Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

A rapid particle swarm optimization algorithm with convergence analysis

Yanjun Li*, Yi Liu, Liangsheng Song*,Shan Liu****

**School of Information & Electrical Engineering, Zhejiang University City College, 310015 Hangzhou, China(e-mail: liyanjun@zucc.edu.cn;974822053@qq.com)*
***State Key Laboratory of Industrial Control, Zhejiang University, 310027, Hangzhou Zhejiang, China(e-mail:978355734@qq.com; sliu@iipc.zju.edu.cn)*

Abstract: Confronting with complex and troublesome optimization problems, Particle Swarm Optimization (PSO) algorithm maybe easy to be led to local optimum and suffer the unacceptable phenomenon as premature convergence. This paper proposes a new rapid PSO algorithm (RPSO), utilizing all particles' individual best positions found so far to update its velocity; providing a distinguishing weight according to the particles' different positions; and an adaptive learning factors turning strategy based on particle's fitness value. Then Jury Criterion is adopted to make convergence analysis of our proposed algorithm. Ultimately, the optimization performance of RPSO is simulated by searching the optimums of four common benchmark functions and training a RBF network for approximating a nonlinear system. The simulation results reveal the satisfactory efficacy of our proposed algorithm.

Keywords: PSO algorithm, RBF network, Convergence, Optimization, Adaptive.

1. INTRODUCTION

Nature-inspired heuristic optimization algorithms mainly consist of physically-inspired heuristics Xie et al. [2011], Ding et al. [2012] and biologically-inspired heuristics Kuo et al. [2013], Müller et al. [2002], Walton et al. [2011], Pan [2012], which all have gained widespread popularity in solving complex global optimization problems. In general, these heuristic optimization algorithms suffer somewhat limitations that may give rise to slow convergence or premature to their local optimum.

Since Particle Swarm Optimization (PSO) was first proposed by Kennedy in 1995, it has gradually gained widespread popularity in the research community for its simplistic implementation, reported satisfactory performance on various practical application problems, e.g. resource allocation Fan et al. [2013], training RBF neural network Fathi et al. [2013], and optimizing controller parameters Hu et al. [2011]. Kadirkamanathan et al. [2006] made a stability analysis of PSO using Lyapunov stability analysis and the concept of passive systems, and revealed that the selections of inertia weight w and learning factors $c1$, $c2$ have profound influence on the convergence performance of PSO. Then a lot of adaptive PSO algorithms have been proposed Hu et al. [2012], Xu [2013], Alireza [2011] to adjust the inertia weight and learning factors of PSO.

In these previous PSO algorithms, each particle updates its velocity according to the information of its current individual best position found so far. While in our view, this way is lack of activity to exploit more useful information to help individuals find better solutions. Therefore a new rapid PSO (RPSO) algorithm was proposed in this paper. We suggest that each particle takes the advantage of all particles' individual best positions found so far, which are better than its current position, to update its velocity. Then a distinguished weight based on its corresponsive object function value to the individual best position is given, to guarantee particles to quickly converge to acceptable position. And a adaptive learning factors turning strategy was also proposed according to the change of particle's fitness value. Furthermore, convergence analysis in view of Jury Criterion is presented here. Ultimately, the optimization performance of RPSO is simulated by searching the optimums of four common benchmark functions and training RBF network for approximating a nonlinear system, and the basic PSO (BPSO) is also used as comparison.

This work was supported by a grant from the National Natural Science Foundation of China (61104149) & the Zhejiang Province Natural Science Fund (Y1090339).

2.A RAPID SWARM OPTIMIZATION ALORITHM

In basic PSO algorithm, each particle is defined as a potential candidate solution to an optimization problem in d-dimensional space. The i-th particle's position and velocity in one dimensional space were given by:

$$V_i^{t+1} = w \cdot V_i^t + c_1 \cdot r_1 (X_{ibest}^k - X_i^t) + c_2 \cdot r_2 \cdot (X_{gbest} - X_i^t) \quad (1)$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \quad (2)$$

where w is the inertia weight, c_1 and c_2 are learning factors and typically equal to 2, r_1 and r_2 are uniformly distributed random numbers between 0 and 1. V_i^t and X_i^t are the i-th particle's velocity and position at t-th iteration. X_{ibest}^k

represents the best position found so far by the *k*-th particle, and X_{gbest} mean.

