Available online at www.sciencedirect.com
ScienceDirect
Applied Soft Computing 8 (2008) 295–304
Applied Soft Computing
www.elsevier.com/locate/asoc

The landscape adaptive particle swarm optimizer

Jin Yisu [a], Joshua Knowles [b,*], Lu Hongmei [a], Liang Yizeng [a], Douglas B. Kell [b]

[a] College of Chemistry and Chemical Engineering, Central South University, Changsha 410083, PR China
[b] School of Chemistry, University of Manchester, Manchester M60 1QD, UK

Received 12 December 2005; received in revised form 12 August 2006; accepted 2 January 2007
Available online 16 February 2007

Abstract

Several modified particle swarm optimizers are proposed in this paper. In DVPSO, a distribution vector is used in the update of velocity. This vector is adjusted automatically according to the distribution of particles in each dimension. In COPSO, the probabilistic use of a ‘crossing over’ update is introduced to escape from local minima. The landscape adaptive particle swarm optimizer (LAPSO) combines these two schemes with the aim of achieving more robust and efficient search. Empirical performance comparisons between these new modified PSO methods, and also the inertia weight PSO (IFPSO), the constriction factor PSO (CFPSO) and a covariance matrix adaptation evolution strategy (CMAES) are presented on several benchmark problems. All the experimental results show that LAPSO is an efficient method to escape from convergence to local optima and approaches the global optimum rapidly on the problems used.

Keywords: Particle swarm optimization; LAPSO; Evolution strategy

1. Introduction

1.1. Particle swarm optimization

Particle swarm optimization (PSO) is a population-based global optimization method based on a simple simulation of bird flocking or fish schooling behavior [1]. In PSO, the search points are known as particles, and each particle is initialized with a random position and random initial velocity in the D-dimensional search space. The position and velocity of all the particles are usually updated synchronously in each iteration of the algorithm. A particle adjusts its velocity according to its own flight experience and the flight experience of other particles in the swarm in such a way that it accelerates towards positions that have had high objective (fitness) values in previous iterations.

There are two kinds of position towards which a particle is accelerated in common use. The first one, a particle's personal best position achieved up to the current iteration, is called $\vec{p}_{\text{best}}$. The other is the global best position obtained so far by all particles, called $\vec{g}_{\text{best}}$. Kennedy and Eberhart [1] devised the following update rule which forms the kernel of PSO.

$$\vec{v}(t+1) = \vec{v}(t) + c_1\vec{R}_1(\vec{p}_{\text{best}} - \vec{x}(t)) + c_2\vec{R}_2(\vec{g}_{\text{best}} - \vec{x}(t)), \tag{1}$$

where $\vec{v}$ is the velocity of a particle and $\vec{x}$ its position and $\vec{R}_1$ and $\vec{R}_2$ are random numbers in the range [0, 1] with the same size of the swarm population. c_1 and c_2 are learning factors which will be fixed through the whole process.

The new position for the particles is the addition of the position at time t and the distance that the particles will fly with the new velocity. The synchronous update of position is thus:

$$\vec{x}(t+1) = \vec{x}(t) + \vec{v}(t+1) \tag{2}$$

The pseudo code of PSO is shown in Fig. 1.

In order to improve the local search precision, Shi and Eberhart [2] introduce the inertia weight w to Eq. (1) to give the following update rule:

$$\vec{v}(t+1) = \omega\vec{v}(t) + c_1\vec{R}_1(\vec{p}_{\text{best}} - \vec{x}(t)) + c_2\vec{R}_2(\vec{g}_{\text{best}} - \vec{x}(t)) \tag{3}$$

A value of w decreasing linearly from 0.95 to 0.4 is the best reduction strategy of those tested in [2], over their suite of test functions. In Chatterjee's work [3], a dynamic change of inertia weight is suggested.

* Corresponding author at: School of Chemistry, University of Manchester, Sackville Street, P.O. Box 88, Manchester M60 1QD, UK.
Tel.: +44 161 306 4450; fax: +44 161 306 4556.
E-mail address: j.knowles@manchester.ac.uk (J. Knowles).

1568-4946/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2007.01.009

```
Initialize both swarm x and velocity v, iteration = 1,
Do
    Evaluate swarm using corresponding fitness function,
    Update the personal best position $\vec{p}_{best}$ and the global best position $\vec{g}_{best}$,
    Update the velocity v using Eq.(1),
    Update the position of swarm x using Eq. (2),
    iteration = iteration + 1,
Until terminati oncriterion is met.
```

Fig. 1. The pseudo code of PSO.

Clerc [4] indicates that the use of a constriction factor K may also be necessary to ensure convergence of the particle swarm algorithm, defined as when all particles have stopped moving. Their update rule for velocity is:

$$\vec{v}(t+1) = K[\vec{v}(t) + c_1\vec{R}_1(\vec{p}_{\text{best}} - \vec{x}(t)) + c_2\vec{R}_2(\vec{g}_{\text{best}} - \vec{x}(t))] \tag{4}$$

$$K = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \tag{5}$$

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

In addition to the global version of PSO, Kennedy [5] has also introduced the use of a local variable $(\vec{l}_{\text{best}})$, which confers more ability to escape from local optima. In this approach, $\vec{g}_{\text{best}}$ is replaced by $\vec{l}_{\text{best}}$ in Eq. (1):

$$\vec{v}(t+1) = \vec{v}(t) + c_1\vec{R}_1(\vec{p}_{\text{best}} - \vec{x}(t)) + c_2\vec{R}_2(\vec{l}_{\text{best}} - \vec{x}(t)), \tag{6}$$

where $\vec{l}_{\text{best}}$ is the best position achieved by a ‘neighboring’ particle and the definition of the neighborhood varies in different implementations of the approach.

