Local Optima Avoidable Particle Swarm Optimization

S.M.A. Salehizadeh, P. Yadmellat, *Graduate Student Member, IEEE*, M.B. Menhaj, *Member, IEEE*

Abstract— **This paper proposes a Local Optima Avoidable Particle Swarm Optimization (LOAPSO) which remarkably outperforms the standard PSO in the sense that it can avoid entrapment in local optimum. Three benchmark functions are used to validate the proposed algorithm and compare its performance with that of the other algorithms known as hybrid PSOs and six functions reported in SIS2005 are used to better verification of the proposed algorithm. Numerical results indicate that LOAPSO is considerably competitive due to its ability to avoid being trapped in local optima and to find the functions' global optimum as well as better convergence performance.**

I. Introduction

PSO is a population-based optimization technique proposed firstly by Kennedy and Eberhart in [1] for the unconstrained minimization problem. Particle swarm optimization (PSO) is inspired from studies of various animal groups and has been proven to be a powerful competitor to other evolutionary algorithms such as genetic algorithms [1], [2].

PSO system combines local search method (through self experience) with global search methods (through neighboring experience), attempting to balance exploration and exploitation. PSO is widely used to solve nonlinear and multiobjective problems such as optimization of weights of neural networks (NN), electrical utility, computer games, and mobile robot path planning, etc.[3–8].

The particle swarm optimization is based on changing the velocity and position of each particle toward its *pbest* and *gbest* locations according to the equations (1) and (2), respectively, at each time step:

$$v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1 \left(p_{id} - x_{id}(t)\right) + c_2 r_2 \left(p_{gd} - x_{id}(t)\right) \tag{1}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{2}$$

where ω is the inertia weight, c_1 and c_2 are acceleration constants, r_1 and r_2 are a random variables in the range [0, 1], $x_{id} \in [x_{min}, x_{max}]$ and $v_{id} \in [-V_{max}, V_{max}]$, where x_{min} and x_{max} denote minimum and maximum of particle position, respectively. And V_{max} specify maximum of velocity. Further details about the algorithm are given in [1], [9-11].

S.M.A. Salehizadeh and P. Yadmellat are master students in Amirkabir University of Technology, Tehran, Iran, smasalehizadeh@gmail.com, p.yadmellat@aut.ac.ir (corresponding author). http://eew.aut.ac.ir/pyadmellat.

M.B. Menhaj is professor of Electrical Engineering Department, Amirkabir University of Technology, Tehran, Iran, tmenhaj@ieee.org.

A series of works has been done on the analysis and development of PSO since it was introduced in 1995. According to [12], although PSO finds good solutions much faster than other evolutionary algorithms, it usually cannot improve the quality of solutions as the number of iterations increases and it suffers from premature convergence when strongly multi-modal problems are being optimized. One of the main reasons is that all particles converge to a single point as the speed of the particles is decreased with time, thus forcing them to converge to the global best point found so far, which is not guaranteed even to be a local minimum. Modifying a single parameter may result in a large effect [13]. For example, increasing the value of the inertia weight w will increase the speed of the particles resulting in more exploration and less exploitation. Many researchers tried to improve the performance by tuning the parameters of standard PSO [14], or by adding new parameters such as mutation [15].

As a stochastic search algorithm, PSO is prone to lack global search ability at the end of a run. PSO may fail to find the required optima in case when the problem to be solved is too complicated and complex.

Although effective, PSO sometimes suffers from premature convergence on problems with many local minima. Convergence is in general a desirable property, allowing the swarm to search regions near the global minimum at increasing levels of detail as time progresses. Unfortunately, in the context of many local minima, the convergence property may cause a swarm to become trapped in one of them and fail to explore more promising neighboring minima [16]. Designers of optimization algorithms therefore face a fundamental trade off: search the current local minimum in detail through quick convergence, or consume resources exploring other areas of the domain [17].

To enhance the exploration capability and to avoid being trapped into local optimum, a mere strategy is to increase search space diversity. One such approach, the Spatial Extension PSO (SEPSO), involves endowing each particle with a radius, then causing particles to bounce off of one another [18]. A related approach, called Attractive-Repulsive PSO (ARPSO), measures the global diversity of the swarm, triggering modes of global attraction or repulsion when it crosses predefined thresholds [17]. Though effective when well-tuned, finding good function- specific tuning parameters for these methods is non-trivial. In [16] authors proposed an adaptation methodology which is used to improve robustness of parameters and performance on multi- modal functions by giving each particle an individual, adaptable radius. They believe that SEPSO behavior can be improved by adapting its radius and bounce parameters in response to collisions.