Obviously, the PSO algorithm possesses the characteristic of simplistic implementations. However, in face of complex and troublesome optimization problems, the PSO will be easy to be lost in local optimum and suffer the unacceptable phenomenon of premature convergence. In this paper, a novel rapid PSO (RPSO) algorithm is proposed to strengthen it's exploration ability.

First of all, the velocity updating formula of *i*-th particle velocity in RPSO is given by:

$$V_i^{t+1} = w \cdot V_i^t + \sum\nolimits_{k \in M_1} c_i^t \cdot \beta_k \cdot r_k \cdot (X_{ibest}^k - X_i^t) \qquad (3)$$

where $M_i = \{k \mid f(X_{lbest}^k) < f(X_i), \forall k \in pop\}$, pop is the set of all individuals, X_{lbest}^k is the previous best position of individual *k*, $f(X_{lbest}^k)$ is the objective function value at the individual best position of the *k*-th particle, β_k is regarded as the contribution of previous best position of the individual k and used to weight the position information adopted by RPSO. In RPSO algorithm, each particle utilizes all particles' individual best position found so far, which is better than its current position, to update its velocity. Thus each particle's velocity embodies more advantageous information, including the information of global best position and its individual previous best position. On the other hand, the contribution β_k is updated as follows:

$$\beta_k = e^{(f(X_{lbest}^k) - f(X_{worst}^{t-1}))/(f(X_{worst}^{t-1}) - f(X_{best}^{t-1}))}, \quad k \in M_i \qquad (4)$$

where $f(X_{worst}^{t-1})$ is the objective function value at the worst position of all particles at (*t-1*)-th iteration. $\mathrm{f}(X_{best}^{t-1})$ is the objective function value at the best position of all particles at (*t-1*)th iteration. C_i^t in (3) represents the learning factor of particle *i* at the *t*-th iteration.

Secondly, a new strategy for adaptive learning factors turning based on the change of particle's fitness value is proposed:

$$C_i^{t+1} = C_i^t \cdot (1 + \alpha \cdot (\tanh(\frac{\mathrm{f}(X_i^t) - \mathrm{f}(X_i^{t-1})}{\mathrm{f}(X_{\mathrm{best}0})}))) \qquad (5)$$

where α is a positive number that determines the change speed of learning factor. $f(X_{best0})$ indicates the objective function value of the initial global best position of all particles. $\tanh(\cdot)$ is hyperbolic tangent function. From this formula, if $f(X_i^t) < f(X_i^{t-1})$, the learning factor of particle *i* will diminish so that particle *i* can carry out local search; otherwise the learning factor of particle *i* will largen so that particle *i* can implement global search.

Thirdly, particle's position updating way is proposed as equation (6).

$$X_i^{t+1} = \begin{cases} X_i^t \cdot (1 + \rho \cdot (\xi - 0.5)) & if\, f(X_{gbest}) = f(X_i^t) \\ X_i^t + V_i^{t+1} & if\, f(X_{gbest}) < f(X_i^t) \end{cases} \qquad (6)$$

where $f(X_{gbest})$ is defined as the global best fitness over all particles. The position updating formula of particle *i* can be seen as local search if the $f(X_i^t)$ is equal to $f(X_{gbest})$, and $\rho = a \cdot (1 - t / \max gen)$, a is the initial amplitude of local search, $\max gen$ is the maximum value of generation. ξ is the random number between 0 and 1. Additionally, the particle's velocity and position are limited to their feasible boundaries to let all candidate solutions feasible.

The algorithm that uses the AEPSO to solve optimization problem is described as follow:

Step 1: Initialize population: both position X and velocity V, and set parameters, the number of population N, D, the maximum number of iterations G, the initial learning factors C_K^0, and inertial weight w, etc.

Step 2: Evaluate all individuals using corresponding objective function (fitness function), and record the corresponsive useful information X_{lbest}, X_{gbest}, etc.