In recent research, Xu and Xin [6] point out that the combined use of $\vec{g}_{\text{best}}$ and $\vec{l}_{\text{best}}$ may be helpful for the search process, and the velocity should still be constricted by the constriction factor:

$$\vec{v}(t+1) = K[\vec{v}(t) + c_1\vec{R}_1(\vec{p}_{\text{best}} - \vec{x}(t)) + c_2\vec{R}_2(\vec{g}_{\text{best}} - \vec{x}(t)) + c_3\vec{R}_3(\vec{l}_{\text{best}} - \vec{x}(t))] \tag{7}$$

Fieldsend and Singh [7] introduce a stochastic variable, turbulence, into the standard PSO. He et al. [8] modify the stochastic variable to a passive congregation which improves the PSO from both accuracy and convergence rate. Leontitsis et al. [9] also put forward a concept of repellor in their paper. The authors believe that the worst particles have the property of repelling the particles to the local optima.

1.2. Related research

As the development of PSO, lots of techniques are introduced into PSO, such as simplex search, evolutionary algorithm, genetic algorithm and support vector machine, etc. in the literature [10–13]. Various PSOs have been applied to different research fields. In reference [14], PSO is implemented to the parameter identification of the induction motor. Lin et al. apply the modified PSO on the quantitative structure-activity relationship (QSAR) models [15]. PSO is also implemented for drug design [16]. In some research work, PSO, which is initially dealt with a single-objective function, has also been extended to deal with multi-objective problems [17–19].

Most of the PSO studies are empirical. To gain a better, general understanding of the behavior of particle swarms, theoretical analyses of particle trajectories are necessary. A few theoretical studies of particle trajectories can be found [20,21]. These studies facilitated the derivation of heuristics to select parameter values for guaranteed convergence to a stable point, which is shown to be a weighted average of the personal best and global best positions, where the weights are determined by the values of the acceleration coefficients [22].

A comprehensive review is beyond the scope of this article.

1.3. Evolution Strategy

In the 1960s, the evolution strategy (ES) was originated as a set of rules for the automatic design and analysis of consecutive experiments with stepwise variable adjustments driving a suitably flexible object/system into its optimal state in spite of environmental noise [23].

Initially, Rechenberg [24] developed the $(1+1)$-ES, a simple mutation plus selection scheme operating on one individual that creates one offspring per generation by means of Gaussian mutation. He also proposed a $(\mu+1)$-ES where $\mu \geq 1$ parent individuals recombine to form one offspring, which, after mutation, eventually replaces the worst parent individual. This strategy is thought to be the foundation of the well-known $(\mu+\lambda)$-ES and (μ, λ)-ES introduced and investigated by Schwefel [25,26], which became the state-of-the-art in ES research [27].

In an evolution strategy, the population is randomly initialized. Then a number of generations involving recombination, mutation and selection are performed. Selection is based on the fitness value of each individual. The best individual from the offspring population ((μ, λ)-ES), or parent and offspring populations ($(\mu+\lambda)$-ES) are selected to continue: this is called truncation selection.

In each ES generation, λ offspring are generated from the set of μ parent individuals. Parameters μ and λ as well as ρ (the mixing number which defines the size of the parent family that is chosen from the parent pool of size μ to create λ offspring) are called “exogenous strategy parameters”, and are kept constant during the evolution run. In ESs there are also *“endogenous* strategy parameters”, which are used to control the genetic operators, especially those of the mutation operator. Endogenous strategy parameters can evolve during the evolution process and are needed in a self-adaptive ES.

During the past years, the improvement of self-adaptation led to the development of the Covariance Matrix Adaptation (CMA) [28]. The objective of CMA is to fit the search distribution to the contour lines of the objective function f to be minimized. Here, the (μ, λ)-ES with covariance matrix adaptation (CMA-ES) is treated as a representative ES to compare with the PSO methods described above and those introduced next.

2. Development of landscape adaptive particle swarm optimizer

2.1. Distribution vector

In the original PSO algorithm, the flight of each particle depends only upon its own history and the influence of a small number of leading particles. The behavior of the population as a whole is not taken into account.

We make the hypothesis that the distribution of the whole group may provide important additional information. Considering an asymmetrical space, for example a flat rectangle, the distribution of randomly moving particles should most probably be close to a rectangle. This means that on the longer direction the particles may be more scattered; while on the shorter direction, the particles may be closer together. In this kind of space, PSO may slow down and even fail to find the optimum if the particles move uniformly in each direction. Thus, we suggest that if the velocities of the particles were responsive to the shape of the manifold in which the particles lie, better performance may be achieved.

During the search process in PSO, the distribution of particles keeps changing. We define a distribution vector $\vec{D}$ to describe the particles' spatial arrangement at a given time:

$$D_j = \frac{\max_{i=1}^{n}(x_{ij}) - \min_{i=1}^{n}(x_{ij})}{\mathrm{abs}(\max_{i=1}^{n}(x_{ij})) + \mathrm{abs}(\min_{i=1}^{n}(x_{ij}))} \quad (j = 1, 2, \ldots, m), \tag{8}$$

where m is the dimension of particles and n is the population size. The dimension of $\vec{D}$ is m.

The value of each component of the distribution vector $\vec{D}$ is between 0 and 1. A large component gives the information that in this direction, particles extend widely.

Compared to the constriction factor update, the readjustment of velocity with the distribution vector $\vec{D}$ depends not only on the numerical value, but also on the direction.

