functions prepared for the special session on real-parameter optimization of CEC 2005 [40] were used. All functions have shifted global optima, and some of them are rotated or with noise. The benchmark functions are summarized in Table 2. The aim is to find the minimum of all functions.

The optimization problem is constrained except for function f_7. Particles move only in restricted space and cannot cross the boundaries. This means that each position of particle i is bounded by lower and upper limits [41]. In the PSO algorithm, it is reflected such that the particles must lie within the range $[x_{\min}, x_{\max}]$. If a new particle position is outside the boundaries, that particle retains the position of its parent. In function f_7, the global optimum is situated outside the range [40] and therefore the optimization problem is unconstrained and particles can cross the boundaries.

Table 2: Summary of benchmark functions.

	Function	Range	$f(\mathbf{X}^{\mathbf{opt}})$
Sphere	$f_1(x) = \sum_{i=1}^{d} z_i^2$	$[-100, 100]^d$	−450
Schwefel 1.2	$f_2(x) = \sum_{i=1}^{d} \left(\sum_{j=1}^{i} z_j \right)^2$	$[-100, 100]^d$	−450
Elliptic rotated	$f_3(x) = \sum_{i=1}^{d} \left(\left(10^6\right)^{(i-1)/(d-1)} z_i^2 \right)$	$[-100, 100]^d$	−450
Schwefel 1.2 noise	$f_4(x) = \left(\sum_{i=1}^{d} \left(\sum_{j=1}^{i} z_j \right)^2 \right) \cdot (1 + 0.4 \lvert N(0,1) \rvert)$	$[-100, 100]^d$	−450
Schwefel 2.6	$f_5(x) = \max \{ \lvert A_i x - B_i \rvert \}$	$[-100, 100]^d$	−310
Rosenbrock	$f_6(x) = \sum_{i=1}^{d-1} \left(100 \left(z_i^2 - z_{i+1} \right)^2 + \left(z_i - 1 \right)^2 \right)$	$[-100, 100]^d$	390
Griewank rotated	$f_7(x) = \sum_{i=1}^{d} \frac{z_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{z_i}{\sqrt{i}} \right) + 1$	$[0, 600]^d$	−180
Ackley rotated	$f_8(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{d} \sum_{i=1}^{d} z_i^2} \right) - \exp\left(\frac{1}{d} \sum_{i=1}^{d} \cos\left(2\pi z_i\right) \right) + 20 + e$	$[-32, 32]^d$	−140
Rastrigin	$f_9(x) = \sum_{i=1}^{d} \left(z_i^2 - 10 \cos\left(2\pi z_i\right) + 10 \right)$	$[-5, 5]^d$	−330
Rastrigin rotated	$f_{10}(x) = \sum_{i=1}^{d} \left(z_i^2 - 10 \cos\left(2\pi z_i\right) + 10 \right)$	$[-5, 5]^d$	−330
Weierstrass rotated	$f_{11}(x) = \sum_{i=1}^{d} \left[\sum_{k=0}^{20} \left(0.5^k \cos\left(2\pi 3^k \left(z_i + 0.5 \right) \right) \right) \right] - d \sum_{k=0}^{20} \left(0.5^k \cos\left(2\pi 3^k \cdot 0.5 \right) \right)$	$[-0.5, 0.5]^d$	90

Table 3: Statistical indices of the best solutions of 11 benchmark functions.

