Egyptian Informatics Journal (2015) **16**, 339–350

Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

FULL-LENGTH ARTICLE

A new hybrid optimization method inspired from swarm intelligence: Fuzzy adaptive swallow swarm optimization algorithm (FASSO)

Mehdi Neshat [a,*], **Ghodrat Sepidname** [b,1]

[a] *Department of Computer Science, College of Software Engineering, Shirvan Branch, Islamic Azad University, Shirvan, No. 6, Amam Khomini 78, Mashhad, Iran*
[b] *Department of Computer Science, College of Hardware Engineering, Shirvan Branch, Islamic Azad University, Shirvan, No. 67, Kooh Sangi 34, Mashhad, Iran*

Received 26 May 2014; revised 5 April 2015; accepted 25 July 2015
Available online 21 November 2015

KEYWORDS

Swarm intelligence;
Fuzzy logic;
Swallow swarm optimization;
Adaptive

Abstract In this article, the objective was to present effective and optimal strategies aimed at improving the Swallow Swarm Optimization (SSO) method. The SSO is one of the best optimization methods based on swarm intelligence which is inspired by the intelligent behaviors of swallows. It has been able to offer a relatively strong method for solving optimization problems. However, despite its many advantages, the SSO suffers from two shortcomings. Firstly, particles movement speed is not controlled satisfactorily during the search due to the lack of an inertia weight. Secondly, the variables of the acceleration coefficient are not able to strike a balance between the local and the global searches because they are not sufficiently flexible in complex environments. Therefore, the SSO algorithm does not provide adequate results when it searches in functions such as the Step or Quadric function. Hence, the fuzzy adaptive Swallow Swarm Optimization (FASSO) method was introduced to deal with these problems. Meanwhile, results enjoy high accuracy which are obtained by using an adaptive inertia weight and through combining two fuzzy logic systems to accurately calculate the acceleration coefficients. High speed of convergence, avoidance from falling into local extremum, and high level of error tolerance are the advantages of proposed method.

* Corresponding author. Tel.: +98 5118533114; fax: +98 5118526851.
E-mail addresses: neshat@ieee.org (M. Neshat), Sepidname@iau-shirvan.ac.ir (G. Sepidname).
[1] Tel.: +98 5856243901; fax: +98 5856243902.

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.eij.2015.07.003

The FASSO was compared with eleven of the best PSO methods and SSO in 18 benchmark functions. Finally, significant results were obtained.

1. Introduction

In the engineering world, one of the most significant issues is optimization and designing optimum systems. It is crystal clear that one of the best solutions for dealing with the problem is Swarm Optimization. These methods are inspired from some specific animals which live together as a colony or group. Swarm Intelligence (SI) algorithms have been ordinarily comprised of an uncomplicated agent's population or particles which interact concerning a specific location with one another and with their environments. The phenomenon is shown natural world regularly, especially biological agents. The systems follow completely simple rules and structures, notwithstanding that there is not centralized supervision structure commanding how sole factor should act, not general, and to a special degree stochastic, reciprocal activities between like agents bring about the impression of "intelligent" global management, unfamiliar to the single agents. As a result, swarm intelligence is not an obvious explanation and some experts present different definitions for instance SI would be a multi-agent environment which has self-organizing behavior that shows striking smart conduct [1]. The recent decade, there are many examples of SI include ant colony optimization [2–4], Artificial Bee Colony (ABC) [37], Termite Colony Optimization (TCO) [38], bird flocking [5–7], animal herding [8–10], bacterial growth and foraging [11], fish schooling [12–14] and Glowworm Swarm Algorithm [15–18], Swallow Swarm Optimization (SSO) [19]. Furthermore, there are some articles about PSO which are included by adaptive inertia weight [43–45] and fuzzy adaptive acceleration and their results are so considerable [39–41].

In this paper, it is presented a novel algorithm called Fuzzy Adaptive Swallow Swarm Optimization (FASSO) for the simultaneous computation of multimodal functions. This new method assesses and evaluates Swallow Swarm Optimization [19] (SSO) with realizing its weak points, trying to implement the new strategy and improve it. Swallows have high swarm intelligence; in addition, their flying speed is high, so they are able to fly long distances in order to immigrate from one point to another and also they fly in great colonies. Flying collectively they misguide hunters in perilous positions. Swallow swarm life has many special features which are more confusing and bewildering in contrast with fish schools and ant colonies. Therefore, it has been appointed as the subject of research and algorithm simulation.

The second section is about the SSO method and then in the third section, proposed method is introduced. The fourth section is about benchmark function and finally, the fifth section is about experimental results examining the diverse kinds of PSO and then comparing them to SSO and proposed method.

2. Methods

The Swallow swarm algorithm is a new swarm optimization method which was introduced by Neshat et al [19]. Another research which shows efficiency of SSO is a combination of PSO with it [42].

2.1. Swallow Swarm Optimization (SSO)

The SSO algorithm inspired by the collective movement of swallows and the interaction between flock members has attained good results. This algorithm has presented a metaheuristic method based on the special properties of swallows, including fast flight, hunting skills, and intelligent social relations. At a glance, the algorithm is similar to PSO but it has unique characteristics which cannot be found in similar algorithms, including the use of three types of particles: Explorer Particles (e_i), Aimless Particles (o_i), and Leader Particles (l_i), each of which has certain responsibilities in the group. The e_i particles are responsible for searching the problem space. They perform this searching behavior under the influence of a number of parameters [19]:

1. Position of the local leader.
2. Position of the global leader.
3. The best individual experience along the path.
4. The previous path.