Step 3: Calculate the contribution of all related particles' individual previous positions by Eq. (3) and adaptively adjust the learning factors by Eq. (4).

Step 4: Updating the velocity and position of particle by Eq. (5) and Eq. (6), and evaluate all individuals using corresponding objective function (fitness function).

Step 5: Updating the particles' related information X_{lbest}, X_{gbest}, etc., and judge the terminal condition (G is reached or $f(X_{worst}^t)$ is equal to $f(X_{best}^t)$ is met. If the terminal condition is satisfied, the optimization process will stop, otherwise, go back to Step 3.

3.CONVERGENCE ANALISIS

The convergence proof of RPSO is developed in this section. It reveals the essential convergence conditions under which RPSO is guaranteed to converge to a fixed equilibrium point. Since the RPSO algorithm is inherently stochastic (each particle's position is a random variable), the convergence for stochastic sequences is defined as follows.

Definition 3.1 (Stochastic sequence convergence Xie et al. [2011]) A stochastic sequence $\{X_i(t)\}, t = 0, 1, \ldots$ of scalars or vectors $X(t)$ converges to the constant value X^* $\forall i \in \{1, 2, \ldots N\}$ if the limit exists: $\lim_{t \to \infty} E[X_i(t)] = X^*$, that is, X^* is the limit of the excepted value of the stochastic sequence $\{X_i(t)\}$, where E is the excepted value operator, N is the population size (number of particles), $X_i(t)$ is the position of individual *i* at time *t*, and $E[X_i(t)]$ is the excepted value of $X_i(t)$.

Without loss of generality, the convergence of RPSO is investigated by observing one-dimensional RPSO model and the particle i is chosen arbitrarily. X_{best} is assumed to be a constant over some number of iterations when current iteration number t is big enough, so that λ in Eq. (6) approximates to be zero, and then all particles update their velocities and positions according to equation (5) and (2).

The position updating Eq. (2) can be rewritten by substituting Eq. (5) into (2) to obtain:

$$\begin{aligned} X_i^{t+1} &= X_i^t + V_i^{t+1} = X_i^t + w\cdot V_i^t + \sum_{k\varepsilon M_i} C_k^t\cdot\beta_k\cdot r_k\cdot\left(X_{lbest}^k - X_i^t\right) \\ &= X_i^t + w\cdot\left(X_i^t - X_i^{t-1}\right) + \sum_{k\in M_i} C_k^t\cdot\beta_k\cdot r_k\cdot\left(X_{lbest}^k - X_i^t\right) \\ &= \left(1+w-\theta_i\right)\cdot X_i^t - w\cdot X_i^{t-1} + \delta_i \qquad (7)\end{aligned}$$

With the following definitions

$$\theta_i = \sum\nolimits_{k\in M_i} C_k^t\cdot\beta_k\cdot r_k \qquad (8)$$

$$\delta_{\mathrm{i}} = \sum_{k\in M_i} C_k^t\cdot\beta_k\cdot r_{\mathrm{k}}\cdot X_{lbest}^k \qquad (9)$$

Assume that the fitness landscape is so smooth that the C_k^t and β_k keep stable in three generations. Eq. (7) may be written to the following non-homogenous recurrence relation:

$$X_i^{t+1} - (1+w-\theta_i)\cdot X_i^t + w\cdot X_i^{t-1} = \delta_i \qquad (10)$$

In order to obtain the convergence properties of RPSO, the corresponsive convergence of stochastic sequence $\{E\left[X_i(t)\right]\}$ may be analyzed. For the stochastic case, adding the expected value operator to Eq. (10) yields

$$E[X_{\mathrm{i}}^{\mathrm{t}+1}] - \left(1+w-\theta_{\mathrm{i}}^*\right)\cdot E[X_i^t] + w\cdot E[X_i^{t-1}] = \delta_{\mathrm{i}}^* \qquad (11)$$

where

$$\theta_{\mathrm{i}}^* = E\left[\theta_i\right] = \frac{1}{2}\sum_{k\in M_i} C_k^t\cdot\beta_k \qquad (12)$$

$$\delta_{\mathrm{i}}^* = E\left[\delta_{\mathrm{i}}\right] = \frac{1}{2}\sum_{k\in M_i} C_k^t\cdot\beta_k\cdot X_{lbest}^k \qquad (13)$$