Function	Modification	SD[a]	min	25%	median	75%	max	mean	std
f_1	AdaptW	2	0.00E + 00	5.68E − 14	1.14E − 13	1.71E − 13	2.27E − 13	1.02E − 13	6.14E − 14
	AdaptW	3	0.00E + 00	5.68E − 14	1.14E − 13	1.14E − 13	5.59E − 05	4.74E − 06	1.53E − 05
f_2	AdaptW	2	5.68E − 14	2.27E − 13	3.98E − 13	6.82E − 13	1.53E − 12	5.16E − 13	4.33E − 13
f_3	AdaptW	2	4.17E + 05	7.08E + 05	1.02E + 06	1.57E + 06	2.30E + 06	1.15E + 06	5.78E + 05
f_4	NonlinTimeConstW	2	7.09E − 02	7.56E − 01	2.21E + 00	4.38E + 00	1.94E + 03	1.20E + 02	4.27E + 02
f_5	NonlinTimeConstW	1	1.31E + 03	2.78E + 03	3.21E + 03	4.57E + 03	6.14E + 03	3.66E + 03	1.26E + 03
f_6	AdaptW	2	1.71E − 03	3.83E + 00	4.06E + 00	7.47E + 00	1.80E + 01	5.76E + 00	4.87E + 00
f_7	NonlinTimeConstW	2	2.84E − 14	9.86E − 03	1.23E − 02	2.22E − 02	6.87E − 02	1.70E − 02	1.73E − 02
f_8	AdaptW	2	2.06E + 01	2.14E + 01	2.16E + 01	2.18E + 01	2.20E + 01	2.15E + 01	3.89E − 01
f_9	ChaoticW	3	2.89E + 01	6.78E + 01	8.51E + 01	1.12E + 02	6.07E + 02	1.21E + 02	1.27E + 02
f_{10}	ChaoticRandW	3	7.87E + 01	1.63E + 02	4.22E + 02	4.85E + 02	6.56E + 02	3.44E + 02	1.88E + 02
	LinTimeVaryingW	3	8.38E + 01	2.03E + 02	4.02E + 02	4.81E + 02	5.71E + 02	3.54E + 02	1.54E + 02
f_{11}	LinTimeVaryingW	2	2.66E + 01	3.12E + 01	3.25E + 01	3.52E + 01	4.78E + 01	3.38E + 01	4.97E + 00

[a]SD = swar distribution, where 1 is for OC-PSO, 2 is for SCE-PSO, and 3 is for SCERand-PSO.

There exist two versions of the PSO algorithm: *global* and *local*. In the global variant, the neighborhood consists of all particles of the swarm. In the local variant, each particle is assigned to a neighborhood consisting of a predefined number of particles [41, 42]. For the OC-PSO method, the global variant is used. For the SCE-PSO and SCERand-PSO, the local variant is used and particles share the information about their best positions only with other particles from a given complex.

4.3. Results and Discussion. The nonparametric Wilcoxon test was used for statistical comparison. Inputs to those calculations were the best fitness values achieved for all modifications. The null hypothesis H_0 of the Wilcoxon test

Table 4: Estimation of the best and poorest modifications.

f_1	Best	Poorest
OC-PSO	AdaptW, ChaoticRandW, ChaoticW, NonlinTimeConstW	ConstantW
SCE-PSO	AdaptW	ConstantW
SCERand-PSO	AdaptW, NonlinTimeConstW	ConstantW
f_2	Best	Poorest
OC-PSO	AdaptW, LinTimeVaryingW	ConstrFactor
SCE-PSO	AdaptW	ConstrFactor
SCERand-PSO	AdaptW	ConstrFactor
f_3	Best	Poorest
OC-PSO	LinTimeVaryingW	ConstantW
SCE-PSO	AdaptW, NonlinTimeW	ConstantW
SCERand-PSO	AdaptW, NonlinTimeW	ConstantW, RandomW
f_4	Best	Poorest
OC-PSO	ChaoticRandW, LinTimeVaryingW	ConstrFactor
SCE-PSO	NonlinTimeConstW	ConstrFactor
SCERand-PSO	NonlinTimeConstW	ConstrFactor
f_5	Best	Poorest
OC-PSO	ChaoticRandW, NonlinTimeConstW	ConstantW
SCE-PSO	ChaoticRandW, ChaoticW, NonlinTimeConstW	ConstantW
SCERand-PSO	ChaoticRandW, ChaoticW, LinTimeVaryingW, NonlinTimeConstW	ConstantW
f_6	Best	Poorest
OC-PSO	ChaoticRandW, NonlinTimeConstW	ConstantW
SCE-PSO	AdaptW	ConstantW
SCERand-PSO	AdaptW	ConstantW
f_7	Best	Poorest
OC-PSO	NonlinTimeConstW	ConstantW, RandomW
SCE-PSO	NonlinTimeConstW	ConstantW, ConstrFactor
SCERand-PSO	AdaptW, NonlinTimeW	ConstantW, ConstrFactor, LinTimeVarying
f_8	Best	Poorest
OC-PSO	AdaptW	ConstrFactor
SCE-PSO	AdaptW	LinTimeVarying
SCERand-PSO	AdaptW	ConstantW, ConstrFactor, LinTimeVaryingW, NonlinTimeConstW
f_9	Best	Poorest
OC-PSO	ChaoticW	ConstantW, ConstrFactor, RandomW
SCE-PSO	ChaoticW	ConstantW, ConstrFactor
SCERand-PSO	ChaoticW	ConstantW
f_{10}	Best	Poorest
OC-PSO	ChaoticRandW	ConstantW
SCE-PSO	ChaoticW	ConstantW
SCERand-PSO	ChaoticRandW, ChaoticW, LinTimeVaryingW	ConstantW
f_{11}	Best	Poorest
OC-PSO	AdaptW, LinTimevaryingW, NonlinTimeConstW, NonlinTimeW	ConstantW, ConstrFactor
SCE-PSO	LinTimevaryingW	ConstantW, ConstrFactor, RandomW
SCERand-PSO	AdaptW	ConstantW, ConstrFactor, RandomW

