Improving Cooperative PSO using Fuzzy Logic

Zahra Afsahi[1], Mohammadreza Meybodi[2]

Abstract. PSO is a population-based technique for optimization, which simulates the social behavior of the fish schooling or bird flocking. Two significant weaknesses of this method are: first, falling into local optimum and second, the curse of dimensionality. In this work we present the FCPSO-H to overcome these weaknesses. Our approach was implemented in the cooperative PSO, which employs fuzzy logic to control the acceleration coefficients in velocity equation of each particle. The proposed approach is validated by function optimization problem form the standard literature simulation result indicates that the approach is highly competitive specifically in its better general convergence performance.

Keywords. Particle swarm optimization, Fuzzy logic, Cooperative learning, Optimization

1 Introduction

Particle swarm optimization (PSO) was motivated from the simulation of simplified social behavior of animals (Kennedy and Eberhart 2005). It has already been applied successfully in many application areas where GA can be applied to (Eberhart and Shi 1998). However, the original PSO has difficulties in controlling the balance between exploration and exploitation where the environment itself is dynamically changed over the time. PSO cannot able to adapt dynamically to the changing environment and quickly converging toward an optimum in the first period of iteration. Another main drawback of the original PSO is that it may get stuck in a sub-optimal solution region and the problem usually gets harder for high-dimensional problems usually known as “curse of dimensionality”. Hence, a new hybrid PSO algorithm is proposed in this paper. The proposed algorithm integrates both fuzzy logic and cooperative learning within a unified framework to further improve the performance. The use of fuzzy logic is suitable for dynamically tuning the programming coefficient C_k, since it starts a run with an initial value which is changed during the run. By using the fuzzy control approach, these parameters can be adaptively regulated according the problem environment and can balance the exploration and exploitation. The cooperative learning splits a composite high dimensional swarm into several smaller dimensional swarms, which cooperate with each other by exchanging information to determine composite fitness of the entire system. The rest of the paper is organized as

[1] Zahra Afsahi, Department of Electrical and Computer Engineering and Information Technology, Islamic Azad University Qazvin, Iran, e-mail: Afsahi_ai@yahoo.com.

[2] MohammadReza Meybodi, Computer Engineering and Information Technology Department, Amirkabir Universsity, Tehran, Iran, e-mail:mmeybodi@ce.aut.ac.ir.

follows. Section 2 describes the PSO algorithm. Section 3 motivates the cooperative learning in PSO algorithm. In section 4 we will introduce pervious works in hybrid models of fuzzy logic and PSO, briefly. Section 5 presents our hybrid fuzzy cooperative PSO algorithm. Performance evaluation of our approach is presented in detail in section 6. Conclusions are presented in section 7.

2 PSO algorithm

The particle swarm optimization (PSO) method is a member of the wide category of *Swarm Intelligence* methods (Kennedy and Eberhart and Shi 2001); for global optimization problems. It was originally proposed by J. Kennedy as a simulation of social behavior, and it was initially introduced as an optimization method in 1995(Eberhart and Kennedy 1995, Eberhart and Simpson and Dobbins 1996). PSO is related with Artificial Life, and specially to Swarming theories. PSO is a population-based technique and each individual of the population has an adaptable velocity (position change), according to which it moves in the search space. Moreover, each individual has a memory, remembering the best position of the search space it has ever visited. Thus, its movement is an aggregated acceleration towards the best individual of a topological neighborhood.

Suppose that the search space is D-dimensional, and then the *i*th particle of the swarm can be represented by a D-dimensional vector, $X_i = (x_{i1}, x_{i2}, ...x_{iD})$. The velocity (position change) of this particle, can be represented by another D-dimensional vector, $V_i = (v_{i1}, v_{i2}, ..., v_{iD})$. The best previously visited position of the *i*th particle is denoted as $P_i = (p_{i1}, p_{i2}, ..., p_{iD})$. Defining g as the index of the best particle in the swarm (*i.e*, the *g*th particle is the best), and let the superscripts denote the iteration number, then the swarm is manipulated according to the following equations:

$$V_{i,j}(t+1) = w * V_{i,j}(t) + C_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) + C_2 r_{2,j}(t)(\hat{y}_{i,j}(t) - x_{i,j}(t)) \quad (1)$$

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

Where C_1, C_2 are positive constants, called acceleration constant; r_1, r_2 are random numbers uniformaly distributed in [0, 1]; and $n = 1,2,...$ determines the iteration numbers. w is the first new parameter added into the original PSO algorithm, *inertia weight* (Shi and Eberhart 1998a, 1998b).