Thus, the update of velocity becomes:

$$\vec{v}(t+1) = \vec{D}[\vec{v}(t) + c_1\vec{R}_1(\vec{p}_{\mathrm{best}} - \vec{x}(t)) + c_2\vec{R}_2(\vec{g}_{\mathrm{best}} - \vec{x}(t))]. \tag{9}$$

Notice, however, that if many particles become trapped in a local minimum, the distribution vector will be very small, near zero, and so the ability to adjust the velocities will be lost. This would lead to premature convergence. Meanwhile, too large velocities might cause oscillation within the search manifold and make it unlikely for the PSO to find minima.

In order to avoid the situation mentioned above, we modify the definition of distribution vector $\vec{D}$ as following:

$$D_j = \begin{cases} D\_\max, & \text{if } D_j < D\_\min, \\ D\_\min, & \text{if } D_j > D\_\max, \\ D_j & \text{otherwise.} \end{cases} \tag{10}$$

Numerical values of these two parameters are discussed in Section 3.

The modified PSO with the application of distribution vector is called DVPSO here.

2.2. Crossing over ability

All the particles in the PSOs mentioned above tend to follow the best particles but in different ways. We can easily imagine that if the leader is not good enough, say, falling in to local optimum, particles following the leader will be adversely influenced. Thus, the information from the leaders does not always have positive effects.

Considering of this, a possibility to escape the misdirecting influence should be allowed. In reference [29], some randomly flying particles have been used to avoid the premature convergence. But when the dimension of the searching space is very large, the distribution of several free particles will be too sparse to significantly increase the probability to find a new optimum.

In our method, we give the opportunity not to follow the leader, but instead crossing over with the leader as follows:

$$\vec{v}(t+1) = \vec{v}(t) + c_1\vec{R}_1(\vec{p}_{\mathrm{best}} + \vec{x}(t)) + c_2\vec{R}_2(\vec{g}_{\mathrm{best}} + \vec{x}(t)). \tag{11}$$

After experimental tests, a probability of 0.1 of using this update, while in 0.9 cases using Eq. (1) to update the velocity, was observed to result in good performance. This modified PSO method is called COPSO because of the so-called crossing over ability.

2.3. Landscape adaptive PSO

Putting the two update rules described above together, results in a new method hypothesized to have both organization and innovation ability. We call this modified PSO, the landscape adaptive PSO (LAPSO).

In the LAPSO, the velocity of next step is generated from the distribution of present particles with the introduction of the distribution vector which can ensure better spread of particles according their distribution in the searching space. And lower bound and upper bound for the distribution vector are provided to avoid local convergence and unstable oscillation. Considering of the possible ill effect of local minima, a probability of 0.1 to step out of the leadership for the particle group is given as shown in Eq. (11). So there are two effective techniques, the limitation of the distribution vector and the crossing over ability, to escape from the local minima in LAPSO.

We also made some small modification on the particles which are out of the searching area. In the searching progress, some particles will leave the searching area. Usually, they are limited to the exact boundary as follows:

$$\begin{aligned} &\vec{x}(\vec{x} < \text{lower bound}) = \text{lower bound} \\ &\vec{x}(\vec{x} > \text{upper bound}) = \text{upper bound} \end{aligned} \tag{12}$$

Here, we recommend redistributing these particles as:

$$\begin{aligned}\vec{x}\,(\vec{x} < \text{lower bound}) &= \text{lower bound} + \text{rand}(\text{upper bound} - \text{lower bound})\\ \vec{x}\,(\vec{x} > \text{upper bound}) &= \text{upper bound} - \text{rand}(\text{upper bound} - \text{lower bound})\end{aligned} \tag{13}$$

The pseudo code of the LAPSO is given in Fig. 2.

3. Experiments

We test the PSO methods, DVPSO, COPSO and LAPSO, introduced above on a suite of benchmark functions and compare them with the inertia weight PSO (IWPSO), Constriction Factor PSO (CFPSO) and CMA-ES (running each algorithm 100 times for each function).

3.1. Benchmark functions

We use the eight non-linear benchmark functions given in Table 1. The functions selected are very often encountered in optimization algorithm benchmarks. They are named after the authors that introduced them as benchmark functions. Functions f_1 to f_5 are origin-centered functions. Functions f_6 to f_8 have minimal solutions not at the origin which are believed to be more difficult to obtain the optima if bias exists in the searching procedure.

In real-world problems, there are unimodal and multimodal problems, with correlated or uncorrelated variables. Most of the benchmark functions, with distinct characteristics, are put forward to mimic the real problems. De Jong's Sphere function f_1 is the most basic problem. It contains no local optima and provides a smooth gradient towards a broad global optimum. The Griewank function, f_2, introduces interdependency between the variables; this is why, this function disrupts the optimization techniques working on one variable at a time. The Rastrigin function, f_3, has lattice-shaped semi-optimum solutions around the global optima, and there is no correlation among design variables. The Ackley function, f_4, is also multimodal at low resolution. The search space defined by the De Jong F2 function (f_6) is unimordal and has correlation among its design variables. Schwefel function f_7 has a semi-optimum solution far from the global optima where many search algorithms are trapped. Moreover, the global optimum exists near the bounds of the domain. There is no correlation among its design variables [30].