is that the differences between algorithms have a median of zero. The H_0 is rejected if the P value is less than 0.05 [43, 44]. Algorithms were written in C++ and computations of P value and graphs were made in the program R.

Table 3 reflects the best modification and distributed strategy for each function. In the table, the minimum, 25% quartile, median, 75% quartile, maximum, mean, and standard deviation are indicated. All 25-program runs of each modification and strategy were compared in each numbered evaluation. The values in Table 3 reflect the best fitness achieved and report other statistical indices belonging to the same numbered evaluation. As can be seen, strategy SCE-PSO produced the best solution in seven functions. Strategy SCERand-PSO produced the best solution in two functions (f_9, f_{10}) and in one function (f_5) the best solution was from strategy OC-PSO. For function f_1, there was no significant difference between strategy SCE-PSO and SCERand-PSO.

Upon closer examination and as seen in Table 4, "AdaptW" and "NonlinTimeConstW" are the best modifications for unimodal functions (f_1–f_5). The poorest variants are "ConstantW" and "ConstrFactor." The best PSO modification for multimodal functions (f_6–f_{11}) is "AdaptW" and the poorest is "ConstantW."

For rotated functions (f_3, f_7, f_8, f_{10}, f_{11}) the best modification of the PSO algorithm appears to be "AdaptW" and the poorest is "CostantW." For functions where there is no transformation matrix to rotate them, "AdaptW" is the best variant and the poorest variants are "ConstantW" and "ConstrFactor."

It is clear that the best modification of the particle swarm optimization algorithm for the selected benchmark functions