The particles use the following equations for searching and continuing the path:

$$V_{HL_{i+1}} = V_{HL_i} + \alpha_{HL} rand()(e_{best} - e_i) + \beta_{HL} rand()(HL_i - e_i) \quad (1)$$

Eq. (1) shows the velocity vector variable in the path of the global leader.

$$\alpha_{HL} = \{\text{if } (e_i = 0 || e_{best} = 0) - - > 1.5\} \quad (2)$$

Eqs. (2) and (3) calculate the acceleration coefficient variable (α_{HL}) which directly impacts individual experiences of each particle.

$$\alpha_{HL} = \begin{cases} \text{if } (e_i < e_{best}) \&\& (e_i < HL_i) & \rightarrow \frac{rand().e_i}{e_i \cdot e_{best}} & e_i, e_{best} \neq 0 \\ \text{if } (e_i < e_{best}) \&\& (e_i > HL_i) & \rightarrow \frac{2rand() \cdot e_{best}}{1/(2 \cdot e_i)} & e_i \neq 0 \\ \text{if } (e_i > e_{best}) & \rightarrow \frac{e_{best}}{1/(2 \cdot rand())} & \end{cases} \quad (3)$$

$$\beta_{HL} = \{\text{if } (e_i = 0 || e_{best} = 0) - - > 1.5\} \quad (4)$$

$$\beta_{HL} = \begin{cases} \text{if } (e_i < e_{best}) \&\& (e_i < HL_i) & \rightarrow \frac{rand() \cdot e_i}{e_i \cdot HL_i} & e_i, HL_i \neq 0 \\ \text{if } (e_i < e_{best}) \&\& (e_i > HL_i) & \rightarrow \frac{2rand() \cdot HL_i}{1/(2 \cdot e_i)} & e_i \neq 0 \\ \text{if } (e_i > e_{best}) & \rightarrow \frac{HL_i}{1/(2 \cdot rand())} & \end{cases} \quad (5)$$

Eqs. (4) and (5) calculate the acceleration coefficient variable (β_{HL}) which directly impacts the collective experiences of each particle. In fact, these two acceleration coefficients are quantified considering the position of each particle in relation to the best individual experience and the global leader [19].

The o_i particles have a completely random behavior in the environment and move through the space without attaining a specific purpose and share the results with other flock members. As a matter of fact, these particles increase the chance of finding the particles which have not been explored by the e_i particles. Also, if other particles get stuck in an optimum local point, there is hope that these particles save them. These particles use the following Eq. (6) for random movements [19]:

$$o_{i+1} = o_i + \left[rand(\{-1,1\}) * \frac{rand(\min, \max)}{1 + rand()} \right] \quad (6)$$

In SSO algorithm there are two types of leaders: the local leader and the global leader. The particles are divided into groups. The particles in each group are mostly similar. Then, the best particle in each group is selected and is called the local leader. Next, the best particle among the local leaders is chosen and is called the global leader. The particles change their direction and converge according to the position of these particles.

2.2. Flowchart of SSO

Programming of the SSO algorithm is some complex, so its flowchart can improve a deeper understanding of its operations. In this research, MATLAB software is used for above simulation. We can see clearly the flowchart of SSO algorithm in Fig. 1.

2.3. Proposal method

The SSO algorithm enjoys many advantages compared to the PSO and fish swarm optimization (FSO), but it has disadvantages as well, which are observed when it is tested and evaluated in various environments. One of its main problems is related to controlling the speed of the particles and their convergence so that sometimes particles pass by an optimum point due to their high speed and do not meet it. This problem causes the occurrence of premature convergence. Using an adaptive inertia weight can be a good help solving this problem. Another problem associated with the SSO is the lack of a proper balance between local and global searches which leads to its poor performance in some environments. A combination of fuzzy inference systems was used in the FASSO to reduce the mentioned problems and to obtain greater flexibility in the acceleration coefficients. These points are discussed thoroughly in the following sections of the article.

2.3.1. Adaptive inertia weight

The movement of particles in the SSO depends on several parameters: the environment related to the problem, the best individual experience of each particle, the position of each particle in local groups, and the study of the best position of each particle among all of the groups. These parameters determine the interaction of each particle with other particles. After finding a suitable position, particles tend to converge around it. Here, one of the most important parameters that determine the speed of convergence is the inertia weight.

According to the research carried out in this regard [21–24], the inertia weight plays the main role in determining the efficiency of swarm optimization methods and this coefficient should have particular features. For example, the speed of the particles should decrease with time so that the environment surrounding the problem is searched more accurately (the local

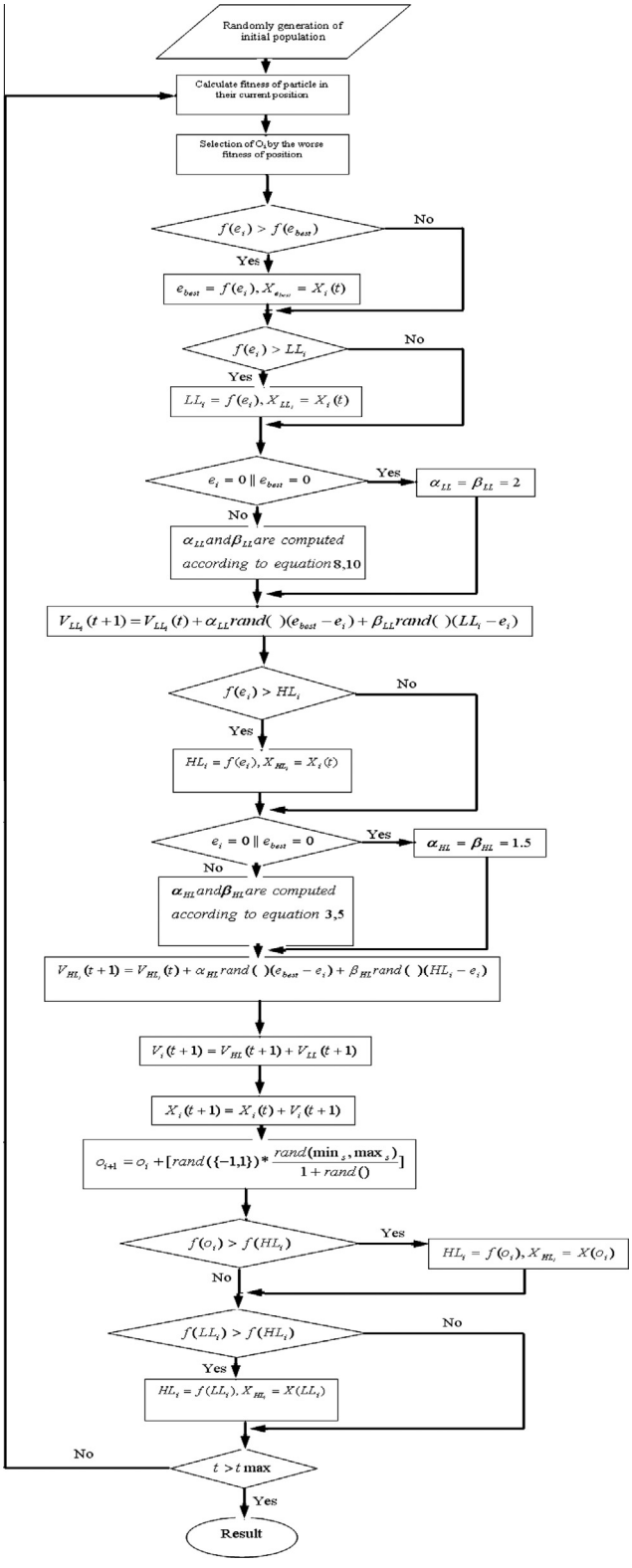

Figure 1 The flowchart of SSO algorithm.