Table 1
Standard multi-modal objective functions

No.	Function	Equation (f^* gives the minima)	Parameters
f_1	DeJong F1	$f(\vec{x}) = \sum_{i=1}^{n} x_i^2, \quad f^*(0,0,\ldots,0) = 0$	Dimensions = 30 $X_{\max} = 100$
f_2	Griewank	$f(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \quad f^*(0,0,\ldots,0) = 0$	Dimensions = 30 $X_{\max} = 600$
f_3	Rastrigin	$f(\vec{x}) = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10), \quad f^*(0,0,\ldots,0) = 0$	Dimensions = 30 $X_{\max} = 5.12$
f_4	Ackley	$f(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{30} x_i^2}\right) - \exp\left(\frac{1}{30}\sum_{i=1}^{30}\cos 2\pi x_i\right) + 20 + \mathrm{e},$ $f^*(0,0,\ldots,0) = 0$	Dimensions = 30 $X_{\max} = 32$
f_5	Schaffer F6	$f(x, y) = 0.5 - \frac{(\sin\sqrt{x^2+y^2})^2 - 0.5}{(1.0+0.001(x^2+y^2))^2}, \quad f^*(0, 0) = 0$	Dimensions = 2 $X_{\max} = 100$
f_6	DeJong F2	$f(x, y) = 100(x^2 - y)^2 + (1 - x)^2, \quad f^*(1, 1) = 0$	Dimensions = 2 $X_{\max} = 100$
f_7	Schwefel	$f(\vec{x}) = \sum_{i=1}^{n}(x_i \sin(\sqrt{\lvert x_i\rvert})), \quad f^*(420.9687, 420.9687, \ldots, 420.9687) = -12,569.5$	Dimensions = 30 $X_{\max} = 500$
f_8	Foxholes	$\mathit{left}\{\, f(\vec{x}) = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right]^{-1},$ $(a_{i,j}) = \begin{pmatrix} -32, -16, 0, 16, 32, -32, \ldots, 32, -32, -16, 0, 16, 32 \\ -32, \ldots, -32, -16, \ldots, -16, 0, \ldots, 0, 16, \ldots, 16, 32, \ldots, 32 \end{pmatrix}$ $(i = 1, 2;\ j = 1, 2, \ldots, 25) \quad f^*(-32, -32) = 0.998\}$	Dimensions = 2 $X_{\max} = 65.536$

$X_{\max}$ gives the limitation of search space of X, which means $-X_{\max} \le X \le X_{\max}$. For each PSO method, $V_{\max}$ is equal to $X_{\max}$.

```
Initialize both swarm x and velocity v, iteration = 1,
Do
    Evaluate swarm using corresponding fitness function,
    Update the personal best position $\vec{p}_{best}$ and the global best position $\vec{g}_{best}$,
    For a random number r,
    If r < 0.1
        Update the velocity v using Eq.(11),
    Else
        Calculate the distribution factor $\vec{D}$ using Eq.(8) and Eq.(10),
        Update the velocity v using Eq.(9),
    End If
    Update the position of swarm x using Eq. (2),
    Restrict the swarm within the searching area using Eq.(13),
    iteration = iteration + 1,
Until termination criterion is met.
Output the $\vec{g}_{best}$ and its fitness value of all iterations.
```

Fig. 2. The pseudo code of LAPSO.

3.2. Parameter settings

3.2.1. Parameters of PSO

In PSO, there are not many parameters to be tuned. Some parameters are determined by the problem, such as the dimension of particles, the range of particles and the stopping condition. Since theoretical guidelines are still absent, experimental work is necessary to find the appropriate parameters. We have taken into account some earlier results reported in the literature to choose parameter settings, as follows.

In most cases, increasing the number of particles decreases the number of required algorithm iterations [31]. But more particles require more function evaluations. The typical range is suggested as [20–40].

$V_{\max}$ determines the maximum change one particle can take. Usually, $V_{\max}$ may be set as $X_{\max}$. Although the constriction factor is proposed to overcome the limit of $V_{\max}$, it has been reported that better results may be obtained while setting $V_{\max}$ as $X_{\max}$ [32].

Learning factors: usually c_1 is set equal to c_2 (popularly being 2) and ranges from [0,4] following the suggestion of Carlisle and Dozier [33]. He also suggests that $c_1 = 2.8$ and $c_2 = 1.3$ produce good results. Jiang explains the parameter selection guideline by stochastic convergence analysis [34].

Global version versus local version: the global versions faster but might converge to a local optimum for some problems; the local version is a little bit slower but not easily trapped into a local optimum. When using the local PSO version, 2 is thought to be enough for the number of neighbors.

Given these considerations, our detailed parameter settings are as following:

(i) In all the tested methods, the global version is adopted.
(ii) $c_1 = c_2 = 2.00$.
(iii) The determination of global best solution is implemented asynchronously.
(iv) In each experiment, the population of particles is set as 30 thus there will be 30 function evaluations for each repetition.
(v) The number of iterations is set as 5000. The number of repetitions is set as 100.
(vi) In IWPSO, the inertia weight is decreased linearly from 0.95 to 0.4. Based on some experimental tests (data not shown), the decreasing velocity is set as (0.95–0.4)/1000. That means the inertia weight will meet 0.4 after 1000 iterations and keep this value until termination.
(vii) In LAPSO, the value of D_max and D_min is important. They are used to limit the magnitude of $\vec{D}$ which is a multiplier of the velocity. According to the research of inertia weight and constriction factor [32], such a multiplier should be in the range of [0.4–1.0]. Based on this result, several combinations of D_max and D_min were tested. These experiments showed that D_max = 0.95, D_min = 0.4 are good settings (data not shown).

3.2.2. Parameters of CMA-ES

The population of the offspring number λ is set as 30 in order to keep the function evaluations consistent with PSO methods. The population of the parents, μ is 15.

For the other parameters we use the original calculations reported in Hanson's work [35].

3.3. Experiments

3.3.1. Experiment No.1

For each function, the stopping condition is the maximum number of iterations set at 5000. Test results are shown in Fig. 3 which gives the performance of different algorithms for each function over 100 repetitions.