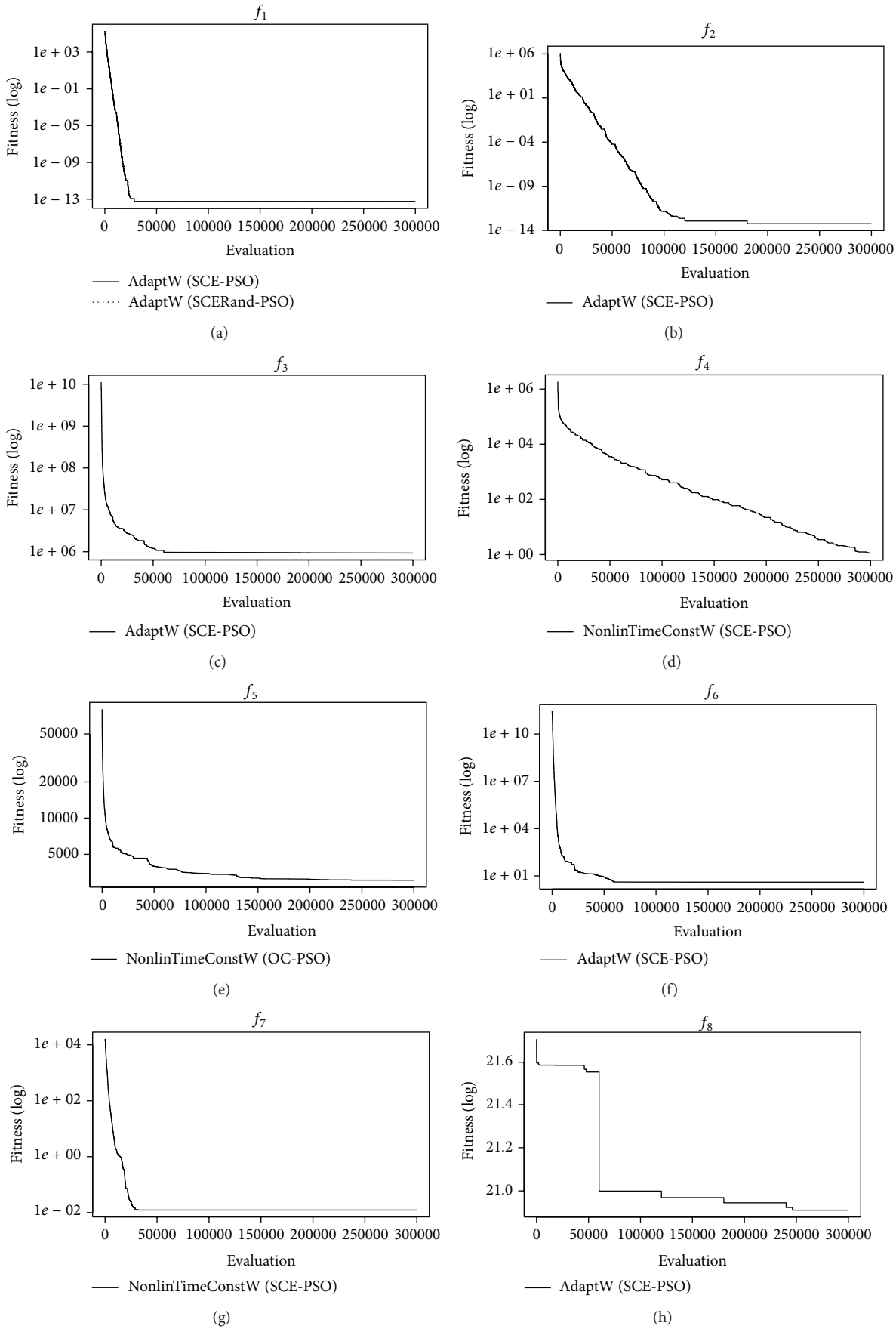

Figure 1: Continued.