By inspection, the characteristic equation of Eq. (11) is:

$$\lambda^2 - \left(1+w-\theta_i^*\right)\lambda + w = 0 \qquad (14)$$

As stable solutions of LTI system exist if and only if its eigenvalues all lie inside the unit circle in the complex λ - plane. The solutions of Eq. (14), in turn, will converge to a deterministic limit under the converging conditions. And, the necessary converging condition will be proved in Theorem 3.1. To better complete the proof of Theorem 3.1, Jury Criterion is cited as Lemma 3.1.

Lemma 3.1. (Jury Criterion in Jury [1974]) Considering the polynomial $P(\lambda) = \sum\nolimits_{j=0}^{N} P_j\lambda^{N-j}$, with P_0=1 , it has no complex root outside the unity circle if and only if all the following conditions are satisfied:

1） $P(\lambda=1)$>0,

2） $(-1)^N P(\lambda=-1)$>0,

3） $|P_N| < P_0$,

4） Defining the following inductive relation:

$$\begin{cases} p(k+1,j) \triangleq p(k,0)\cdot p(k,j) - p(k,N-k)\cdot p(k,N-k-j) \\ \qquad \forall k\in[0,N-2] \quad \text{and} \quad \forall j\in[0,N-k] \end{cases}$$

with $p(0,j) \triangleq p_j$, $\forall j\in[0,N+1]$, last assumptions are: $\left|p(k,0)\right| > \left|p(k,N-k)\right|, \forall k\in[1,N-2]$

Thus the necessary converging condition is demonstrated in Theorem3.1.

Theorem 3.1. If and only if $|w|<1$ and $0<\theta_i^*<2(1+w)$, then $X_i(t)$ converges to X_{gbest} .

Proof. According to Lemma 3.1 and Eq. (14), the convergence conditions of Eq. (11) are:

$$\begin{cases} p(\lambda=1) = 1-\left(1+w-\theta_i^*\right)+w = \theta_i^* > 0 \\ p(\lambda=-1) = 1+\left(1+w-\theta_i^*\right)+w = 2\left(1+w\right)-\theta_i^* > 0 \\ |w|<1 \end{cases} \qquad (15)$$

Thus, the stable conditions of Eq. (11) are

$$|w|<1,\ 0<\theta_i^*<2(1+w) \qquad (16)$$

Thus, $\{E[X_i^t]\}$ will converge to the corresponding convergent position X_i^* if Eq. (16) is met. Next, to take the limit on both sides of Eq. (15), note that $\lim_{t\to\infty}\{E[X_i^t]\} = X_i^*$, $i=1,\ldots,N_p$, with the result:

$$X_i^* - \left(1+w-\theta_i^*\right)\cdot X_i^* + w\cdot X_i^* = \delta_i^* \qquad (17)$$

To simplify Eq. (17) as

$$\theta_i^*\cdot X_i^* = \delta_i^* \qquad (18)$$

Substitute Eq. (8) and Eq. (9) into Eq.(18), one has

$$\sum\nolimits_{k\in M_i} C_k^t\cdot\beta_k\cdot X_i^* = \sum\nolimits_{k\in M_i} C_k^t\cdot\beta_k\cdot X_{lbest}^k \qquad (19)$$

Which may be written:

$$X_i^* = \frac{\sum\nolimits_{k\in M_i} X_{lbest}^k}{\mathrm{NM}} \qquad (20)$$

According to the definition 3.1, a limit of sequence $\{X_i^t\}$ exists and converges to X_i^* if Eq. (15) is met. Thus, to take the limit on both sides of Eq. (5) and Eq. (6), with the result:

$$\begin{aligned} &\sum\nolimits_{k\in M_i} C_k^t\cdot\beta_k\cdot r_k\cdot(X_{lbest}^k - X_i^*) = 0 \\ &\Rightarrow \sum\nolimits_{k\in M_i} r_k\cdot(X_{lbest}^k - X_i^*) = 0 \end{aligned} \qquad (21)$$

Considering that r_k is a random number between 0 and 1, thus if and only if $X_i^* = X_{lbest}^1 = \cdots = X_{lbest}^{NM} = X_{gbest}$, $k \in M_i$, Eq. (18) is satisfied. So $X_i(t)$ will converge to X_{gbest} .