search is performed more accurately). Furthermore, performance will improve if this declining trend is parabola [25]. In the proposed algorithm, an adaptive inertia weight is used that possesses both of the above-mentioned features. Eq. (7) shows this inertia coefficient.

$$W_{t+1} = \{(W_t - W_{\min}) \cdot [(t_{\max} - t)/t_{\max}] + W_{\min}\} \cdot e^{-\left[\frac{t}{\left(\frac{t_{\max}}{4}\right)}\right]^2} \quad (7)$$

2.3.2. Fuzzy acceleration coefficients

Acceleration coefficients have a central role in controlling the establishment of equilibrium between local and global searches. Parameters α_{HL}, α_{LL} illustrate the "self-cognition" that pulls the particle to its own historical best position, assisting explores local recesses and preserving the variety of the swarm. Parameters β_{HL}, β_{LL} represent the "social influence" that shoves the swarm to converge to the current globally best region, improving with fast convergence.

The fuzzy system with the ψ, δ and the θ inputs was used to calculate β_{HL} and α_{HL} parameters, and these inputs were calculated by employing the following equations:

$$\theta = e_{best} - e_i \quad (8)$$

$$\psi = HL_i - e_i \quad (9)$$

$$\delta = HL_i - LL_i \quad (10)$$

The θ parameter (Eq. (8)) shows the difference between the best individual experience of each particle and its present position. In fact, the magnitude of this parameter has a significant effect on the local search process. The ψ parameter (Eq. (9)) indicates the difference between the position of the best leader among all of the groups and the current position of the particle, and based on this difference we can determine how successful the particle has been in its global search and find out whether it has followed a correct path so far. The third parameter δ (Eq. (10)) represents the difference between the best leader among the whole population and the local leader, compares the behavior of the current local group with that of all of the particles, and determines the spatial position of this local group. Of course, all three parameters were used, after fuzzy inferences were made, to determine the acceleration coefficients β_{HL} and α_{HL}. Fuzzy membership functions of all these three inputs are shown below:

The fuzzy system with θ, δ and τ inputs was employed to calculate acceleration coefficients β_{HL} and α_{HL} β_{HL} α_{HL}. Two of the three inputs have equations similar to fuzzy system 1, but the τ input is calculated using Eq. (11).

$$\tau = LL_i - e_i \quad (11)$$

The input parameter τ stands for the difference between the position of the local leader and the current position of the particle and expresses the position of the particle among its group (how close it is to the leader of the group and whether the local search should be improved or not).

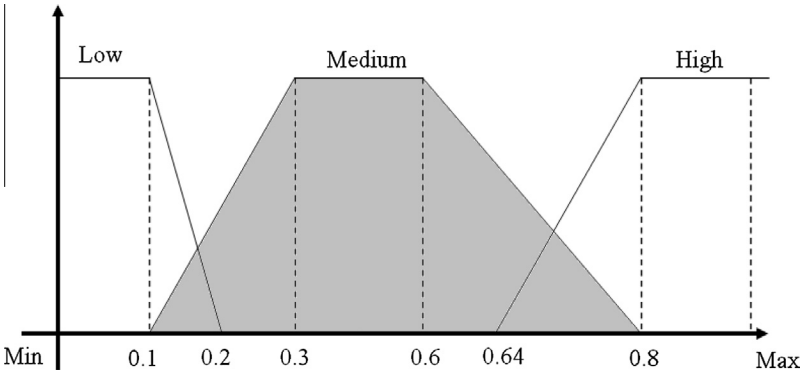

Figure 2 Fuzzy membership functions for the input θ.

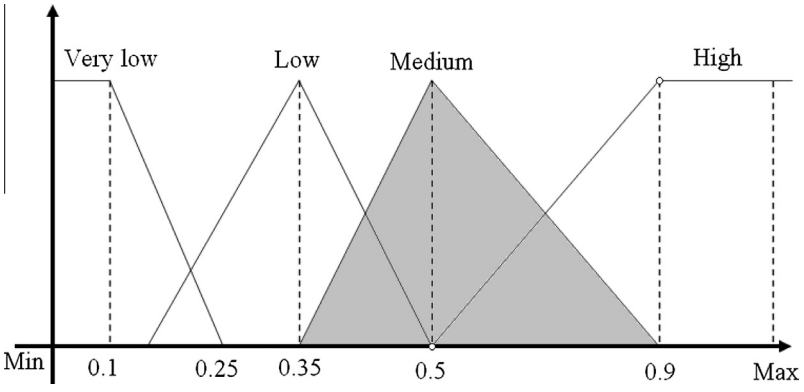

Figure 3 Fuzzy membership functions for the input ψ.

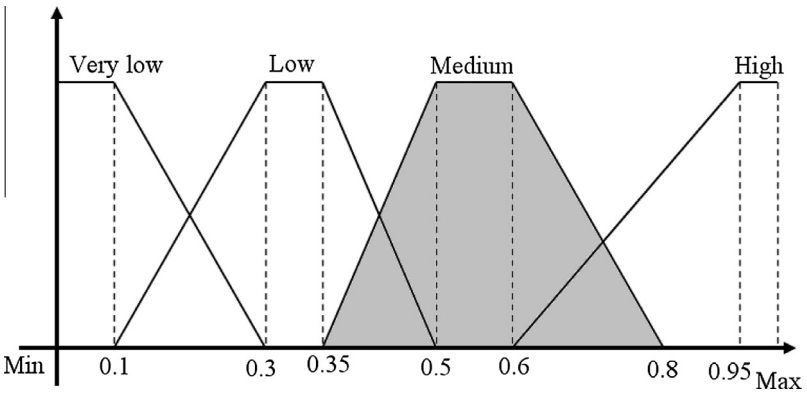

Figure 4 Fuzzy membership functions for the input τ.