Depending on the social network structure of the swarm *l*best and *g*best experience of particles, exchange among them. Thereto, Kennedy and Mendes recommended the Van-Neumann architecture (Kennedy and Mendes 2002), in which a particle's neighbors are above, below and on each side on a two dimensional lattice, to be the most promising one. In this paper, the social network structure of the swarm is *g*best. Therewith for each iteration of a *g*best PSO algorithm, the j_{th} dimension of particle i_{th} 's velocity vector, and its position vector, X_i is updated as follows:

$$y_i(t+1) = \begin{cases} y_i(t) & \text{if} \quad f(X_i(t+1)) \geq f(y_i(t)) \\ X_i(t+1) & \text{if} \quad f(X_i(t+1) < f(y_i(t)) \end{cases} \tag{3}$$

The PSO algorithm performs repeated applications of the update equations until a specified number of iterations have been exceeded, or until a user-defined stopping criterion has been reached.

3 Cooperative learning in PSO

The original PSO uses a population of *n*-dimensional vector. In cooperative learning (Bergh and Engelbercht 2004), instead of creating a composite swarm of *S* particles for an *n*-dimensional problem, we create *n* one-dimensional swarms, each comprising *S* particles. Each swarm attempts to optimize a single component of the solution vector, essentially a one-dimensional optimization problem. This decomposition used in the relaxation method (Southwell 1946). One complication is the fact that the function to be minimized, *f*, requires an *n*-dimensional vector as input. If each swarm represents only a single dimension of the search space, it is clearly not possible to directly compute the fitness of the individuals of a single population considered in isolation. Hence the fitness evaluation for a cooperative PSO is carried out by the introduction of a "context vector" (denote by *b*). For a *d*-dimensional problem, the context vector is also of *d*-dimensional size. Here when a given j_{th} swarm is active the "context vector" is formed by the *g*best particles of the other (*d-1*) swarms, which are kept constant during the evaluation of the j_{th} swarm, and the j_{th} row of the "context vector" is filled with each particle of the j_{th} swarm one by one. Each such *context vector* formed is evaluated for its composite fitness. But note that, should some of the components in the vector be correlated, they should be grouped in the same swarm, since the independent changes made by the different swarms will have a detrimental effect on correlated variables (Friedman and Savage 1947). This results in some swarms having one-dimensional vectors and others having *c*-dimensional vector (*c*<*d*). In this paper we used $CPSO - H_k$ algorithm, which combines $CPSO - S_k$ with PSO in an attempt to retain the best properties of both algorithms, and the mechanism for information exchange is to replace some of the particles in one half of the algorithm with the best solution discovered so far by the other half of the algorithm. Specially, after an iteration of the $CPSO - S_k$ half of the algorithm, the context vector $b(1, P_1.\hat{y})$ is used to overwrite a randomly chosen particle in the PSO half of the algorithm (the Q swarm). This is followed by an iteration of the Q swarm component of the algorithm, which yields a new global best particle, $Q.\hat{y}$. This vector is then split into sub-vectors of the right dimensions and used to overwrite the positions of randomly chosen particles in the P_j swarms.

During subsequent iterations more particles will be drawn to the global best particle, possibly discovring better solutions along the way, thus, the normal operation of the swarm is not disurbed.

4 Related works

Several algorithms have so far been proposed in literatures that have addressed the issue of using fuzzy logic in particle swarm optimization. Shi and Eberhart after introducing a linearly decreasing inertia weight to the PSO over the course of PSO, designed fuzzy systems to nonlinearly changing the inertia weight (Shi and Eberhart 2001a). This fuzzy system has some measurements of the PSO performance as the input and the new inertia weight as the output of the fuzzy systems.