3.3.2. Experiment No.2

The final minimal fitness values after 5000 iterations are analyzed using the Kruskal–Wallis test (K–W test) [36]. The test statistic T is defined as:

$$T = \frac{1}{S^2}\left(\sum_{i=1}^{k}\frac{R_i^2}{n_i} - N\frac{(N+1)^2}{4}\right) \tag{14}$$

$$S^2 = \frac{1}{N-1}\left(\sum_{\text{all ranks}} R(X_{ij})^2 - N\frac{(N+1)^2}{4}\right) \tag{15}$$

where k is the number of groups, N the total number of samples, n_i the sample size for group I, R_i the sum of the ranks for group I and $R(X_{ij})$ is the rank of all samples.

Reject H_0 at the level α if T is greater than its $1-\alpha$ quantile from the null distribution.

Box plots of the distribution of fitness values are given in Fig. 4. The top of the notch is at the 75th percentile and bottom is at the 25th percentile. The median is plotted as a horizontal line inside the box. Lines extend from the box to the maximum and minimum values of the data, except when outliers are detected. An outlier is defined as any point that is

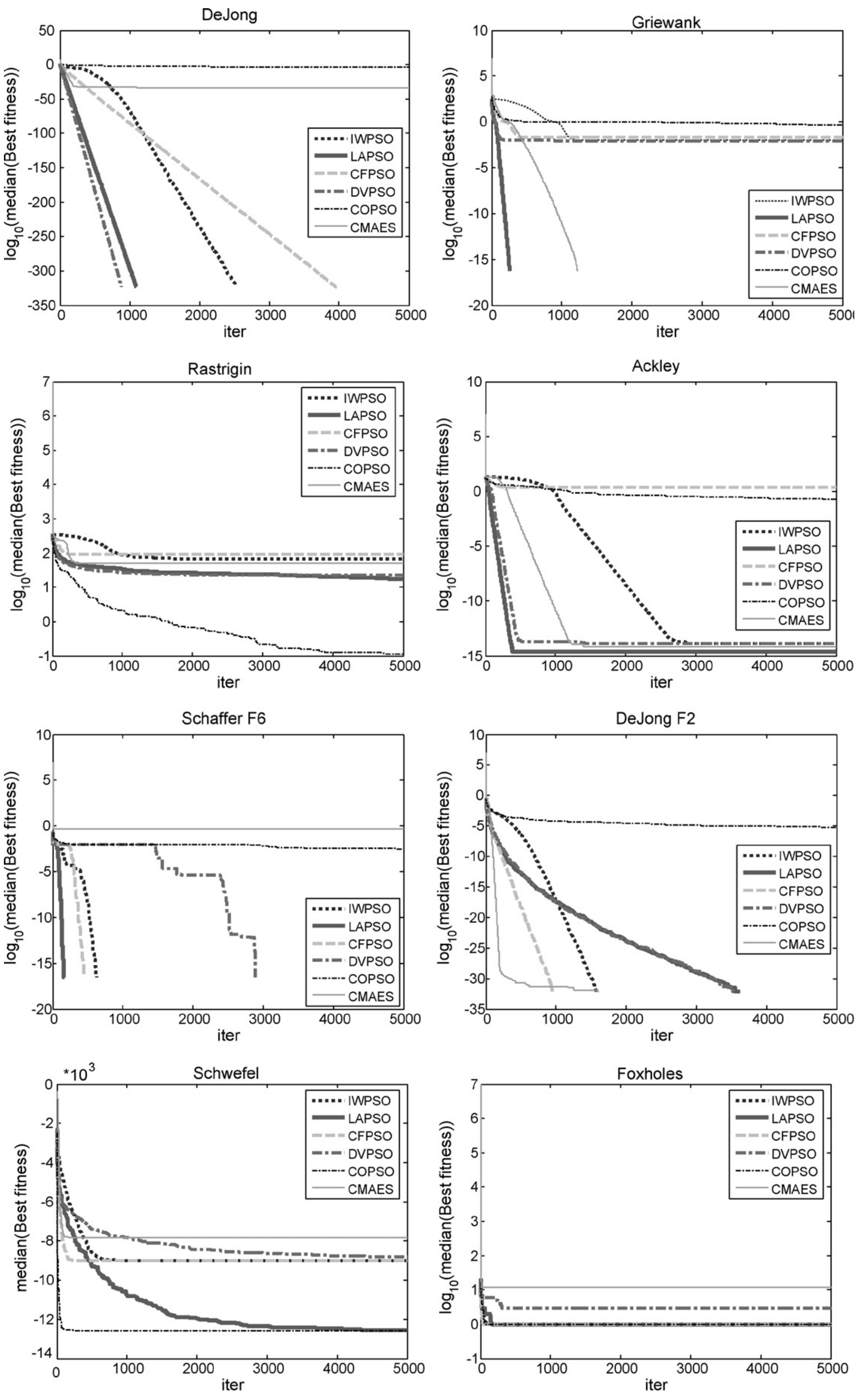

Fig. 3. Plots of performance of each stochastic method on different benchmark functions.