As triangular and trapezoidal membership functions have the simplest calculations for fuzzification, they were selected to use in the proposed method; thus, its time complexity is improved. The domains of these functions depend on the type of the benchmark function, so the values in the figures are factors of the intervals of the benchmark functions (min, max). The fuzzy membership functions of the inputs together with their fuzzification equations are explained in the following sections (see Figs. 2–7).

This membership function has the three states of low, medium, and high which show differences between the current situation of each particle and its best individual experience. In the low state, the particle has not made good progress with regard to its past situation and must pay closer attention to the global search in order to improve the conditions. The local and the global searches of the particles will be in equilibrium in the medium state. In the high state, the particles must spend more time on the local search and on the search for the local leader.

$$\mu_{medium}(\theta) = \begin{cases} \frac{x-0.1}{0.2} & 0.1 < x < 0.3 \\ 1 & 0.3 < x < 0.6 \\ \frac{0.8-x}{0.2} & 0.6 < x < 0.8 \end{cases} \quad (12)$$

The ψ input indicates the difference between the present position of the particle and that of a global leader. The fuzzy membership function has the four different states of very low, low, medium, and high. In the low and very low states, there is a short distance between the particle and the global leader and, hence, in general, the colony has a good position and must maintain it. However, in the high state the particles must perform a better global search to reach better conditions in the colony. Eq. (13) was used for the fuzzification of the medium state.

$$\mu_{medium}(\psi) = \begin{cases} \frac{x-0.35}{0.15} & 0.35 < x < 0.5 \\ 0 & x = 0.5 \\ \frac{0.9-x}{0.4} & 0.5 < x < 0.9 \end{cases} \quad (13)$$

The input τ represents the difference between the current position of the particle and that of a local leader. The fuzzy membership function has the four different states of very low, low, medium, and high. In the very low and low states, there is very little distance between the particle and the local leader and hence, the colony enjoys a good position and should focus more on the global search. Therefore, in the high state, particles should carry out better local searches so that the colony can attain more ideal conditions. Eq. (14) was used for the fuzzification of the medium state.

$$\mu_{medium}(\tau) = \begin{cases} \frac{x-0.35}{0.15} & 0.35 < x < 0.5 \\ 1 & 0.5 < x < 0.6 \\ \frac{0.8-x}{0.2} & 0.6 < x < 0.8 \end{cases} \quad (14)$$

The δ input shows the difference between a global leader and a local one. This difference expresses the general situation of a local colony and indicates whether it is in a good position in relation to a global optimum. This fuzzy membership function has the three different states of low, medium, and high. In

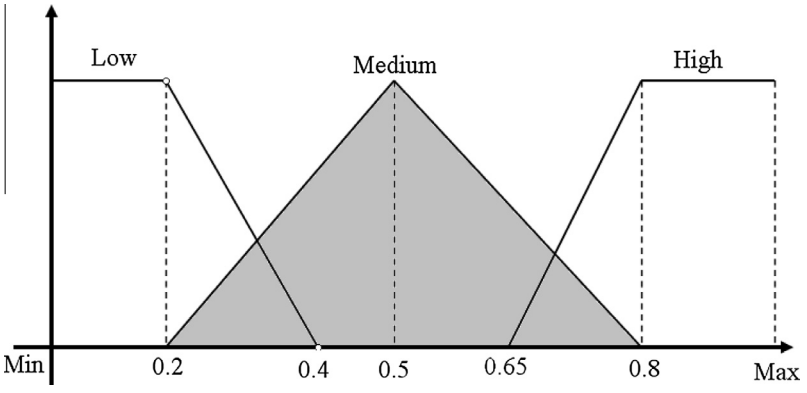

Figure 5 Fuzzy membership functions for the input δ.

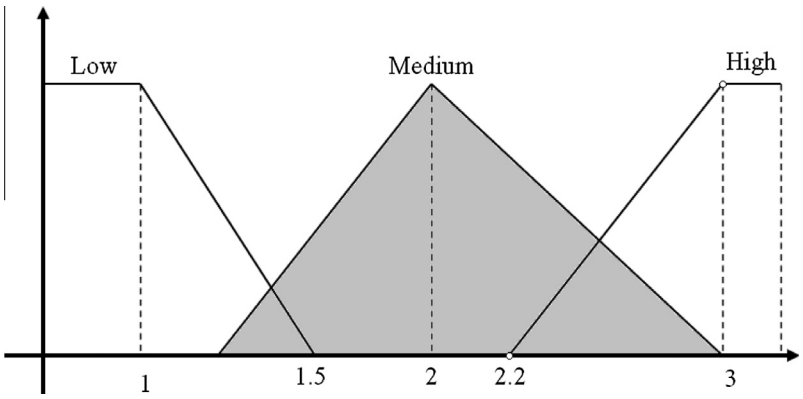

Figure 6 Fuzzy membership functions for the output α_{HL}.

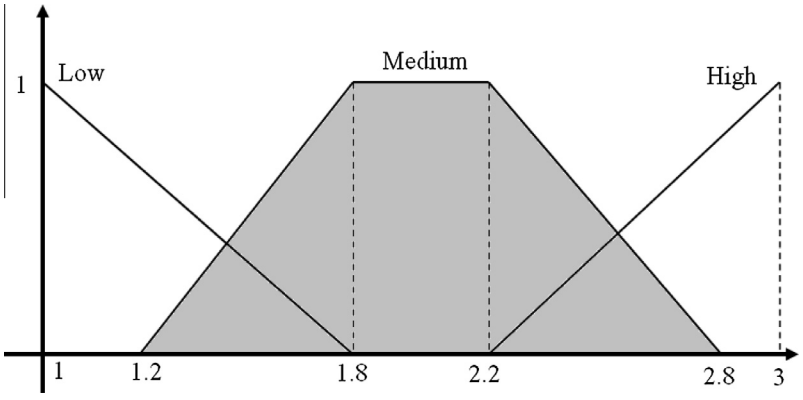

Figure 7 Fuzzy membership functions for the output β_{HL}.

the low state, the local leader has a good strategy, the group is in a suitable position, and particles should continue their local search, so in the high state, the group has not been successful in its search and it is better for the local leader to appear.