Kang, Wang, and Wu proposed a novel fuzzy adaptive optimization strategy for the PSO (Kang, Wang and Wu 2006). In multi-optimum static programming mode (MSPPSO), the programming proportion factor of multi-optimum cannot be dynamically adjusted in the optimization process. On the basis of MSPPSO, a kind of fuzzy adaptive programming strategy based on double-variable and single-dimensional fuzzy control structure is proposed. Liu, Abraham and Zhang introduced the FATPSO algorithm. In this method, they proposed a fuzzy logic-based system to tune adaptively the velocity threshold (Liu, Abraham and Zhang 2007). Niu, *et al*. employed T-S fuzzy model to represent a non-linear system. T-S fuzzy system is described by a set of fuzzy IF-THEN rules that represent local linear input-output relations of non-linear systems. The overall system is then an aggregation of all such local linear models (Niu *et al* 2007). Noroozi and Meybodi combined fuzzy logics with PSO. In this method C_1, C_2 are controlled by 4 fuzzy rules. The result of this algorithm is affected directly by the way of choosing the fuzzy rules (Noroozi and Meybodi 2008a, 2008b). Zahiri and Seyedin introduced particle swarm classifier with the concept of intelligently controlling the search process of PSO to develop an efficient swarm intelligence based classifier. An intelligent fuzzy controller is designed to improve the performance and efficiently of the proposed classifier by adapting three important parameters of PSO, inertia weight, cognitive parameter and social parameter (Zahiri and Seyedin 2007). Abdelbar, Abdelshahid and Wunsch introduced fuzzy PSO, a generalization which differs from standard PSO, because of a new fuzzy variable, *charisma* (Abdelbar, Abdelshahid, Doland and Wunsch 2005). In this method more than one particle in each neighbourhood can have a non-zero degree of charisma and consequently is allowed to influence others to a degree that depends on its charisma. Liu and Abraham proposed a hybrid meta-heuristic fuzzy scheme, called as variable neighbourhood fuzzy particle swarm algorithm, based on fuzzy particle swarm optimization and variable neighbourhood search to solve the QAP (Liu and Abraham 2007).

5 The Proposed System

5.1 Overview

The acceleration constants C_1, C_2 represent the weighting of the stochastic acceleration terms that pull each particle toward *p*best and *g*best positions. Low values allow particles to roam far from target regions before being tugged back, while high values result in abrupt movement toward target regions. Early

experience with PSO (mostly trial and error) led us to set the acceleration constants C_1, C_2 equal to 2.0, but it is not a usual rule (Rantaweera, Halgamuge and Watson 2004).

In the search process, sometimes social behaviour may be more important than cognitive behaviour and vice versa. Regarding some statistical parameters, which are calculated in each iteration, it is possible to tune the cognitive parameter and social parameter adaptively to steer the whole swarm and each particle toward proper trajectory. For example, in the beginning of the search process, the global search is more important than local search. By tuning the cognitive parameter to a small value, the global search ability and maximal interaction in the swarm is achieved. Contrariwise, after enough iteration, when a good position is achieved, the cognitive parameter must be increased to emphasize local search ability. When, the fitness value of a particle becomes better and better, part of search space explored by the particle should becomes smaller and smaller. It means social interaction should be decreased. On the other hands, less improvement in the particle fitness causes a sociality search for the exploration. This means that the value of the cognitive parameter should be decreased and the value of the social parameter should be increased. The use of fuzzy logic would be suitable for dynamically tuning the cognitive parameter and social parameter, since it starts a run with an initial value which is change during the run. By using the fuzzy control approach, these parameters can be adaptively regulated according to the problem environment.

5.2 Fuzzy logic in CPSO – H_k

In this section we introduce our proposed algorithm; $CPSO-H_k$. Figure 1 illustrates the structure of our fuzzy system controller.

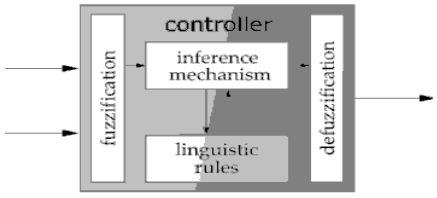


Fig.1 The fuzzy system controller

This system is composed of a knowledge base, that includes the information given by the expert in the form of linguistic control rules that include 25 rules which are shown in table 1, a fuzzification and a defuzzification interface, two variables are selected as inputs to the fuzzy system: the distance between current position and its best position and the distance between current position and the best position of the whole swarm on each dimension and the output variable is the ratio between the programming coefficients, $\Delta C_1 / C_2$. For example when D_1 is "z" and D_2 is "HN", means that we are very close to local optimum and far away from global optimum by increasing C_2 and decreasing C_1, we can fly local optimum, thus we set the fuzzy value of $\Delta C_1 / C_2$ to "LN". All the rules of table 2 are based on this concept.