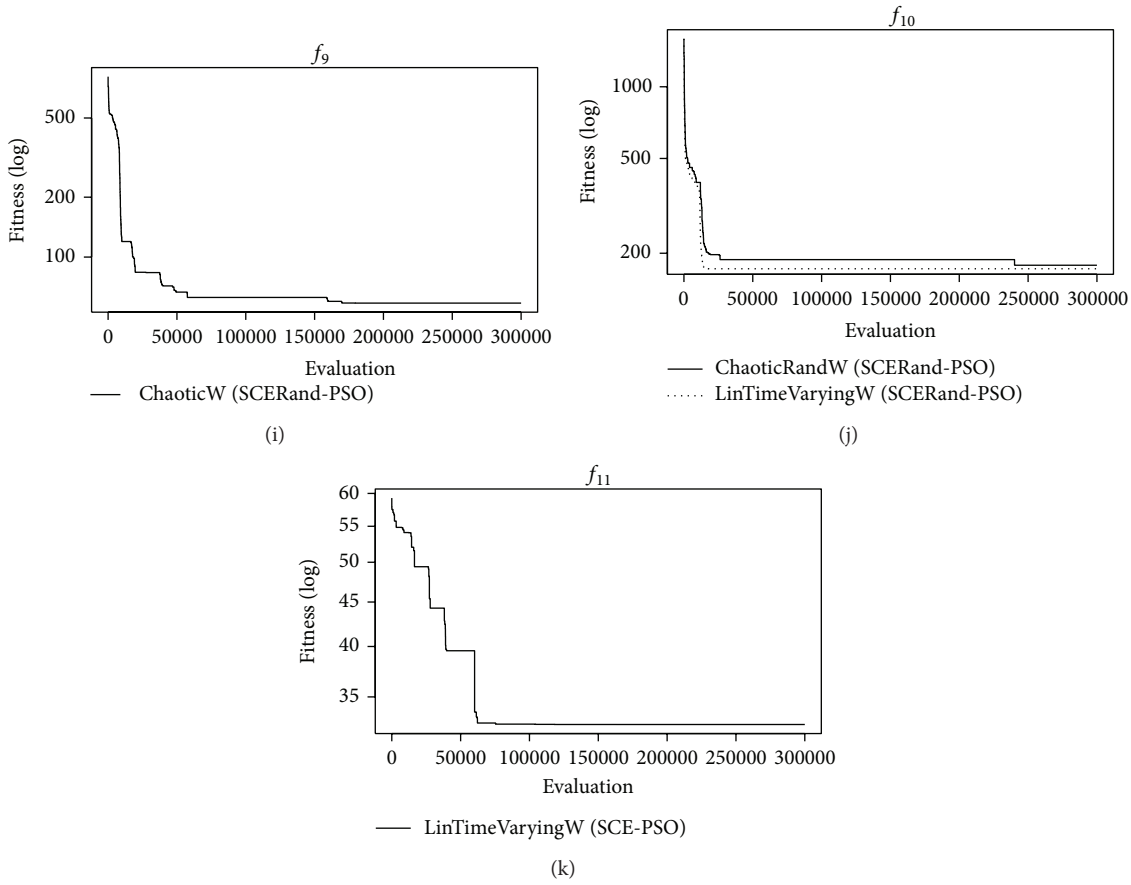

Figure 1: Convergence graphs for best fitness of the best modifications and strategy of PSO for functions f_1–f_{11} according to Table 3.

is "AdaptW," that is, adaptive inertia weight. The variant called "NonlinTimeConstW" also produced good results. On the other hand, the poorest modifications appear to be "ConstantW" and "ConstrFactor."

The convergence to the global optimum using particle swarm optimization is good, but only in three of the eleven benchmark functions is the obtained error value less than 10^{-8}. In spite of this, the shuffling mechanism improves the optimization. Strategies SCE-PSO and SCERand-PSO are better than OC-PSO in ten functions.

The global minimum was achieved in three benchmark functions, f_1, f_2, and f_7, and our results are comparable with those of Hansen [45] who compares eleven optimization algorithms on twelve functions. The particle swarm optimization algorithm achieved the global minimum in functions f_1, f_2, f_3, f_6, and f_7. Our results using SCE-PSO are better than those of the optimization algorithms BLX-MA and DE and are as good as those from CoEVO as reported by Hansen [45].

Bui et al. [12] achieved the global optimum in two unimodal functions f_1 and f_2 using the PSO method APSO1 and in three unimodal functions f_1, f_2, and f_4 using the DPSO method. APSO is PSO algorithm with adaptive constriction factor and in DPSO the bound of the velocity is adapted. None of their algorithms converge to the global minimum in multimodal functions [12]. In this regard, our results are better.

Nickabadi et al. [13] compare six inertia weight adjusting methods on 15 test problems with dimension set to 30. If we look only at functions solved for this paper, Nickabadi et al. [13] achieved the global optimum in functions f_1, f_2, and f_8. They obtained the best results using the adaptive inertia weight, which they had proposed (modification "AdaptW" in this paper). Their results are comparable with those achieved in the present study. The difference being that our functions have shifted the global optimum, whereas Nickabadi et al. [13] did not.

Figure 1 presents the convergence graphs for each function while utilizing the best modification of the PSO algorithm. The x-axis indicates the number of function evaluations and the y-axis the logarithmic value of the best fitness, which is the difference between the searched and the best achieved functional value. A decline with the number of evaluations is clearly visible for all functions, thus indicating the approach to the global optimum.

5. Conclusions

This paper compared 27 variants of particle swarm optimization algorithm. Eight modifications were performed using the parameter inertia weight and one modification using

constriction factor. Both parameters improved the optimization. All modifications were tested with three strategies of swarm distribution, which were in terms of population. The population was either considered as a single unit (OC-PSO) or it was divided into several complexes. Division into complexes was made according to the functional value of each particle (SCE-PSO) or through random permutation (SCERand-PSO).

The main aim of this work was to find the global minima of eleven benchmark functions prepared for the special session on real-parameter optimization of CEC 2005. The achievement of the minimum is when the obtained error value is less than 10^{-8}. We obtained the global minimum in two unimodal functions (f_1 and f_2) and in one multimodal function (f_7). The original particle swarm optimization has slow convergence to the global optimum, but the shuffling mechanism improves the optimization.

The best modification of the PSO algorithm is the variant called "AdaptW." The best choice for selected benchmark functions is to use the parameter of inertia weight, where the w value is adapted based on a feedback parameter. The best strategy for swarm distribution is SCE-PSO. Shuffled complex evolution particle swarm optimization with allocation of particles into complexes according to their functional values is better than OC-PSO and SCERand-PSO.

Conflict of Interests

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

Acknowledgments

The authors are grateful to the anonymous referees for their valuable comments and suggestions for improving the presentation of this paper. This work was supported by the Internal Grant Agency, Faculty of Environmental Sciences, Czech University of Life Sciences Prague (Project no. 42200131231582014422 7).

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Australia, December 1995.