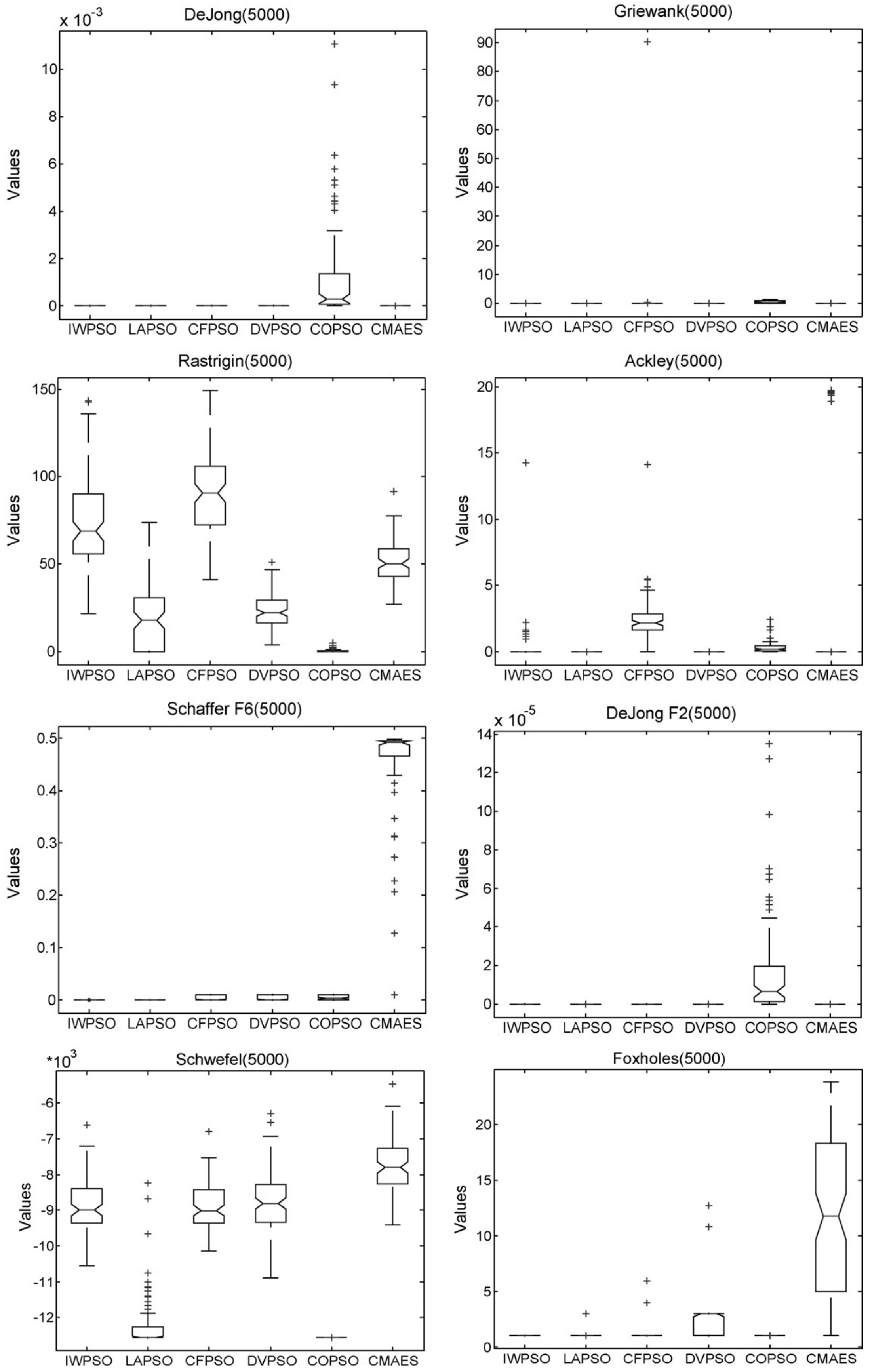

Fig. 4. Box plots of the distribution of the fitness values for each stochastic method on different benchmark functions.

Table 2
Multiple Comparison tests of each function

Optm_*i*-Optm_*j*	f_1 ($L = 3.8966$)	f_2 ($L = 24.945$)	f_3 ($L = 17.979$)	f_4 ($L = 18.736$)	f_5 ($L = 18.663$)	f_6 ($L = 9.7135$)	f_7 ($L = 20.073$)	f_8 ($L = 23.128$)
LAPSO-IWPSO	**0**	−175.3950	−280.3900	−194.7750	**0**	**7.6750**	−212.5300	213.2000
LAPSO-CFPSO	**0**	−222.6550	−331.5350	−475.2700	−93.6800	**7.6750**	−212.7400	201.7800
LAPSO-DVPSO	**0**	−147.4300	−41.8400	−253.6350	−104.9500	**5.1050**	−225.9600	**−7.8000**
LAPSO-COPSO	−350	−385.0700	98.6900	−387.3200	−237.0000	−300.8250	87.9100	**−19.6000**
LAPSO-CMAES	−250	−38.6300	−196.6050	−189.0000	−386.3700	−24.5800	−357.6800	−173.3800
IWPSO-CFPSO	**0**	−47.2600	−51.1450	−280.4950	−93.6800	**0**	**−0.2100**	**−11.4200**
IWPSO-DVPSO	**0**	27.9650	238.5500	−58.8600	−104.9500	**−2.5700**	**−13.4300**	−221.0000
IWPSO-COPSO	−350	−209.6750	379.0800	−192.5450	−237.0000	−308.5000	300.4400	−232.8000
IWPSO-CMAES	−250	136.7650	83.7850	5.7750	−386.3700	−32.2550	−145.1500	−386.5800
CFPSO-DVPSO	**0**	75.2250	289.6950	221.6350	**−11.2700**	**−2.5700**	**−13.2200**	−209.5800
CFPSO-COPSO	−350	−162.4150	430.2250	87.9500	−143.3200	−308.5000	300.6500	−221.3800
CFPSO-CMAES	−250	184.0250	134.9300	286.2700	−292.6900	−32.2550	−144.9400	−375.1600
DVPSO-COPSO	−350	−237.6400	140.5300	−133.6850	−132.0500	−305.9300	313.8700	**−11.8000**
DVPSO-CMAES	−250	108.8000	−154.7650	64.6350	−281.4200	−29.6850	−131.7200	−165.5800
COPSO-CMAES	100	346.4400	−295.2950	198.3200	−149.3700	276.2450	−445.5900	−153.7800

The values in the table are the left part of inequality (16) and L denotes the right part of this inequality. Numbers in bold show that there is insufficient evidence to reject the null hypothesis of no difference between the optimizers in the row.

above the 75th percentile by 1.5 times the interquartile interval, or similarly below the 25th percentile. When there are outliers, the outlying points are plotted with '+' and the lines extend only to the cutoff value for defining an outlier.