The goal of these rules is controlling the balance between exploration and exploitation. There into, the value of the programming coefficients is obtained dynamically, because this value depends on the fuzzification of the D_1 , D_2 and these parameters (D_1 , D_2) change in each iteration.

Table 1 Fuzzy rule base for our system

D_1 / D_2	**HN**	**LN**	**Z**	**LP**	**HP**
HN	LP	MN	LN	LN	LP
LN	HP	Z	MN	Z	MP
Z	HP	MP	Z	MP	HP
LP	HP	Z	MN	LP	HP
HP	LP	LN	LN	LN	HP

It should be mentioned that different kinds of inputs, outputs, membership function shapes, membership function locations and fuzzy rules may be introduced and even these parameters can be optimized by another optimization algorithm. In this paper the membership function and its locations are selected and tuned manually. In the process of the fuzzification of input and output variables, we selected the same fuzzy sets.

In $FCPSO\_H$ algorithm, $CPSO-S_k$ algorithm executes for one iteration and follows by one iteration of the PSO algorithm and fuzzy rules are used in each iteration.

6 Experiments

The proposed algorithm was tested on ten numerical functions as benchmarks. The list of these function are given in table 2. Simulations were carried out to find the global minimum of each function. In our experiments, the algorithms used for comparison were PSO, $CPSO-H_k$, FCPSO and $CPSO-H_k$. Each algorithm was tested with all the numerical functions. In our experiment the specific parameter settings for each of the considered algorithms are described in table 2. In table 3 all the value of fuzzy sets as an example for Rosenbrock function which is a simple unimodal function is shown. These values are different for all the other numerical function and depend on the optimum values of these functions.

The first 2 functions, namely Sphere and Rosenbrock functions have a single minimum while the other functions are highly multimodal with multiple local minima. We tested the algorithms on the different functions in 10 and 30 dimensions for each of these functions. The goal was to find the global minima. Each algorithm (for each benchmark) was repeated 25 times. Each trial used fixed number of 30000 iterations. Since the swarm size in all algorithms was 30. The average fitness values of the best solutions throughout the optimization run were recorded and the averages and the standard deviation indicate the differences in the results during the 10 different trials.

Table 2 Test functions, global optimum, search ranges and initialization ranges

Name	F	X^*	Search Range	Initiation Range
Sphere	$f_1(x)=\sum_{i=1}^{D} x_i^2$	[0,0, …,0]	[-100, 100]^D	[-100, 50]^D
Rosenbrock	$f_2(x)=\sum_{i=1}^{D-1}(100(x_i^2-x_{i+1}^2)^2+(x_i^2-1)^2)$	[1,1, …,1]	[-2.048, 2.048]^D	[-2.048, 2.048]^D
Ackley	$f_3(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\Pi x_i)\right)+20+e$	[0,0, …0]	[-32.76, 32.76]^D	[-32.768, 16]^D
Greiwank	$f_4(x)=\sum_{i=1}^{D}\frac{x_i^2}{4000}-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[0,0 …0]	[-600, 600]^D	[-600, 200]^D
Weierstrass	$f_5(x)=\sum_{i=1}^{D}\left(\sum_{k=0}^{k\max}[a^k\cos(2\pi b^k(x_i+0.5))]\right)-D\sum_{k=0}^{k\max}[a^k\cos(2\pi b^k 0.5)],a=0.5,b=3,k\max=20.$	[0,0, …0]	[-0.5, 0.5]^D	[-0.5, 0.2]^D
Rastrigin	$f_6(x)=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10)$	[0,0, …0]	[-5.12, 5.12]^D	[-5.12, 2]^D
Noncontinuous Rastrigin	$f_7(x)=\sum_{i=1}^{D}(y_i^2-10\cos(2\pi y_i)+10)$ $y_i=\begin{cases}x_i & \lvert x_i\rvert<1/2\\ \frac{round(2x_i)}{2} & \lvert x_i\rvert>=1/2\end{cases}$ $fori=1,2,...,D$	[0,0, …0]	[-5.12, 5.12]^D	[-5.12, 2]^D
Schwefel	$f_8(x)=418.9829*D-\sum_{i=1}^{D}x_i\sin\left(\lvert x_i\rvert^{\frac{1}{2}}\right)$	[420.96, … 420.96]	[-500, 500]^D	[-500, 500]^D