Each of the two systems f1 and f2 has two outputs. These outputs are acceleration coefficients α_{HL}, β_{HL}, α_{LL}, β_{LL}. In the sections below, fuzzy membership functions α_{HL} and β_{HL} are presented (Figs. 6 and 7).

Mamdani's fuzzy inference was used for both fuzzy systems. The set of fuzzy rules for fuzzy system 1 is as follows (see Table 1):

The first fuzzy rule was designed for creating a balance between local and global searches. In this state, chances of particles to find a local or a global optimum are the same. In the both second and third fuzzy rules, particles perform the local search with greater care and rely more on their individual experiences. This search may result in not only finding a good local optimum, but also selecting an appropriate local leader. The forth and fifth Fuzzy rules are related to the situation in which the coefficient of the local search declines, the coefficient of the global search rises, and the particles converge around a global optimum. These rules are also known as the convergence rules and it causes a rise to the speed of convergence as well. The sixth and seventh rules help particles to escape from a situation of premature convergence and from a local optimum as well.

The fuzzy rules in the second fuzzy system are also like those in the first fuzzy system; however, there is a great difference for instance in the second fuzzy system, an attempt is made to find a local optimum point and to converge around that point. As a matter of fact, in this system, particles converge around a local leader, and this local leader will converge toward a global leader. Of course, in the SSO algorithm, we never forget the importance of o_i, because not only do these particles give the group the chance of finding unknown areas through performing random and irregular movements, but also allow particles to escape from local optimum points.

Table 1 The set of fuzzy rules for fuzzy inference system 1.

Rules	θ	ψ	δ	$\alpha_H L$	β_{HL}
Rule 1	Medium	Medium	Medium	Medium	Medium
Rule 2	High	Very low	Low	High	Low
Rule 3	High	Low	Low	High	Low
Rule 4	Low	Medium	Medium	Low	High
Rule 5	Low	High	High	Low	High
Rule 6	Medium	High	High	Medium	High
Rule 7	Low	High	Medium	Medium	High

2.3.3. Pseudo-code FASSO

The algorithm of proposed method is shown for better understanding and clarifying.

Fuzzy Adaptive Swallow Swarm Optimization Algorithm:

1. Randomly generation of initial population
2. Calculate fitness of particle in their current position
3. Selection of O_i by the worse fitness of position
4. if $f(e_i) > f(e_{best})$ then $e_{best} = f(e_i), X_{e_{best}} = X_i(t)$
5. if $f(e_i) > LL_i$ then $LL_i = f(e_i), X_{LL_i} = X_i(t)$
6. if $e_i = 0 || e_{best} = 0$ then $\alpha_{LL} = \beta_{LL} = 2$

Else the inputs of first fuzzy system is calculated (δ, θ, ψ)

AND local fuzzy acceleration coefficients $(\alpha_{LL}, \beta_{LL})$ are computed

7. The adaptive inertia weight is determined

$$W_{t+1} = \{(W_t - W_{\min}).[(t_{\max} - t)/t_{\max}] + W_{\min}\}.e^{-\left[\frac{t}{\left(\frac{t_{\max}}{4}\right)}\right]^2}$$

8. The vector of local leader velocity is calculated

$$V_{LL_i}(t+1) = W_t.V_{LL_i}(t) + \alpha_{LL} rand()(e_{best} - e_i) + \beta_{LL} rand()(LL_i - e_i)$$

9. if $f(e_i) > HL_i$ then $HL_i = f(e_i), X_{HL_i} = X_i(t)$
10. if $e_i = 0 || e_{best} = 0$ then $\alpha_{HL} = \beta_{HL} = 1.5$

Else the inputs of second fuzzy system is calculated (δ, θ, τ)

AND global fuzzy acceleration coefficients are computed $(\alpha_{HL}, \beta_{HL})$

11. The vector of global leader velocity is calculated

$$V_{HL_i}(t+1) = W_t.V_{HL_i}(t) + \alpha_{HL} rand()(e_{best} - e_i) + \beta_{HL} rand()(HL_i - e_i)$$

12. The vector of particle velocity is calculated

$$V_i(t+1) = V_{HL}(t+1) + V_{LL}(t+1)$$

13. The position of particle is computed

$$X_i(t+1) = X_i(t) + V_i(t+1)$$

14. Aimless particles are recognized

$$o_{i+1} = o_i + \left[rand(\{-1, 1\}) * \frac{rand(\min_x, \max_x)}{1 + rand()}\right]$$

15. if $f(o_i) > f(HL_i)$ then $HL_i = f(o_i), X_{HL_i} = X(o_i)$
16. if $f(LL_i) > f(HL_i)$ then $HL_i = f(LL_i), X_{HL_i} = X(LL_i)$
17. if $t > t\max$ then goto second level
18. End

2.4. Benchmark functions

To show the performance of the proposed algorithm and to compare it to some different PSOs and FSOs 18 benchmarks have been attended. Each of these functions tests has special conditions and features; thus, feebleness points of the optimization methods will be clear.

3. Experiments

Benchmark functions which have various features were used to evaluate the proposed algorithm all the better and show its weak and strong points. The efficiency of this method (the FASSO) was compared with that of 11 other methods, from PSO to SSO. Table 2 lists the features of all these methods.

This is done to assess the significance of the FASSO among other current methods. To do it, these methods should be tested in same software and hardware. Applied software is MATLAB version 7.0.4 (R14) service pack2. Current hardware is Celeron 2.26-GHz CPU, 256-MB memory, and Windows XP2 operating system. The number of particles is 20 and the number of iterations is 1000. Table 3 shows the performance of the methods of Table 2 and the FASSO method.

As can be seen clearly in Table 3, the proposed method performed better than SSO in all benchmark functions except for the Shifted Rosenbrock's Function. The use of an adaptive inertia weight coefficient, together with the two fuzzy control systems for better determination of the acceleration coefficients improved the efficiency of the SSO algorithm. Interesting results are observed in the above-mentioned table if it is examined more deeply. Firstly, the FASSO method exhibited more intelligent behaviors than all other methods in seven of the benchmark functions (Rosenbrock, Schwefel's P2.22, Quadric, Quadric noise, Perm, Rotated Rastrigin, Rotated Griewank) and had a very good performance so that the results obtained from that were more optimal than those of the other methods. Secondly, the proposed method yielded relatively good results in five of the benchmark functions (Sphere, Rastrigin, Schwefel, N-Rastrigin and Shift Rastrigin). These results were equal to the best results obtained from using the other methods. Thirdly, in six of the benchmark functions used, the FASSO algorithm could not find the best answers, but the answers found were close to the best ones obtained from using the other methods.