In [19-21], authors suggest to incorporate chaotic dynamics into the standard PSO to deal with the problem of entrapment into local optima. Chaos is a kind of characteristic of non-linear systems, which is a bounded dynamic behavior but not stable in the sense of Lyapunov that exhibits sensitive dependence on initial conditions and includes infinite unstable limit cycles. Although, it is stochastic apparently, it occurs in a deterministic non-linear system under deterministic conditions.

Some approaches try to hybrid algorithms with the characteristic of a high exploration performance such as GA, DE, etc [22-24].

Authors in [22] in order to avoid the problem of trapping into local optimum firstly proposed a hybrid adaptive mutation PSO with GA (HAMPSO) which is tried to coordinate the PSO parameters to get high convergence speed, and then introducing the mechanism of mutation of GA, when the particle trapped to optimization result for a long time. In [23], a new algorithm known as breeding swarm has been proposed. The main concern in this approach is to incorporate PSO and GA simultaneously in order to obtain a new robust algorithm for training recurrent neural networks. The authors acclaimed that by applying this training algorithm to 80 different recurrent neural networks it can exclusively outperform the standard particle swarm and genetic algorithms distinctively. The combination of PSO with other evolutionary algorithms has introduced new parameters and increased the computational effort [25], with varying degree of success for different problems. In [26] genetic programming is used as a tool to derive PSO variants.

Ratnaweera et al. [15] stated that the lack of population diversity in PSO algorithms is understood to be a factor in their convergence on local optima. Therefore, the addition of a mutation operator to PSO should enhance its global search capacity and thus improve its performance. There are mainly two types of mutation operators: one type is based on particle position [11], [27-30] and the other type is based on particle velocity [15], [31-33]. The former method is by far the most common technique found in the literature. However, the work of the latter one is very few [24]. In [24], several mutation operators that are based on the global best particle are investigated for PSO. An adaptive mutation operator is also designed for PSO.

In [34] authors proposed a gregarious approach (G-PSO), where the population is attracted by the global best position and each particle is re-initialized with a random velocity if it is stuck close to the global best position. In this manner, the algorithm proceeds by aggressively and greedily scouting the local minima whereas Basic-PSO proceeds by trying to avoid them. Therefore a re-initialization mechanism is needed to avoid the premature convergence of the swarm.

In this paper, we will propose a modified PSO algorithm based on the traditional PSO structure, in which a major modification is introduced. Standard PSO is similar to the other evolutionary algorithms in that the system is initialized with a population of random solutions. However, each potential solution is also assigned a randomized velocity, and the potential solutions, call *particles*, corresponding to individuals. Each particle in PSO flies in the D-dimensional problem space with a velocity which is dynamically adjusted according to the best flying experiences of its own and its colleagues. However, if the best experience of the whole swarm was a local optimum, the solution would not vary as the algorithm proceeds, and the problem of entrapment in local minimum arises. In order to overcome this problem, authors in this paper introduce a strategy based on avoiding local optimum and increasing in diversity. The fact behind this strategy is to divide the population into two populations by defining a new variable γ which is called as "avoidance rate" which is chosen iteratively in $[0,1]$.

$$\gamma(t)=\begin{cases} rand(1), & t<0.75t_{\max} \\ 1, & OW. \end{cases}, \tag{3}$$

where $t_{\max}$ denotes maximum of iteration.

Let N be the number of particles in swarm, then $\gamma\times N$ denotes the number of particles must perform the standard PSO velocity update

$$\begin{aligned} v_{id}(t+1)=\omega v_{id}(t)+c_1r_1(p_{id}-x_{id}(t)) \\ +c_2r_2(P_{gd}-x_{id}(t)), \end{aligned}, \tag{4}$$

$x_{id}\in\{1,...,n\}, Round\left(\gamma(t)N\right)$, and the remainders implement the following velocity update formula,

$$\begin{aligned} v_{iL}(t+1)=\omega v_{iL}(t)-L(t)[r_1(p_{id}(t)-x_{iL}(t)) \\ +r_2(p_{gd}(t)-x_{iL}(t))], \end{aligned}, \tag{5}$$

where $x_{iL}\in\{n+1,...,N\}$, L(t) is "avoidance coefficient", which is chosen iteratively according to the following equation

$$L(t)=2\left(1-\frac{t}{t_{\max}}\right). \tag{6}$$

Finally, all the particles in both above populations must update their position according to (2).