Table 3 Benchmark for simulation

Function	Fuzzy Variable	H_n	L_n	Z	L_p	H_p
f_1	$D1_{j,i}$	(-100,-50)	(-30,5)	(-5,5)	(-5,30)	(50, 100)
	$D2_{j,i}$	(-100,-50)	(-30,5)	(-5,5)	(-5,30)	(50, 100)
	$\Delta C_1 / C_2$	(-0.1,-0.04)	(-0.025 , 0.005)	(-0.005 , 0.005)	(-0.005 ,0.025)	(0.04, 0.1)

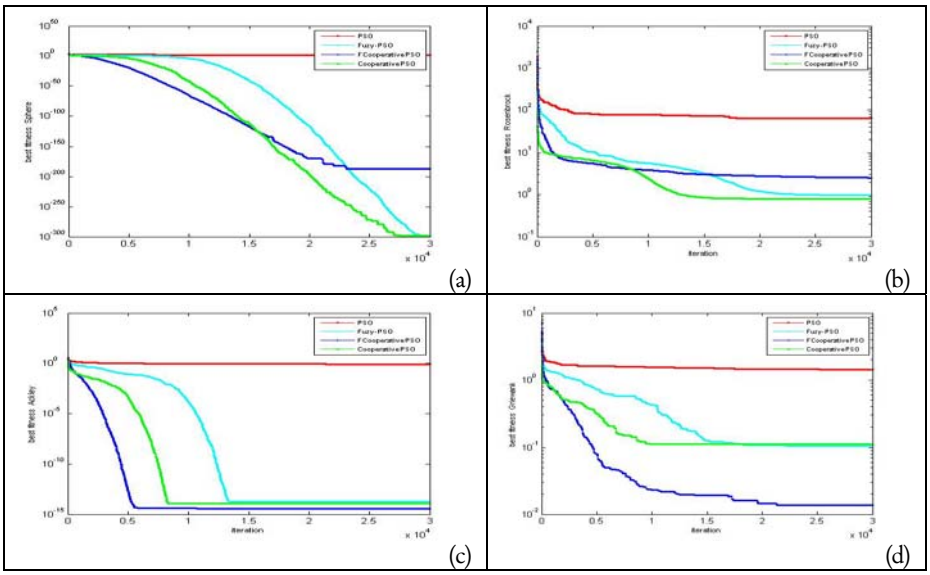
Table 4 and 5 present the results for 10 dimensional and 30 dimensional versions of test functions, respectively. The means and variances of 30 runs for four algorithms and eight functions are reported and best results are shown in bold.

Figure 2 (a-h) illustrate the mean best function values for ten benchmark functions with 10 dimensions and figure 3 (a-h) illustrate it in 30 dimensions, using the four algorithms. Each algorithm for different dimensions of the same objective function has similar performance. It is observed that for PSO algorithm, the solutions get trapped in a local minimum for the low dimensional problems; fuzzy-

PSO is usually a cost efficient cost choice. It is memorable that, our proposed algorithm $FCPSO-H$ could converge to zero in Rastirigin function and could converge to a better solution in Griewank function.

Table 4 Results for 10-D problems

Algorithm	**F1**	**F2**	**F3**	**F4**
Original PSO	61.6 ± 207.12	68.824 ± 97.387	0.7199 ± 0.01949 5	**1.5058 ± 0.0026229**
Fuzzy PSO	**3.3297e-314 ± 0**	**0.11752 ± 1.2629**	2.1316e-014 ± 1.7895e-028	**0.59075 ± 0.0015774**
Cooperative PSO/CPSO-H	1.0872e-311 ± 0	3.9871 ± 2.8252	7.1054e-015 ± 5.9603e-029	**0.23848 ± 0.0025699**
Fuzzy CPSO-H	9.1231e-214 ± 0	2.5446 ± 1.2453	**3.5527e-015 ± 0**	0.012316 ± 0.00011212
Algorithm	**F5**	**F6**	**F7**	**F8**
Original PSO	8.8185 ± 0.39671	6.2292 ± 1.2521	34.066 ± 15.034	**4151.1 ± 0.011978**
Fuzzy PSO	0.52964 ± 0.13468	0 ± 2.9801e-030	**8.8818e-015 ± 2.1778**	**4150.8 ± 0.001993**
Cooperative PSO/CPSO-H	**0.0021739 ± 0.078251**	0 ± 1.7744e-028	1 ± 0.23333	**4150.4 ± 2.6164e-017**
Fuzzy CPSO-H	**0.0039687 ± 1.4817e-006**	0 ± 0	**1 ± 0.44444**	4150.3 ± 9.1909e-026