Multiple comparisons (MC) will be implemented only if the null hypothesis is rejected. The null hypothesis of no difference between a pair of optimizers may be rejected if the following inequality condition holds:

$$\left|\frac{R_i}{n_i} - \frac{R_j}{n_j}\right| > t_{1-(\alpha/2)}\left(S^2\frac{N-1-T}{N-k}\right)^{1/2}\left(\frac{1}{n_i}+\frac{1}{n_j}\right)^{1/2} \qquad (16)$$

where k is the number of groups and $t_{1-\alpha/2}$ is the $(1-\alpha/2)$ quantile of the t distribution with $N-k$ degrees of freedom.

The values of R_i/n_i–R_j/n_j are listed in Table 2.

4. Results and discussion

It can be seen in Fig. 3 that in almost all of the functions, LAPSO gives the fastest convergence toward the minimum and can find the global minima successfully. CMA-ES is thought to be a very efficient optimizer but only has visible advantage in function DeJong F2.

In function Rastrigin, Schwefel and Foxholes, COPSO is very efficient. We should notice that all these three functions contain many local minima. The results suggest that Crossing Over ability helps avoid local convergence. But COPSO failed in function Schaffer F6, which has many steep-sided basins of attraction. The reason could be that it lacks local searching ability which is very important especially when the searching area is very narrow and deep. In contrast, DVPSO which is thought to be efficient in local searching ability succeeds in this function and also in functions DeJong F2 and Ackley. But because of lacking exploring ability, DVPSO falls into local convergence as can be seen on other functions.

LAPSO with the two abilities combined is more robust to the above mentioned conditions. It displays a good performance on all the test functions.

In the box plots, the y-value of the narrowest part of each box shows the median of 100 fitness values for each method, while the extent of y-axis shows the reliability of the method to some degree.

In Table 2, with the results of KW–MC test, the difference between each algorithm is displayed. If the absolute value of the numbers in this table is smaller than the L value in the corresponding column, the null hypothesis of no difference cannot be rejected.

If we focus on the first five rows which show the comparison results of LAPSO with other methods, almost all the test values are negative and significant, which means LAPSO has better performance than other methods in most cases.

It is also interesting to see that there are never all negative or positive values in the same row. That means no method is always better or worse than the others on this selection of problems. Even for LAPSO, the convergence speed on function DeJong F2 is slow and the fitness value is higher than COPSO on the function Rastrigin. Nevertheless, the performances on other functions show that LAPSO has high convergence speed and good exploring capability to find the global minima.

5. Conclusion

In this paper, several modified PSO methods, DVPSO, COPSO and LAPSO, are proposed. LAPSO performs better behavior than other PSOs and CMA-ES, on aggregate, over a suite of test problems. Compared to other PSO methods, the readjustment of velocity with distribution vector $\vec{D}$ is not only on the numerical value but also on the direction. The application of distribution vector seems to accelerate the searching speed of PSO methods and strengthen the local searching ability according to the experimental results.

In addition, the ability to escape local minima is important on some functions. Permission to escape the leaders' control and search other places is important. The crossing over ability we introduce gives particles the possibility to stride over the local best, as well as the probability to find a global minimum.

LAPSO, which combines these two improvements, performs well on the test functions. Because of its fast searching speed and well behavior of global searching, LAPSO is expected to be applied on some real systems, such as molecular docking and parameter estimation. Both of docking and parameter estimation can be viewed as global search problems. In the docking problem, where for a given energy function, one has to find the most stable conformation of the receptor and ligand molecules. The searching space is extremely complex and contains thousands of local optima, and normally large amounts of ligand molecules are needed to be analyzed. In the parameter estimation problem, unknown parameters in the dynamic systems, mostly ordinary differential equation (ODE) systems, are needed to be obtained to fit the system to the experimental data. The high-dimensional searching space is also complicated for the probably exponential differences among these parameters values. Thus, for these situations, both efficiency and accuracy is highly required for a chosen optimizer. Some research work of applying LAPSO in these fields is under way.

References

[1] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the 1995 IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.

[2] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Proceeding of Congress on Evolutionary Computation, IEEE Service Center, Piscataway, NJ, (1999), pp. 1945–1950.

[3] A. Chatterjee, P. Siarry, Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization, Computers & Operations Research 33 (2006) 859–871.

[4] M. Clerc, The swarm and the queen: towards a deterministic and adaptive particle swarm optimization, in: Proceedings of the Congress of Evolutionary Computation, Washington, DC, (1999), pp. 1951–1957.

[5] J. Kennedy, Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance, in: Proceedings of Congress on Evolutionary Computation, 1999, pp. 1931–1938.

[6] J.J. Xu, Z.H. Xin, An extended particle swarm optimizer, in: 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)—Workshop 6, 2005, p. 193.

[7] J.E. Fieldsend, S. Singh, A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence, in: Proceedings of UK Workshop on Computational Intelligence (UKCI'02), 2002, pp. 37–44.