According to Fig. 8, the proposed method enjoyed a better efficiency than the other ones. For example, with respect to the Sphere Function, the APSO was the best method after 100 iterations and enjoyed a greater speed of convergence as well. However, after 250 iterations, the proposed method could rapidly follow an optimal path, yield excellent results, and converge around the absolute optimal point at the 500th iteration. Concerning the Ackley function, the FASSO could achieve the best results in fewer than 250 iterations and reached better results than the SSO in a close competition; however, it, unfortunately, fell into a local optimal point and the particles converged around that point. At the 650th iteration, particles could escape from that point with the help of randomly

Table 2 Different methods of PSO and their features.

Algorithm	Year	Topology	Parameters setting	Reference
GPSO	1998	Global star	$w: 0.9-0.4, c_1, c_2 = 2$	[26]
LPSO	2002	Local ring	$w: 0.9-0.4, c_1, c_2 = 2$	[27]
VPSO	2002	Local von Neumann	$w: 0.9-0.4, c_1, c_2 = 2$	[28]
FIPS	2004	Local URing	$\chi = 0.729, \sum c_i = 4.1$	[29]
HPSO-TVAC	2004	Global star	$w: 0.9-0.4, c_1 = 2.5-0.5, c_2 = 0.5-2.5$	[30]
DMS-PSO	2005	Dynamic multi-swarm	$w: 0.9-0.2, c_1 = c_2 = 2, m = 3, R = 5$	[31]
CLPSO	2006	Comprehensive learning	$w: 0.9-0.2, c = 1.49445, m = 7$	[32]
OPSO	2008	Orthogonal particle swarm	$w: 0.9-0.4, c_1 = c_2 = 0.2, V_{\max} = 0.5 * rang$	[33]
APSO	2009	Adaptive swarm	Adaptation of the inertia weight	[34]
OLPSO	2010	Orthogonal learning particle swarm	$w: 0.9-0.4, c = 2, G = 5, V_{\max} = 0.2 * rang$	[35]

Table 3 Comparison between FASSO and several optimization methods.

Algorithm	Sphere	Rosenbrock	Ackley	Griewank	Rastrigin	Schwefel
GPSO	1.98e−053	28.1	1.15e−014	2.37e−002	8.68	−10090.16
LPSO	4.77e−029	21.8627	1.85e−014	1.10e−002	7.25	−9628.35
VPSO	5.11e−038	37.6469	1.4e−014	1.31e−002	8.07	−9845.27
FIPS	3.21e−030	22.5387	7.69e−015	9.04e−004	10.92	−10113.8
HPSO-TVAC	3.38e−041	13	2.06e−010	1.07e−002	3.71	−10868.57
DMS-PSO	3.85e−054	32.3	8.52e−015	1.31e−002	6.42	−9593.33
CLPSO	1.89e−019	11	2.01e−012	6.45e−013	6.64e−011	−12557.65
OPSO	6.45e−018	49.61	6.23e−009	2.29e−003	6.97	−8402.53
APSO	1.45e−150	2.84	1.11e−014	1.67e−002	1.01e−014	*−12569.5*
OLPSO-G	4.12e−054	21.52	7.98e−015	4.83e−003	1.07	−9821.74
OLPSO-L	1.11e−038	1.26	*4.14e−015*	*0*	*0*	−12150.63
SSO	*0*	2.4373e−001	4.7025e−012	4.8516e−008	1.8104e−010	*−12569.5*
FASSO	*0*	*2.1547e−001*	3.1042e−014	1.92e−009	*0*	*−12569.5*
Best method	FASSO	FASSO	OLPSO-L	OLPSO-L	OLPSO-L & FASSO	SSO&APSO &FASSO
Algorithm	Schwefel's P2.22	Quadric	Quadric noise	Perm	N_Rastrigin	Generalized penalized
GPSO	2.51e−034	6.45e−002	7.77e−003	1.02e−001	15.5	1.04e−002
LPSO	2.03e−020	18.6	1.49e−002	1.41e−002	30.4	2.18e−030
VPSO	6.29e−027	1.44	1.08e−002	12.5	21.33	3.46e−003
FIPS	1.32e−017	0.77	2.55e−003	5.68e−001	35.91	1.22e−031
HPSO-TVAC	6.9e−023	2.89e−007	5.54e−002	2.02e−002	1.83	7.07e−030
DMS-PSO	2.61e−029	47.5	1.1e−002	2.78	32.8	2.05e−032
CLPSO	1.01e−013	395	3.92e−003	4.05e−001	16.7e−002	1.59e−021
OPSO	1.26e−010	2.44e−002	4.87e−002	2.33e−002	2.49e−006	1.56e−019
APSO	5.15e−084	1.13e−010	4.66e−003	2.94e−003	4.14e−016	3.27e−031
OLPSO-G	9.85e−030	5.59e−006	6.21e−003	1.28	1.05e−011	1.59e−032
OLPSO-L	7.67e−022	1.56e−001	1.32e−002	5.31e−002	6.32e−009	*1.57e−032*
SSO	1.58e−078	4.16e−015	2.86e−003	1.01e−004	*6.04e−019*	1.84e−031
FASSO	*3.47e−086*	*4.95e−016*	*2.05e−003*	*6.02e−006*	*6.04e−019*	1.27e−031
Best method	FASSO	FASSO	FASSO	FASSO	SSO &FASSO	OLPSO-L
Algorithm	Rotated schwefel	Rotated rastrigin	Rotated Ackley	Rotated griewank	Shifted rosenbrock	Shifted rastrigin
GPSO	4.61e−003	60.02	1.93	1.80e−002	427.93	−223.18
LPSO	4.50e−003	53.36	1.55	1.68e−003	432.33	−234.95
VPSO	4.29e−003	71.05	2.56e−002	4.91e−003	501.29	−284.39
FIPS	4.41e−003	1.5−e002	3.16e−007	1.28e−008	424.83	−245.77
HPSO-TVAC	5.32e−003	52.90	9.29	9.26e−003	494.20	−318.33
DMS-PSO	4.04e−003	41.97	2.42e−014	1.02e−002	502.51	−303.17
CLPSO	4.39e−003	87.14	5.91e−005	7.96e−005	*403.07*	*−330*
OPSO	4.48e−003	63.78	1.49e−008	1.28e−003	2.45−e007	−284.11
APSO	*2.98e−003*	51.78	6.41e−012	2.25e−008	431.47	−314.21
OLPSO-G	4.00e−003	46.09	7.69e−015	1.68e−003	424.75	−328.57
OLPSO-L	3.13e−003	53.35	*4.28e−015*	4.19e−008	415.94	*−330*
SSO	3.11e−003	41.02	1.08e−014	1.93e−011	403.48	*−330*
FASSO	3.04e−003	*34.93*	8.16e−015	*9.04e−012*	411.26	*−330*
Best method	APSO	FASSO	OLPSO-L	FASSO	CLPSO	CLPSO&OLPSO-L &SSO&FASSO