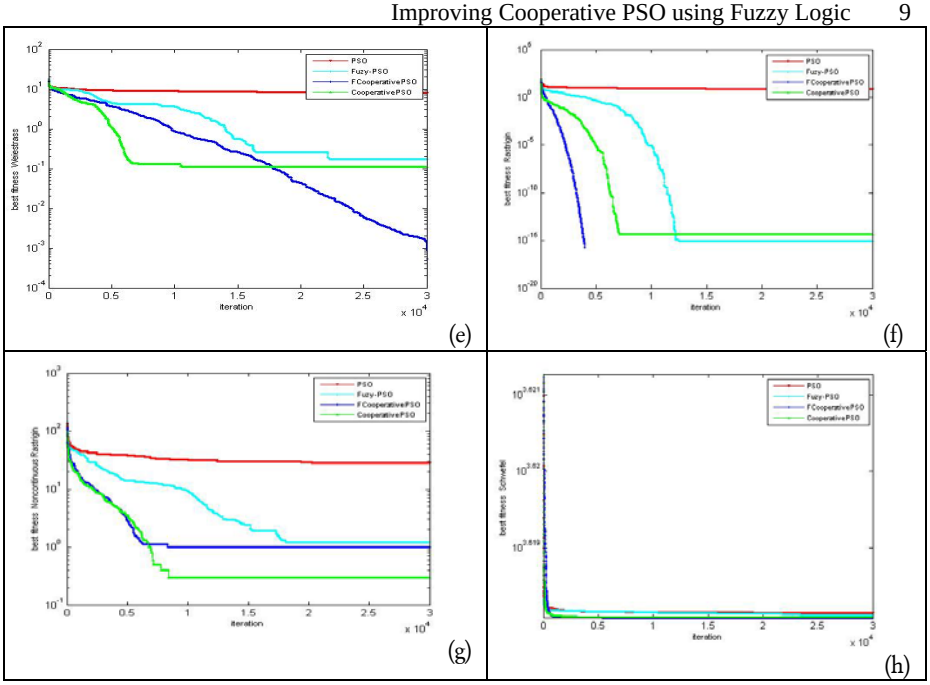


Fig. 2 The median convergence characteristics of 10-D test functions, (a) Sphere. (b) Rosenbrock. (c) Ackley. (d) Griewank. (e) Weierstrass. (f) Rastrigin. (g) Noncontinuous Rastrigin. (h) Schwefel's

Table 5 Results for 30-D problems

Algorithm	**F1**	**F2**	**F3**	**F4**
Original PSO	941.31 ± 13119	1610.6 ± 7127.1	2.1537 ± 0.012453	**7.3611 ± 0.30691**
Fuzzy PSO	1.1967e-040 ± 8.7323e-072	77.241 ± 567.57	3.4204e-010 ± 3.3604e-020	**0.022151 ± 0.0007942**
Cooperative PSO/CPSO-H	4.7755e-033 ± 1.2981e-048	23.7 ± 6.7224	1.676e-007 ± 4.5419e-014	**0.031942 ± 0.00046773**
Fuzzy CPSO-H	**3.0918e-077 ± 1.8117e-1134**	**14.574 ± 21.261**	**6.3949e-014 ± 2.5819e-028**	0 ± 3.6682e-005
Algorithm	**F5**	**F6**	**F7**	**F8**
Original PSO	36.845 ± 0.82859	109.4 ± 40.833	291.79 ± 184.2	**12456 ± 0.21194**
Fuzzy PSO	1.2692 ± 0.76157	6.2172e-012 ± 1.2733e-023	50 ± 1154.2	**12454 ± 0.13615**
Cooperative PSO/CPSO-H	1.4738 ± 4.975	1.5987e-014 ± 5.981e-018	25 ± 367.34	12451 ± 0.0010138
Fuzzy CPSO-H	0.031958 ± 0.026626	0 ± 5.6097e-031	9 ± 11.556	12451 ± 9.8048e-022