Indeed, here we have two groups of particles that one tries to move toward its best previous experience and its social best experiences according to (3), and particles in the other, decide to move away from the best experience of first group, in such a way that it increases the diversity while avoiding entrapment into local optimums by satisfying (4).

II. LOCAL OPTIMA AVOIDABLE PSO ALGORITHM (LOAPSO)

As mentioned above, the proposed LOAPSO algorithm can be expressed as follows

Step1: Initialize particles position and velocity

Step2: Defining the avoidance rate randomly in [0, 1]

Step3: Population division in two sub group of particles:
$$\begin{cases} Group1:\gamma\times N \\ Group2:(1-\gamma)\times N \end{cases}$$

Step3-1: Velocity update for particles in group 1 according to (4)

Step3-2: Velocity update for particles in group 2 according to (5) in order to avoid entrapment in local optima while increasing diversity

Step4: Position update for both groups particles according to (2)

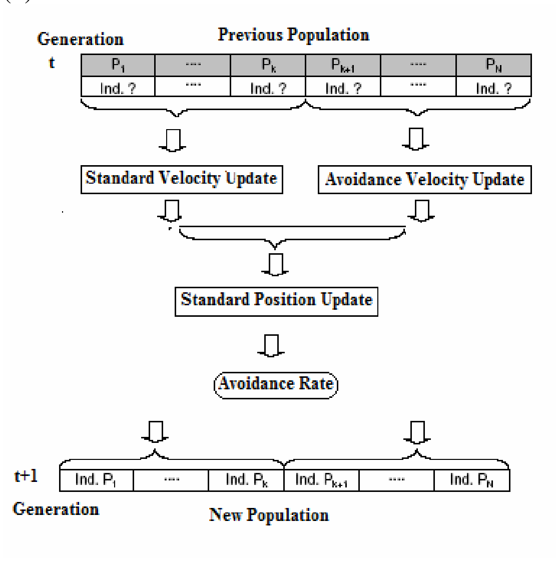

Figure1. Local optima avoidable PSO (LOAPSO) procedure.

TABLE 1
BENCHMARK FUNCTIONS

Rosenbrock	$f_1 = \sum_{i=1}^{D-1}\left[100\left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2\right],$ $x_i \in [-30, 30]$
Griewank	$f_2 = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1,$ $x_i \in [-600, 600]$

TABLE 2
ASYMMETRIC INITIALIZATION RANGES

Function	Asymmetric Initialization Range
f_1	$(-15, 30)^n$
f_2	$(-300, 600)^n$

TABLE 3
PSO PARAMETERS

Specification	**PSOs**
c_1, c_2	2.05
ω	0.5
$\gamma(t)$	$\begin{cases} rand(1), & t < 0.75t_{\max} \\ 1, & OW. \end{cases}$
L	$\begin{cases} 4 \cdot rand(1), & t < 0.75t_{\max} \\ -4 & O.W. \end{cases}$

III. SIMULATION RESULTS

We use two benchmark functions reported in [1, 9, 11 and 35] (see Table1) to compare the results with standard PSO and some previous modified PSOs and then Ackley function in order to investigate the effect of dimension on searching quality of LOAPSO, finally the algorithm applied to six composite benchmark functions reported in [36] which are used in SIS Competition in 2005 for better verification of the proposed algorithm performance.