The best results are italicized.

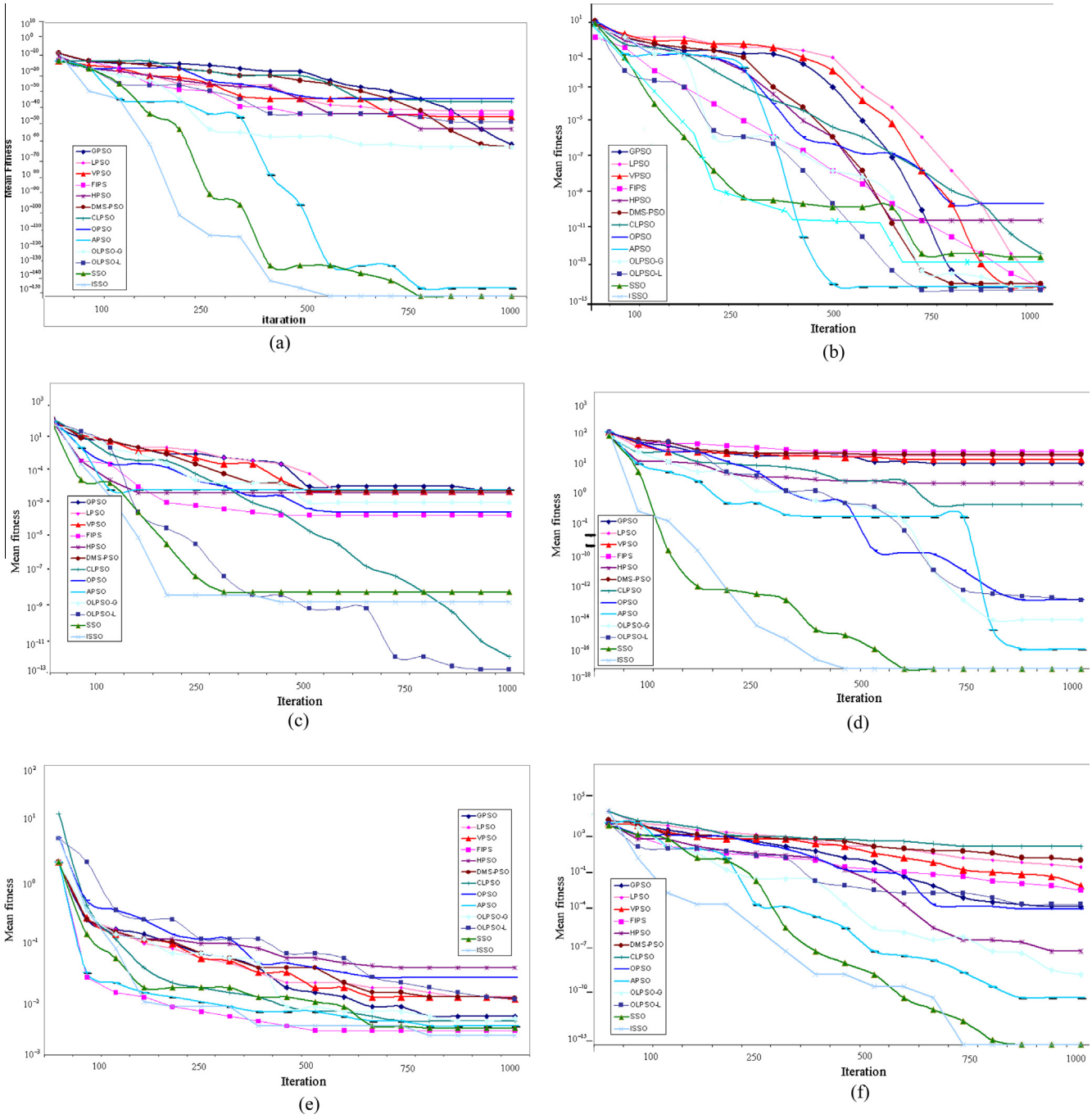

Figure 8 Convergence performance of the eleven different PSOs, SSO and FASSO on the 6 test functions. (a) Sphere. (b) Ackley. (c) Griewank. (d) N_Rastrigin. (e) Quadric noise. (f) Quadric.

moving particles and reached more optimal points at a remarkable speed, but this behavior was not consistent and the previous tragedy happened again and the APSO and OLPSO-L performed better than the proposed method. As for the Quadric and N-Rastrigin functions, the proposed method exhibited an intelligent behavior and, through preventing from falling into local points, could successfully find the absolute optimal point.

One of the most complicated functions used for testing optimization methods is the Rotate functions four of which were employed in this research [20]. As shown in Table 3 and in Fig. 9, the proposed method showed good behaviors in the two functions of Rotated Rastrigin and Rotated Griewank and achieved the best results. The results it obtained in the other two Rotate functions were close to the best ones achieved.