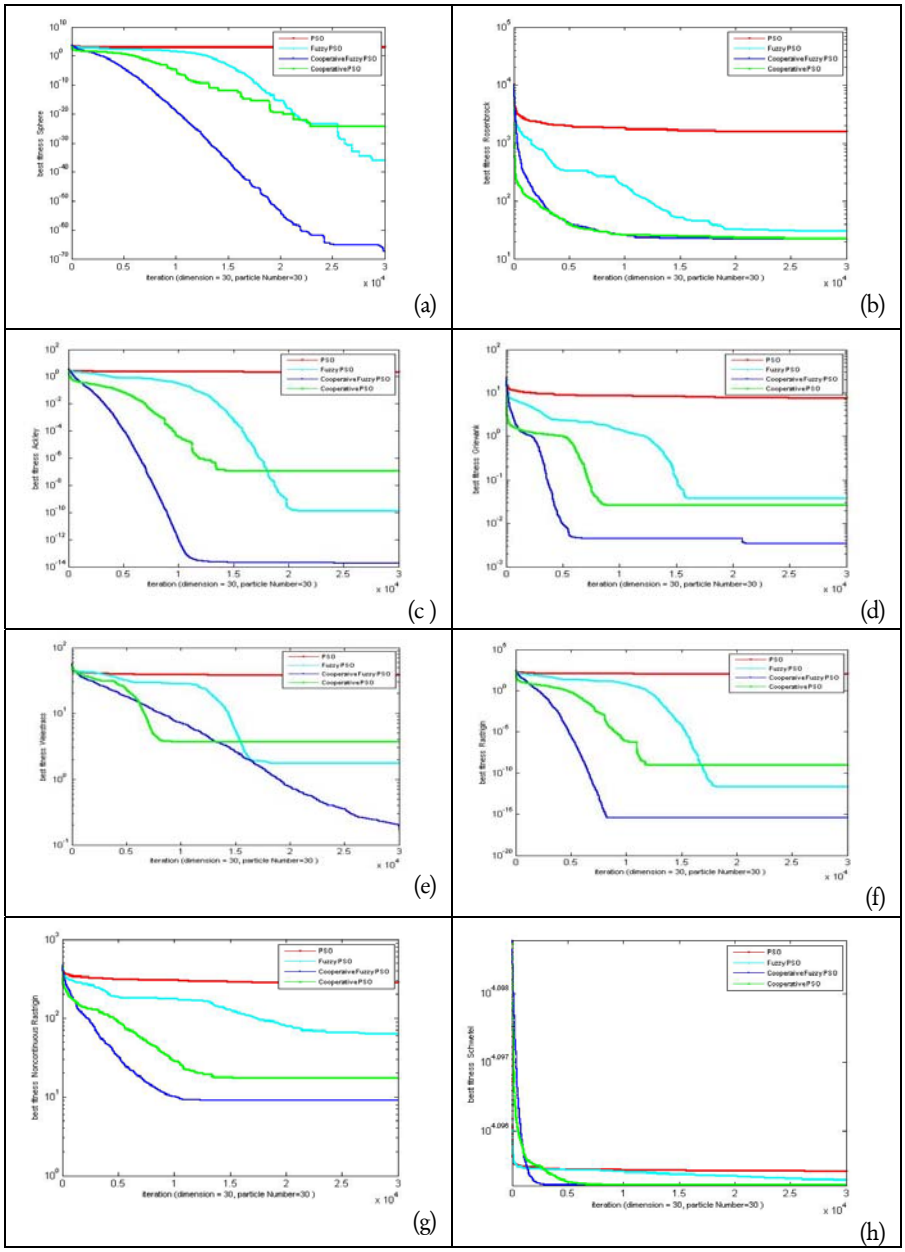


Fig. 3 The median convergence characteristics of 30-D test functions, (a) Sphere. (b) Rosenbrock. (c) Ackley. (d) Griewank. (e) Weierstrass. (f) Rastrigin. (g) Noncontinuous Rastrigin. (h) Schwefel.