4.SIMULATIONS AND RESULTS

Some simulation experiments were conducted as employing four commonly used benchmark functions and training RBF networks to investigate the optimization performance of the RPSO algorithm proposed in this paper.

4.1 four benchmark functions

Benchmark functions are widely applied on inspecting the optimization performance of various algorithms due to their characteristics of diversity of decision space topologies. Four common benchmark functions shown in Table 1 are used in this paper to examine the usefulness of RPSO, and the basic PSO (BPSO) algorithm is chosen as comparisons.

To keep the fairness of performance comparison between RPSO and BPSO, their initial swarm population size N , maximum number of iterations G and feasible searching space are all in the beginning stage. In detail, N =30, is G=2000, the inertial weight w=0.75 in BPSO, the inertial weight and learning factors of RPSO are $w = C_K^0$ =0.9, and uniformly distributed random numbers on $\left[\frac{5}{N}, \frac{25}{N}\right]$. For each experimental setting, 20 independent runs of BPSO and RPSO are performed. The implement results on the functions of $f_1 \sim f_4$ with dimension 10, 20, 30 are in terms of the best (minimum) fitness value (B), worst (maximum) fitness value (W) and average fitness value (A) over 20 runs as shown in Table 2.

Table 1 Four common benchmark functions

	Functions
Tablet	$f_1(X) = 10^6 X_1^2 + \sum_{i=2}^{n} X_i^2$, $X_i \in [-50,50]^d$
Rosenbrock	$f_2(X) = \sum_{i=1}^{n-1}(100(X_{i+1} - X_i^2)^2 + (X_i - 1)^2)$, $X_i \in [-50,50]^d$
Griewank	$f_3(X) = \frac{1}{4000}\sum_{i=1}^{n} X_i^2 - \prod_{i=1}^{n}\cos\left(\frac{X_i}{\sqrt{i}}\right) + 1$, $X_i \in [-50,50]^d$
Rastrigin	$f_4(X) = \sum_{i=1}^{n}(X_i^2 \text{ -} 10 \cdot \cos(2\pi X_i) + 10)$, $X_i \in [-5,5]^d$

Table 2 Results of BPSO (no.1) and RPSO (no.2) on four benchmark functions with dimension=10, 20, 30, respectively.

*f*1~*f*4		Tablet	Rosenbrock	Griewank	Rastrigin
B(10)	1	6.62e-08	0.2842	0.0295	4.9748
	2	4.59e-68	2.5874	0	0.9952
A(10)	1	5.79e-04	5.4799	0.0953	12.5365
	2	1.07e-55	4.6701	0.0053	3.3358
W(10)	1	0.0037	9.6045	0.1944	19.8991
	2	1.13e-54	7.5223	0.0541	6.9647
B(20)	1	0.1167	21.3663	0.01	16.3278
	2	7.52e-34	14.9357	0	2.078
A(20)	1	0.6003	193. 289	0.0465	25.9782
	2	7.85e-31	16.3144	0.0058	7.1112
W(20)	1	1.0664	1.91e+03	0.0915	38.0477
	2	1.31e-29	19.3644	0.0473	14.0054
B(30)	1	0.9964	197.8931	0.0214	33.9406
	2	3.24e-24	23.7176	0	6.4494
A(30)	1	3.0831	723.6394	0.1046	53.8696
	2	4.34e-21	29.0332	0.0284	11.6493
W(30)	1	8.2219	1.86e+03	0.181	93.2765
	2	4.64e-20	84.2917	0.2448	15.907