TABLE 4
$x_{\min}$ AND $x_{\max}$ VALUES

Function	$x_{\min}$	$x_{\max}$
f_1	-100	100
f_2	-600	600

F1: ROSENBROCK FUNCTION			
D	$t_{\max}$	Algorithm	Average
10	1000	SPSO	40.2940
		LIWPSO	67.9036
		AIWPSO	59.6966
		MPSO	10.6114
		CPSO	7.7192
		TCPSO	**4.1497**
		LOAPSO	4. 9142
20	1500	SPSO	53.0865
		LIWPSO	133.9739
		AIWPSO	123.2690
		MPSO	16.0597
		CPSO	17.3572
		TCPSO	14.8649
		LOAPSO	**13.8531**
30	2000	SPSO	86.1312
		LIWPSO	197.2661
		AIWPSO	170.9275
		MPSO	95.7392
		CPSO	136.3357
		TCPSO	79.8258
		LOAPSO	**28.8621**

F2: GRIEWANK FUNCTION			
D	$t_{\max}$	Algorithm	Average
10	1000	SPSO	0.0923
		LIWPSO	0.1040
		AIWPSO	0.1099
		MPSO	0.0517
		CPSO	0.0639
		TCPSO	0.0221
		LOAPSO	**0.00 ±1.9763e-323**
20	1500	SPSO	0.0295
		LIWPSO	0.0296
		AIWPSO	0.0309
		MPSO	0.0027
		CPSO	0.0074
		TCPSO	3.3806e-7
		LOAPSO	**0.00 ±1.9763e-323**
30	2000	SPSO	0.0153
		LIWPSO	0.0140
		AIWPSO	0.0143
		MPSO	3.4099e-5
		CPSO	4.2619e-4
		TCPSO	1.6704e-8
		SCPSO	**0.00 ±1.9763e-323**

* 0.00±1.9763e-323 is the round of error in MATLAB

Like in [35], for first two functions, corresponding to the dimension 10, 20 and 30, the number of generation is set to1000, 1500 and 2000, respectively. For all the PSO runs, the general parameters of PSO are shown in Table 2 and Table 3. $v_{\max} = x_{\max}$ and $x_{\max}, x_{\min}$ are listed in Table4.

TABLE 5
ALGORITHMS' PARAMETERS

FES	50,000
PSO [40]	population size: 20
CPSO [42]	population size: 10 Dimension: 10
CLPSO [43]	Population size: 10, $P_c = 0.05$
LOAPSO	population size: 20
CMA-ES [44]	Coordist: 5
G3 [45]	Population size: 150 $\mu : 3$ $\lambda : 2$ Replace number: 1
DE [46]	Population size: 50 Strategy: rand/1/exp

Simulation results have been shown that the proposed LOAPSO remarkably has a better performance compared to the other methods (SPSO [37], LIWPSO [38], AIWPSO [39], MPSO [11] CPSO [19] and TCPSO [21]).

Furthermore, to investigate the effect of dimension on searching quality of LOAPSO, Similar to [40] Ackley function [41] is chosen for test

$$f_3(X) = -20\exp\left(-0.2\sqrt{1/n\sum_{i=1}^{n} x_i^2}\right)$$
$$-e^{1/n\sum_{i=1}^{n}\cos 2\pi x_i} + 20 + e, \qquad x \in [-35, 35]$$

Ackley function has thousands of local minima in the region and is challenging benchmark to be optimized, while the global minimum is located in $x_i = 0, i = 1,...,n$. Figure 2 illustrates the effect of dimension on LOAPSO in comparison of GA, PSO and Chaotic PSO (CPSO). The number of function evolution is fixed as 4×10^5 (20,000 generation for 20 particles) and dimension varies by 10 steps.

Finally in order to verify the proposed algorithm performance rigidly, we apply the algorithm to six benchmark function reported in IEEE swarm intelligence symposium in 2005 [36]. The algorithms' parameters and simulation results are shown in Table 5, 6 and 7. Resulting LOAPSO mean objective values for six composite functions are shown in Fig. 3.

Simulation results have been shown that the proposed LOAPSO is remarkably simple and less time consuming compared to the other methods. As mentioned in [47] this feature is extremely important when the PSO is applied to real-time online problems where the time is critical and the solution evaluation consumes a significant amount of computation time. Furthermore, it can be seen from the tables that the performance of the proposed PSO does not degrade significantly as the problem dimension scales up.

TABLE 6
SIMULATION RESULTS OF LOCAL OPTIMA AVOIDABLE PSO (LOAPSO) FOR SIX SIS-05'S BENCHMARK FUNCTIONS.