7 Result

In this paper, we introduced the $FCPSO-H$ as a solution to overcome the weakness of the PSO algorithm. $FCPSO-H$ algorithm uses the cooperative learning method to overcome the curse of dimensionality. We proposed a fuzzy logic-based system to tune adaptively the acceleration coefficient. Tuning these coefficients adaptively can control balance between global search and local search. Moreover can make the particle continue moving and maintain the diversity of the population until the algorithm converges. We evaluated and compared the performance of our approach, PSO, FPSO and CPSO algorithms on a suite of 8 widely used benchmark problems. The results from our research demonstrated that FCPSO-H generally outperforms the other algorithms, especially for high dimensional functions.

8 Refrences

Abdelbar AM, Adbelshahid S, Wunsch DC (2005) Fuzzy PSO: A generalization of particle swarm optimization. Proceedings of International Joint Conference on Neural Networks, Montreal, Canada, July, 2005.

Bergh VB, Engelbrecht F, A. P (2004) A cooperative approach to particle swarm optimization. IEEE Transactions on Evolutionary Computation (accepted for special issue on PSO).

Eberhart RC, Simpson PK, Dobbins RW (1996) Computational intelligence PC tools. Boston, MA: Academic Press Professional.

Eberhart RC, Shi Y (1998 b) Comparison between genetic algorithm and particle swarm optimization. In V. W.Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds. Evolutionary Programming VII:Proc. 7th Ann. Conf. on Evolutionary Programming Conf., San Diego, CA.Berlin: Springer-Verlag.

Friedman M, Savage SL (1947) Planning experiments seeking minima. In Selected Techniques of Statistical Analysis for Scientific and Industrial Research, and Production and Management Engineering, C. Eisenhart, M. W. Hastay, and W. A. Wallis, Eds. New York: McGraw-Hill, pp. 363–372.

Kang Q, Wang L, Wu Q (2006) Research on fuzzy adaptive optimization strategy of particle swarm algorithm. International Journal of Information Technology, Vol. 12, No.3.

Kennedy J, Eberhart RC (1995) Particle swarm optimization. Processing's of IEEE International Conference on Neural Networks (ICNN), Australia, pp 1942-1948.

Kennedy J, Eberhart RC, Shi Y (2001) Swarm intelligence. Morgan Kaufmann, 2001.

Kennedy J, Mendes R (2002) Population structure and particle swarm performance. Proceeding of the 2002 Congress on Evolutionary Computation, Honolulu, Hawaii.

Liu H, Abraham A, Zhang W (2007) Fuzzy adaptive turbulent particle swarm optimization. Int. J. Innovative Computing and Application, Vol. 1, No. 1.

Liu H, Abraham A (2007) A hybrid fuzzy variable neighbourhood particle swarm optimization algorithm for solving quadratic assignment problems. Journal of Universal Computer Science, Vol. 13, No. 7, pp. 1032-1054.

Niu *et al* (2007) A multi-swarm optimizer based fuzzy modelling approach for dynamics systems. Neurocomputing.

Noroozibeyrami MH, Meybodi MR (2008) Cooperative fuzzy particle swarm optimization. Proceedings of the 2nd Joint Congress on Fuzzy and Intelligent Systems, Malek Ashtar University of Technology, Tehran, Iran, 28-30 October, 2008.

Noroozibeyrami MH, Meybodi MR (2008) Improving particle swarm optimization using fuzzy logic. Proceedings of the Second Iranian Data Mining Conference, Amir Kabir University of Technology, Tehran, Iran, Sept. 21-22, 2008.

Rantaweera A, Halgamuge SK, Watson HC (2004) Self-organizing hierarchical particle swarm optimizer with time varying accelerating Coefficients. IEEE Transactions on Evolutionary Computation (accepted for special issue on PSO).

Shi Y, Eberhart RC (1998a) Parameter selection in particle swarm optimization, Proceedings of the 1998 Annual Conference on Evolutionary Computation.

Shi Y, Eberhart RC (1998b) A modified particle swarm optimizer. Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, 69-73. Piscataway, NJ: IEEE Press.

Shi Y, Eberhart RC (2001a) Fuzzy adaptive particle swarm optimization. Proc. Congress on Evolutionary Computation 2001, Seoul, Korea. Piscataway, NJ: IEEE Service Center.

Southwell RV (1946) Relaxation methods in theoretical physics. Oxford, U.K: Clarendon Press.

Zahiri SH, Seyedin SA (2007) Swarm intelligence based classifiers. Journal of the Franklin Institute 344, pp. 362–376.