[8] S. He, Q.H. Wu, J.Y. Wen, J.R. Saunders, R.C. Paton, A particle swarm optimizer with passive congregation, Biosystems 78 (1–3) (2004) 135–147.

[9] A. Leontitsis, D. Kontogiorgos, J. Pagge, Repel the swarm to the optimum, Appl. Math. Comput. 173 (11) (2006) 265–272.

[10] P.J. Angeline, Evolutionary optimization versus particle swarm optimization: philosophy and performance difference, in: The 7th Annual Conference on Evolutionary Programming, San Diego, CA, (1998), pp. 601–610.

[11] M. Lovbjerg, T.K. Rasmussen, T. Krink, Hybrid particle swarm optimiser with breeding and subpopulations, in: Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, USA, (2001), pp. 469–476.

[12] L. Tsair-Fwu, C. Ming-Yuan, S. Chin-Shiuh, L. Hong-Jen, F. Fu-Min, Particle swarm optimization-based SVM for incipient fault classification of power transformers, in: 16th International Symposium on Methodologies for Intelligent Systems, ISMIS 2006, Bari, Italy, 2006.

[13] F. Van Den Bergh, A.P. Engelbrecht, Effects of swarm size on cooperative particle swarm optimizers, in: Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, USA, (2001), pp. 892–899.

[14] W.-Q. Lin, J.-H. Jiang, Q. Shen, H.-L. Wu, G.-L. Shen, R.-Q. Yu, Piecewise hypersphere modeling by particle swarm optimization in QSAR studies of bioactivities of chemical compounds, J. Chem. Inf. Model. 45 (3) (2005) 535–541.

[15] W. Cedeño, D. Agrafiotis, Particle swarms for drug design, in: Proceedings of IEEE Congress on Evolutionary Computation, IEEE CEC 2005, vol. 2, 2005, pp. 1218–1225.

[16] M.K. Gill, Y.H. Kaheil, A. Khalil, M. McKee, L. Bastidas, Multiobjective particle swarm optimization for parameter estimation in hydrology, Water Resources Research 42 (7) (2006) W07417.

[17] J.S. Heo, K.Y. Lee, R. Garduno-Ramirez, Multiobjective control of power plants using particle swarm optimization techniques, IEEE Transactions on Energy Conversion 21 (2) (2006) 552–561.

[18] S.K. Goudos, J.N. Sahalos, Microwave absorber optimal design using multi-objective particle swarm optimization, Microw. Opt. Technol. Lett. 48 (8) (2006) 1553–1558.

[19] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (1) (2002) 58–73.

[20] K. Yasuda, A. Ide, N. Iwasaki, Adaptive particle swarm optimization, in: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Washington, DC,USA, (2003), pp. 1554–1559.

[21] Y. Zheng, L. Ma, L. Zhang, J. Qian, On the convergence analysis and parameter selection in particle swarm optimization, in: Proceedings of the International Conference on Machine Learning and Cybernetics, Xi'an, PRC, (2003), pp. 1802–1807.

[22] F. Van Den Bergh, A.P. Engelbrecht, A study of particle swarm optimization particle trajectories, Inf. Sci. 176 (2006) 937–971.

[23] H.-P. Schwefel, Experimentelle Optimierung einer Zweiphasendüse Teil I. Technical Report No. 35 of the Project MHD–Staustrahlrohr 11.034/68, AEG Research Institute, Berlin, 1968.

[24] I. Rechenberg, Evolutionsstrategie-Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Frommann-Holzboog, Stuttgart (1973), second ed., 1994.

[25] H.-P. Schwefel, Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie, Interdisciplinary Systems Research, 26, Birkhäuser, Basel, 1977.

[26] H.-P. Schwefel, Numerical Optimization of Computer Models, Wiley, Chichester, 1981.

[27] H.-G. Beyer, H.-P. Schwefel, Evolution strategies—a comprehensive introduction, Natural Computing. 1 (1) (2002) 3–52.

[28] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation, in: Proceedings of the 1996 IEEE Conference on Evolutionary Computation (ICEC '96), 1996, pp. 312–317.

[29] L. Li, W.Q. Lin, J.H. Jiang, G.L. Shen, R.Q. Yu, QSAR analysis of substituted bis[(acridine-4-carboxamide) propyl] methylamines using optimized block-wise variable combination by particle swarm optimization for partial least squares modeling., Eur. J. Pharm. Sci. 25 (2005) 245–254.

[30] L. Temby, P. Vamplew, A. Berry, Accelerating real-valued genetic algorithms using mutation-with-momentum, in: The 18th Australian Joint Conference on Artificial Intelligence, Sydney, Australia, (2005), pp. 1108–1111.

[31] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Inform. Process. Lett. 85 (6) (2003) 317–325.

[32] R.C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Proceedings of Congress on Evolutionary Computation, San Diego, CA, (2000), pp. 84–88.

[33] A. Carlisle, G. Dozier, An off-the-shelf PSO, in: Proceedings of the 2001 Workshop on Particle Swarm Optimization, Indianapolis, IN, (2001), pp. 1–6.

[34] M. Jiang, Y.P. Luo, S.Y. Yang, Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm, Inform. Process. Lett. 102 (2007) 8–16.

[35] N. Hansen, S. Kern, Evaluating the CMA evolution strategy on multimodal test functions, in: 8th International Conference on Parallel Problem Solving from Nature PPSN VIII, Proceedings, Springer, Berlin, (2004), pp. 282–291.

[36] W.J. Conover, Practical Nonparametric Statistics, Wiley, 1999, pp. 288–299.