FES	#CF	1	2	3	4	5	6
5×10^4	1st (Best)	0.1745	1.4550	0.0238	0.1301	0.0994	0.5413
	$\frac{10^{th}+11^{th}}{2}$ (Median)	122.6258	159.5705	116.1540	149.5923	159.1874	145.5215
	20th (Worst)	241.6300	319.8349	300.0891	263.3976	263.4093	453.7094
	Mean	127.3714	152.1363	113.5342	138.6563	164.5203	155.6960
	Std	49.3078	69.7621	99.1415	68.3947	69.7170	101.4565

TABLE 7
COMPARISON WITH THE RESULTS MENTIONED IN [36]

	PSO	Cooperative PSO (CPSO)	Comprehensive Learning PSO (CLPSO)	Local Optima Avoidable PSO (LOAPSO)
CF1: Mean	100.00	156.26	**5.7348e-008**	127.3714
Std.	81.650	134.27	1.0352e-007	49.3078
CF2: Mean	155.91	242.29	**19.157**	152.1363
Std.	131.76	148.95	14.748	69.7621
CF3: Mean	172.03	362.64	132.81	**113.5342**
Std.	32.869	196.31	20.027	99.1415
CF4: Mean	314.30	522.37	322.32	**138.6563**
Std.	20.066	122.09	27.461	68.3947
CF5: Mean	83.450	255.56	**5.3705**	164.5203
Std.	101.11	175.63	2.6056	69.7170
CF6: Mean	861.42	853.14	501.16	**155.6960**
Std.	125.81	127.98	7.7800e-1	101.4565

TABLE 7 (CONT.)
COMPARISON WITH THE RESULTS MENTIONED IN [36]

	CMA-ES	G3-PCX	DE
CF1: Mean Std.	100.00 188.56	60.00 69.921	6.7459e-002 1.1057e-001
CF2: Mean Std.	161.99 151.00	92.69 99.06	28.75 8.62
CF3: Mean Std.	214.06 74.18	319.80 125.19	144.41 19.40
CF4: Mean Std.	616.40 671.92	429.96 142.49	324.86 14.78
CF5: Mean Std.	358.53 168.26	26.02 41.57	10.78 2.60
CF6: Mean Std.	900.26 8.3186e-002	772.08 189.39	490.94 39.46

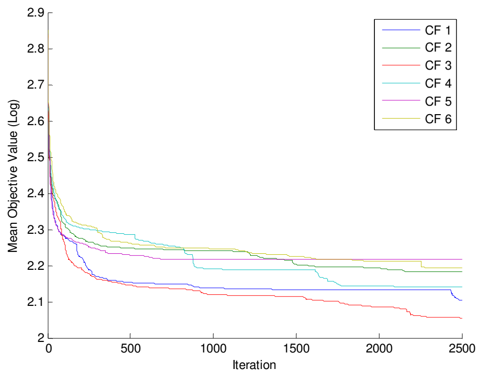

Figure 3. LOAPSO mean objective value.

IV. CONCLUSION

In this study, novel local optima avoidable PSO is proposed which remarkably outperforms the standard PSO in the sense that it can avoid entrapment in local optimum. Nine benchmark functions (i.e. three typical benchmarks and six composite ones) have been used for numerical simulations. The averaged results compared with the other methods show that the proposed LOAPSO algorithm has a remarkable performance regarding to convergence and avoiding to entrapment into local optimum and thus finding global optimum.

REFERENCES

[1] J. Kennedy, and R. Eberhart, “Particle Swarm Optimization”, IEEE International Conference on Neural Networks , pp. 1942-1948, 1995.

[2] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science (MHS’95), Nagoya, Japan, 1995, pp. 39–43.

[3] S. M. A. Salehizadeh, P.Yadmellat, A. R. Pourshoghi, M. J. Aein, “A hybrid algorithm for training recurtrent fuzzy neural network”, In Proc: Artificial Intelligence and Pattern Recognition, 2008, pp.181-186.

[4] C. F. Juang, “A hybrid of genetic algorithm and particle swarm optimization for recurrent network design,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 2, pp. 997–1006, Apr. 2004.

[5] H. Yoshida, K. Kawata, Y. Fukuyama, and Y. Nakanishi, “A particle swarm optimization for reactive power and voltage control considering voltage stability,” in *Proc. Int. Conf. Intell. Syst. Appl. Power Syst.*,Riode Janeiro, Brazil, 1999, pp. 117–121.