[2] T. Weise, *Global Optimization Algorithms—Theory and Application*, 2009, http://www.it-weise.de/projects/bookNew.pdf.

[3] 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.

[4] N. Gershenfeld, *The Nature of Mathematical Modeling*, Cambridge University Press, New York, NY, USA, 1999.

[5] R. Mendes, *Population topologies and their influence in particle swarm performance [Ph.D. thesis]*, University of Minho, 2004.

[6] Z. Michalewicz and D. Fogel, *How to Solve It: Modern Heuristics*, Springer, New York, NY, USA, 2004.

[7] A. M. Baltar and D. G. Fontane, "Use of multiobjective particle swarm optimization in water resources management," *Journal of Water Resources Planning and Management*, vol. 134, no. 3, pp. 257–265, 2008.

[8] M. K. Gill, Y. H. Kaheil, A. Khalil, M. McKee, and L. Bastidas, "Multiobjective particle swarm optimization for parameter estimation in hydrology," *Water Resources Research*, vol. 42, no. 7, 2006.

[9] D. M. Munoz, C. H. Llanos, L. D. S. Coelho, and M. Ayala-Rincon, "Opposition-based shuffled PSO with passive congregation applied to FM matching synthesis," in *Proceedings of the IEEE Congress of Evolutionary Computation (CEC '11)*, pp. 2775–2781, IEEE, New Orleans, La, USA, June 2011.

[10] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, IEEE Computer Society, Washington, DC, USA, May 1998.

[11] Y. Shi and R. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation*, pp. 1945–1950, Washington, DC, USA, 1999.

[12] L. T. Bui, O. Soliman, and H. A. Abbass, "A modified strategy for the constriction factor in particle swarm optimization," in *Progress in Artificial Life: Proceedings of the 3rd Australian Conference; ACAL 2007 Gold Coast, Australia, December 4–6, 2007*, vol. 4828 of *Lecture Notes in Computer Science*, pp. 333–344, Springer, Berlin, Germany, 2007.

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

[14] Q. Y. Duan, V. K. Gupta, and S. Sorooshian, "Shuffled complex evolution approach for effective and efficient global minimization," *Journal of Optimization Theory and Applications*, vol. 76, no. 3, pp. 501–521, 1993.

[15] J. A. Vrugt, B. A. Robinson, and J. M. Hyman, "Self-adaptive multimethod search for global optimization in real-parameter spaces," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 243–259, 2009.

[16] M. Pant, R. Thangaraj, and A. Abraham, "Particle swarm optimization: performance tuning and empirical analysis," in *Foundations of Computational Intelligence*, A. Abraham, A.-E. Hassanien, P. Siarry, and A. Engelbrecht, Eds., vol. 3 of *Studies in Computational Intelligence*, pp. 101–128, Springer, Berlin, Germany, 2009.

[17] R. Hassan, B. Cohanim, O. D. Weck, and G. Venter, "A comparison of particle swarm optimization and the genetic algorithm," in *Proceedings of the 1st AIAA Multidisciplinary Design Optimization Specialist Conference*, pp. 1–13, 2005.

[18] F. V. D. Bergh, *An analysis of particle swarm optimizers [Ph.D. thesis]*, University of Pretoria, 2001.

[19] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '00)*, vol. 1, pp. 84–88, IEEE, La Jolla, Calif, USA, July 2000.

[20] R. Eberhart, P. Simpson, and R. Dobbins, *Computational Intelligence PC Tools*, Academic Press Professional, San Diego, Calif, USA, 1996.

[21] D. Corne, M. Dorigo, F. Glover et al., Eds., *New Ideas in Optimization*, McGraw-Hill, Maidenhead, UK, 1999.

[22] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the Congress on Evolutionary Computation*, vol. 1, pp. 81–86, Seoul, Republic of Korea, May 2001.

[23] J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon, and A. Abraham, "Inertia weight strategies in particle swarm optimization," in *Proceedings of the 3rd World Congress on Nature and Biologically Inspired Computing (NaBIC '11)*, pp. 633–640, Salamanca, Spain, October 2011.

[24] J. Gimmler, T. Stützle, and T. E. Exner, "Hybrid particle swarm optimization: an examination of the influence of iterative improvement algorithms on performance," in *Ant Colony Optimization and Swarm Intelligence: Proceedings of the 5th International Workshop, ANTS 2006, Brussels, Belgium, September 4–7, 2006*, vol. 4150 of *Lecture Notes in Computer Science*, pp. 436–443, Springer, Berlin, Germany, 2006.