4.2 parameters identification for a RBF network based nonlinear system

Considering a single-input two outputs nonlinear system in Du [2010]:

$$\begin{aligned} y_1(k) &= 0.5 \times y_1(k\text{-}1) + u(k\text{-}1) + 0.4 \times \tanh\tanh(u(k\text{-}2)) \\ &\quad + 0.1 \times \sin\sin(\pi \times y_2(k\text{-}2)) \times y_2(k\text{-}1) \\ y_2(k) &= 0.3 \cdot y_2(k-1) + 0.1 \cdot y_2(k-2) \cdot y_1(k-1) \\ &\quad + 0.4 \cdot \exp(-u^2(k-1)) \cdot y_1(k-2) \end{aligned} \tag{22}$$

where the system input $u(k)$ is uniformly distributed within [-0.5, 0.5]. $y_1(k)$ and $y_2(k)$ are the system outputs.

RBF network consists of three separate layers, named input layer, hidden layer and output layer respectively, is considered here with a common Gaussian function:

$$y_j = \sum\nolimits_{i=1}^{M} w_{ji} \cdot \varphi(x, c_i)$$

and $$\varphi(x, c_i) = \exp(-\| x - c_i \| / \sigma_i^2) \tag{23}$$

where M is the number of the hidden nodes, w_{ji} is the weight between *j*-th output and *i*-th input, c_i and σ_i are the center and width of *i*-th unit in the hidden layer. $y_1(k-1)$, $y_1(k-2)$, $y_2(k-1)$, $y_2(k-2)$, $u(k-1)$, $u(k-2)$ are regarded as the inputs of RBF network, whose initial conditions are all set as zero, and 200 sample data are generated for training.

Relative parameters are set as: Number of hidden nodes=10; Number of inputs/outputs=6/2; Number of centers =10; Number of weights =20.

A least-squares error criterion J_e is utilized to evaluate the efficacy of algorithms.

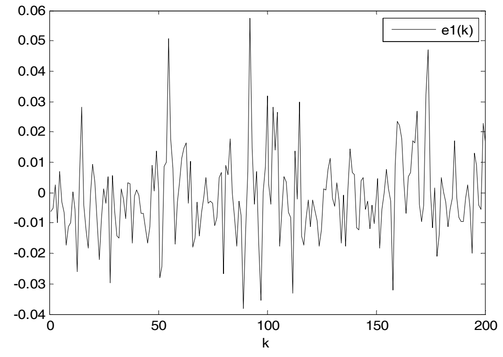
$$J_e = \frac{\sum_{i=1}^{N}\sum_{k=1}^{L}(y_d(k)-y^*(k))^2}{N} \tag{24}$$

where y_d is the desired output vector, and y^* is the output of RBF neural network, L the length of y_d and N the number of system output.

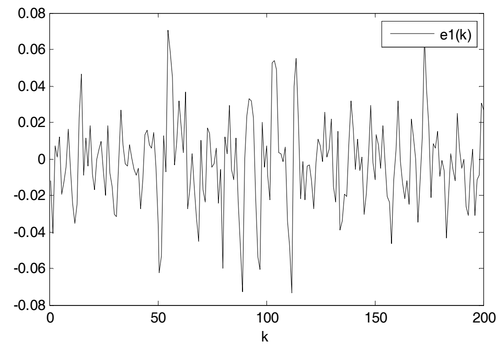
RPSO and basic PSO (BPSO) are used to optimize (c_i, σ_i, w_{ji}) in (23) and for comparison. Related parameters are setting as follows: N =500, G =2000, w = 0.75, the learning factors of RPSO=$\frac{20}{N}$, and $X_{\max} = 20$. Furthermore, 20 independent runs of RPSO and BPSO are performed. The simulation results over 20 runs are compared in Table 3. It can be easily seen that the values of RPSO are all smaller than BPSO's revealing the prior optimization capability of RPSO. And Fig. 1 shows the training errors for one output of RBF constructed by RPSO and BPSO respectively, averaged for 20 runs. The simulation results show the better performance and acceptable robustness of RPSO based RBF network.