[6] M. A. Abido, “Particle swarm optimization for multimachine power sys- tem stabilizer design,” in *Proc. Power Eng. Soc. Summer Meeting*, 2001, pp. 1346–1351.

[7] L. Messerschmidt and A. P. Engelbrecht, “Learning to play games using a PSO-based competitive learning approach,” *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 280–288, Jun. 2004.

[8] Y. Li and X. Chen, “Mobile robot navigation using particle swarm optimization and adaptive NN,” in *Proc. 1st Int. Conf. Nat. Comput.*, Changsha, China, *LectureNotesinComputerScience*, vol. 3612. Berlin, Germany: Springer-Verlag, 2005, pp. 554–559.

[9] P. J. Angeline, “Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences,” in *Proc. 7th Annu. Conf. Evol. Programming VII*, 1998, pp. 601–610.

[10] M. Clerc and J. Kennedy, “The particle swarm—Explosion, stability, and convergence in a multidimensional complex space,” *IEEE Trans. Evol. Comp.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

[11] A. Stacey, M. Jancic, and I. Grundy. Particle swarm optimization with mutation, Proc. of the 2003 IEEE Congr. on Evol. Comput., pp. 1425-1430,2003.

[12] P. J. Angeline. Evolutionary optimization versus particle swarm optimization: Philosophy and performance difierences. Evolutionary Programming VII, pages 601-610, 1998.

[13] A. Carlisle and G. Dozier. An ofi-the-shelf pso.Proceedings of the Particle Swarm Optimization Workshop , pages 1-6, April 2001.

[14] P. Yadmellat, S.M.A. Salehizadeh, M.B. Menhaj, "Fuzzy Parameter Particle Swarm Optimization". *In Proc: International Conference on Intelligent Networks and Intelligent Systems*, Whuan, China, pp. 93-98, Nov. 2008.

[15] A. Ratnaweera, S. Halgamuge, and H. Watson. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration co e–cients. In IEEE Transactions on Evolutionary Computation , volume 8, pages 240-255, 2004.

[16] Christopher K. Monson, Kevin D. Seppi, “Adaptive diversity in PSO,”, Proceedings of the 8th annual conference on Genetic and evolutionary computation, Seattle, Washington, USA, SESSION: Ant colony optimization and swarm intelligence, Pages: 59 – 66, 2006.

[17] Jacques Riget and Jakob S. Vesterstom. A diversity-guided particle swarm optimizer — the ARPSO. Technical Report 2002-02, Department of Computer Science, University of Aarhus, 2002.

[18] Thiemo Krink, Jakob S. Vestertroem, and Jacques Riget. Particle swarm optimisation with spatial particle extension. In P roceed i n g s o f t h e IEEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii, 2002.

[19] B. Liu, L.Wang, and Y. H. Jin, “Improved particle swarm optimization combined with chaos,” *Chaos, Solitons Fractals*, vol. 25, no. 5, pp. 1261–1271, 2005.

[20] B. Li and W. S. Jiang, “Optimizing complex functions by chaos search,” *Cybern. Syst.*, vol. 29, no. 4, pp. 409–419, 1998.

[21] Y. Song, Z. Chen, and Z. Yuan, "New chaotic PSO-based neural network predictive control for nonlinear process," IEEE Transactions on Neural Networks, vol. 18, no. 2, pp. 595-601, 2007.

[22] Fuqing Zhao, “A Hybrid Algorithm Based on PSO and GA to Dynamic Virtual Holon Mechanism and Negotiation Model”, In the Proceedings of ISKE-2007, October 2007.

[23] Matthew Settles, Paul Nathan, Terence Soule, “Breeding swarms a new approach to recurrent neural network training”. GECCO 2005 185-192.

[24] C. Li, S. Yang, and I. A. Korejo. “An adaptive mutation operator for particle swarm optimization”. Proceedings of the 2008 UK Workshop on Computational Intelligence, pp. 165-170, 2008.

[25] M. Lovb jerg, T. K. Rasmussen, and T. Krink. Hybrid particle swarm optimizer with breeding and subp opulation. Genetic And Evolutionary Computation Conference (GECCO’01) , pages 469-476, July 2001.