[25] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proceedings of the Congress on Evolutionary Computation*, vol. 1, pp. 94–100, May 2001.

[26] J. Xin, G. Chen, and Y. Hai, "A particle swarm optimizer with multi-stage linearly-decreasing inertia weight," in *Proceedings of the International Joint Conference on Computational Sciences and Optimization (CSO '09)*, vol. 1, pp. 505–508, IEEE Computer Society, Sanya, China, April 2009.

[27] C.-H. Yang, C.-J. Hsiao, and L.-Y. Chuang, "Linearly decreasing weight particle swarm optimization with accelerated strategy for data clustering," *IAENG International Journal of Computer Science*, vol. 37, no. 3, p. 1, 2010.

[28] Y. Feng, G.-F. Teng, A.-X. Wang, and Y.-M. Yao, "Chaotic inertia weight in particle swarm optimization," in *Proceedings of the 2nd International Conference on Innovative Computing, Information and Control (ICICIC '07)*, p. 475, Kumamoto, Japan, September 2007.

[29] M. Clerc, "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation*, pp. 1951–1957, Washington, DC, USA, 1999.

[30] P. J. Angeline, "Using selection to improve particle swarm optimization," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 84–89, Anchorage, Alaska, USA, May 1998.

[31] K. E. Parsopoulos, V. P. Plagianakos, G. D. Magoulas, and M. N. Vrahatis, "Improving particle swarm optimizer by function stretching," in *Advances in Convex Analysis and Global Optimization*, pp. 445–457, Springer, New York, NY, USA, 2001.

[32] K. E. Parsopoulos and M. N. Vrahatis, "Initializing the particle swarm optimizer using the nonlinear simplex method," in *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, pp. 216–221, WSEAS Press, 2002.

[33] J. Yan, H. Tiesong, H. Chongchao, W. Xianing, and G. Faling, "A shuffled complex evolution of particle swarm optimization algorithm," in *Adaptive and Natural Computing Algorithms: Proceedings of the 8th International Conference, ICANNGA 2007, Warsaw, Poland, April 11–14, 2007, Part I*, vol. 4431 of *Lecture Notes in Computer Science*, pp. 341–349, Springer, Berlin, Germany, 2007.

[34] V. C. Mariani, L. G. Justi Luvizotto, F. A. Guerra, and L. D. S. Coelho, "A hybrid shuffled complex evolution approach based on differential evolution for unconstrained optimization," *Applied Mathematics and Computation*, vol. 217, no. 12, pp. 5822–5829, 2011.

[35] M. Weber, F. Neri, and V. Tirronen, "Shuffle or update parallel differential evolution for large-scale optimization," *Soft Computing*, vol. 15, no. 11, pp. 2089–2107, 2011.

[36] R. Durstenfeld, "Algorithm 235: random permutation," *Communications of the ACM*, vol. 7, no. 7, p. 420, 1964.

[37] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[38] G. D. Wyss and K. H. Jorgensen, *A User's Guide to LHS: Sandias Latin Hypercube Sampling Software*, Sandia National Laboratories, 1998.

[39] R. Eberhart and Y. Shi, *Computational Intelligence: Concepts to Implementations*, Morgan Kaufmann, Boston, Mass, USA, 2007.

[40] P. N. Suganthan, N. Hansen, J. J. Liang et al., "Problem definitions and evaluation criteria for the CEC, 2005 special session on real-parameter optimization," Tech. Rep., Nanyang Technological University, Singapore, 2005.

[41] J. C. F. Cabrera and C. A. C. Coello, "Handling constraints in particle swarm optimization using a small population size," in *MICAI 2007: Advances in Artificial Intelligence: Proceedings of the 6th Mexican International Conference on Artificial Intelligence, Aguascalientes, Mexico, November 4–10, 2007*, vol. 4827 of *Lecture Notes in Computer Science*, pp. 41–51, Springer, Berlin, Germany, 2007.

[42] K. Parsopoulos and M. N. Vrahatis, "On the Computation of all global minimizers through particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 211–224, 2004.

[43] C. Dytham, *Choosing and Using Statistics: A Biologist's Guide*, Blackwell Science, Oxford, UK, 2011.

[44] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms'behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.

[45] N. Hansen, "Compilation of results on the 2005 CEC benchmark function set," Online, 2006.