Table 3 Performance comparisons

Criteria / Algorithms	minimum value (MIN)	average value (AVG)	maximum value (MAX)
J_e by RPSO	0.0648	0.1673	0.4361
J_e by BPSO	0.0684	0.5131	2.5185



(a) Training error of output $y_1(k)$ by RPSO



(b) Training error of output $y_1(k)$ by BPSO

Fig.1. $e_1(k)$ of RBF network averaged for 20 runs

5.CONCLUSIONS

This paper proposed a rapid PSO (RPSO) algorithm aiming to improve PSO's optimization performance. A novel velocity updating way is proposed taking advantage of all particles' individual best positions found so far; secondly, a distinguishing weight is given based on its corresponsive object function value to guarantee particles to quickly converge to a acceptable position; thirdly, a new parameter turning strategy is presented for learning factor based on the change of particle's fitness value to enhance the performance of the RPSO. Then, a Jury Criterion has been applied to obtain the convergence conditions of RPSO with simplistic and efficient. Finally, the optimization performance of RPSO is assessed by searching the optimums of four common benchmark functions and training RBF network for approximating a nonlinear system, and the basic PSO is chosen as comparison. The simulation results have reflected the prior performance of RPSO.

REFERENCES

Xie L., Zeng J., and Formato R.A.. (2011). Convergence analysis and performance of the extended artificial physics optimization algorithm. *Applied Mathematics and Computation* , vol. 218 , no. 8 , pp. 4000–4011.

Ding D., Qi D., and Luo X., *et al.* (2012). Convergence analysis and performance of an extended central force optimization algorithm. *Applied Mathematics and Computation*, vol. 219, pp. 2246–2259.

Kuo H.C., and Lin C. H.. (2013). A Directed Genetic Algorithm for global optimization. *Applied Mathematics and Computation* , vol. 219 , pp. 7348–7364.

Müller S.D., Marchetto J., Airaghi S., and Koumoutsakos P.. (2002). Optimization Based on Bacterial Chemotaxis, *IEEE Trans. on Evol. Comput.,* vol. 6 , no. 1 , pp. 16-29,.

Walton S., Hassan Morgan O., K., and Brown M.R.. (2011).Modified cuckoo search: A new gradient free optimization algorithm. *Chaos, Solitons & Fractals,* vol. 44 , pp. 710–718.

Pan W.T.. (2012). A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example. *Knowledge-Based Systems,* vol. 26 , pp. 69–74.

Fan K., You W., and Li Y.. (2013). An effective modified binary particle swarm optimization algorithm for multi-objective resource allocation problem (MORAP). *Applied Mathematics and Computation,* vol. 221 , pp. 257–267.

Fathi V., and Montazer G.A.. (2013). An improvement in RBF learning algorithm based on PSO for real time applications. *Neurocomputing,* vol. 111 , pp. 169–176.

Hu D., Sarosh A., and Dong Y.. (2011). An improved particle swarm optimizer for parametric optimization of flexible satellite controller. *Applied Mathematics and Computation* , vol. 217 , no. 21 , pp. 8512–8521.

Kadirkamanathan V., Selvarajah K., and Fleming P.J.. (2006). Stability Analysis of the Particle Dynamics in Particle Swarm Optimizer. *IEEE Trans. on Evol. Comput.,* vol. 10, no.3, pp. 245-255,

Hu, M., Wu, T., Weir, J.D.. (2012). An Adaptive Particle Swarm Optimization with Multiple Adaptive Method. *IEEE Trans. on Evol. Comput.,* vol. 17 , pp. 1-15,

Xu G.. (2013). An adaptive parameter tuning of particle swarm optimization algorithm. *Applied Mathematics and Computation* , vol. 219 , no. 9 , pp. 4560–4569.

Alireza A.. (2011). PSO with Adaptive Mutation and Inertia Weight and Its Application in Parameter Estimation of Dynamic Systems. *Acta Automatica Sinica,* vol. 37 , no. 5 , pp. 541-549.

Jury E. I.. (1974). *Inners and Stability of Dynamic Systems.* John Wiley & Sons ,New York.

Du D., Li K., and Fei M.. (2010). A fast multi-output RBF neural network construction method. *Neurocomputing,* vol. 73 , no. 10-12 , pp. 2196–2202.