[26] R. Poli, C. D. Chio, and W. B. Langdon. Exploring extended particle swarms: a genetic programming approach. Genetic And Evolutionary Computation Conference (GECCO’05) , pages 169-176, 2005.

[27] R. A. Krohling. Gaussian particle swarm with jumps. Proc. of the 2005 IEEE Congr. on Evol. Comput., pp. 1226-1231, 2005.

[28] R. A. Krohling and L. dos Santos Coelho. PSO-E: Particle swarm with exp onen- tial distribution. Proc. of the 2006 IEEE Congr. on Evol. Comput., pp. 1428-1433, 2006.

[29] N. Higashi and H. Iba. Particle swarm optimization with Gaussian mutation, Proc. of the 2003 IEEE Swarm Intelligence Symphosium, pp. 72-79, 2003.

[30] S. C. Esquivel and C. A. Coello. On the use of particle swarm optimization with multimodal functions, Proc. of the 2003 IEEE Congr. on Evol. Comput., pp. 1130- 1136, 2003.

[31] C. Li, Y. Liu, L. Kang, and A. Zhou. A Fast Particle Swarm Optimization Algorithm with Cauchy Mutation and Natural Selection Strategy. ISICA2007, LNCS4683, pp. 334-343, 2007.

[32] H. Wang, Y. Liu, C. Li, and S. Zeng, A Hybrid Particle Swarm Algorithm with Cauchy Mutation, Proc.of the 2007 IEEE Swarm Intelligence Symposium, 2007.

[33] H. Wang, Y. Liu, S. Zeng, and C. Li. Opposition-based Particle Swarm Algorithm with Cauchy Mutation. Proc. of the 2007 IEEE Congr. on Evol. Comput., 2007.

[34] Srinivas Pasupuleti and Roberto Battiti, “The Gregarious Particle Swarm Optimizer (GPSO)”, GECCO 2006.

[35] K. Funahashi and Y. Nakamura, “Approximation of dynamical systems by continuous-time recurrent neural network,” Neural Networks, vol. 6, pp. 801–806, 1993.

[36] J. J. Liang, P. N. Suganthan and K. Deb, "Novel Composition Test Functions for Numerical Global Optimization”, *In Proc: Swarm Intelligence Symposium, 2005 (SIS 2005), Proceedings 2005 IEEE*, pp. 68-75, June 2005.

[37] Shi Y. Eberhart RC, “Parameter selection in particle swarm optimization”, *1998 Annual Conference on Evolutionary Programming*, San Diego, March 1998.

[38] Shi Y. Eberhart RC, “Empirial, study of particle swarm optimization”, *1999 Conference on Evolutionary Programming*, Washington DC, USA, July 6-9.

[39] Zheng Qin, Fan Yu, Zhewen Shi, Yu Wang, “Adaptive inertia weight particle swarm optimization”, In Proc: ICAISC 2006, pp.450-459.

[40] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer". *Proc. of the IEEE Congress on Evolutionary Computation (CEC 1998)*, Piscataway, NJ, pp. 69-73, 1998.

[41] Wang L. "Intelligent optimization algorithms with applications", Beijing: Tsinghua University & Springer Press; 2001.

[42] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization." *IEEE Transactions on Evolutionary Computation*, 8(3):225 - 239, June 2004.

[43] J. J Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Evaluation of Comprehensive Learning Particle Swarm Optimizer." *Lecture Notes in Computer Science,* Vol. 3316, pp. 230-235, 2004.

[44] N. Hansen, S. D. Muller and P. Koumoutsakos, "Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)." *Evolutionary Computation*, 11(1):1-18, 2003.

[45] K. Deb, A. Anand, and D. Joshi, "A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimization," *KanGAL Report* No. 2002003, April 2002.

[46] R. Storn, and K. Price, "Differential evolution-A simple and Efficient Heuristic for Global Optimization over Continuous Spaces." *Journal of Global Optimization,* 11:341-359, 1997.

[47] Shi Y. Eberhart RC, “Empirical Fuzzy adaptive particle swarm optimization”, *Evolutionary Computation, 2001. Proceedings of the 2001 Congress,* Volume 1, Issue, 2001, Page(s):101 – 106 vol. 1.