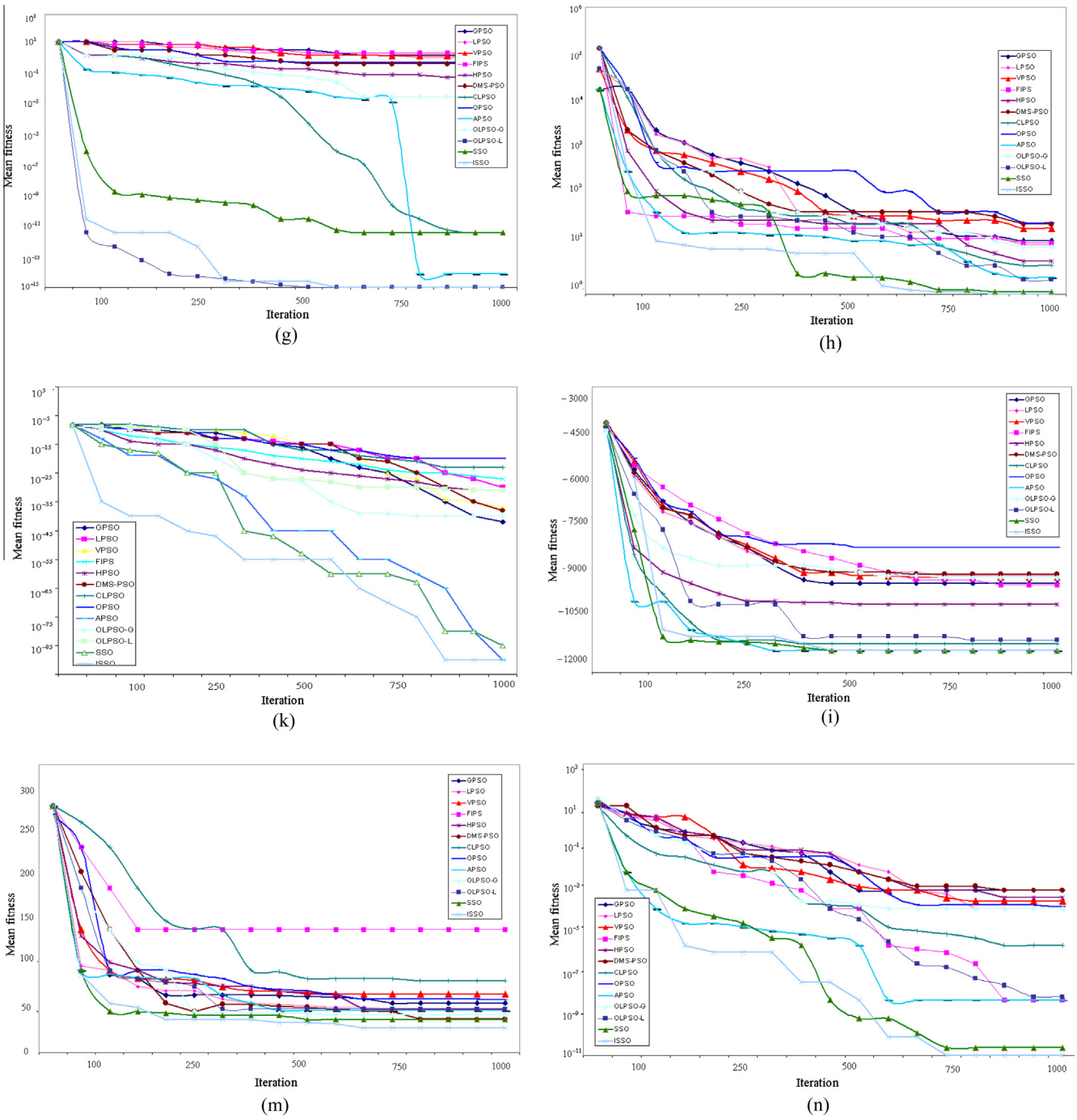

Figure 9 Convergence performance of the eleven different PSOs, SSO and FASSO on the 6 test functions. (g) Rastrigin. (h) Rosenbrock. (k) Schwefel's P2.22. (i) Schwefel. (m) Rotated Rastrigin. (n) Rotated Griewank.

4. Conclusion

After investigating and analyzing the SSO, the conclusion was drawn that this method does not behave well in some of the functions. The main reasons for this include the lack of an inertia weight to control the speeds of particles so that particles sometimes pass by the optimal points. Moreover, the acceleration coefficients do not reflex appropriately to the environment; in other words, they lack the required flexibility and hence, no suitable balance is established between the local and global searches. Two suggestions were put forward to solve these shortcomings. The first is to introduce an adaptive inertia weight that decreases parabolically with time. The second is to use fuzzy control in optimizing the acceleration coefficients (and here, the two fuzzy inference systems played the central role). The proposed method had a high speed of convergence, could avoid premature convergence around local optimal points, and had substantial flexibility in complex environments. It cannot be claimed that the FASSO is the best optimization method; however, it is hoped that this method

will be improved with the help of other researchers and experts.

References

[1] Beni G, Wang J. Swarm intelligence in cellular robotic systems. In: Proceed. NATO advanced workshop on robots and biological systems, Tuscany, Italy, June 26–30; 1989.

[2] Dorigo M, Maniezzo V, Colorni A. The ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B 1996;26(1):29–41.

[3] Dorigo M, Gambardella LM. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evol Comput 1997;1(1):53–66.

[4] Dorigo M, Stützle T. Ant colony optimization. Cambridge: MIT Press; 2004.

[5] Kennedy J, Eberhart RC. Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks. Piscataway: IEEE Press; 1995. p. 42–8.

[6] Clerc M. Particle swarm optimization. London: ISTE Ltd.; 2007.

[7] Poli R, Kennedy J, Blackwell T. Particle swarm optimization: an overview. Swarm Intell 2007;1(1):33–57.

[8] Wang B, Jin XP, Cheng B. Lion pride optimizer: an optimization algorithm inspired by lion pride behavior, Berlin Heidelberg; Science China Press and Springer Verlog.

[9] He S, Wu QH, Saunders JR. A novel group search optimizer inspired by animal behavior ecology. In: Proc. IEEE congr. evol. comput.. Vancouver, BC: Sheraton Vancouver Wall Center; 2006. p. 1272–8.

[10] He S, Wu QH, Saunders JR. Group search optimizer: an optimization algorithm inspired by animal searching behavior. IEEE Trans Evolutionary Comput 2009;13(5).

[11] Passino KM. Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst Mag 2002:52–67.

[12] Li XL. A new intelligent optimization-artificial fish swarm algorithm. PhD thesis. China: Zhejiang University; 2003.

[13] Jiang MY, Yuan DF. Artificial fish swarm algorithm and its applications. In: Proc. of the international conference on sensing, computing and automation, (ICSCA'2006). Chongqing, China; 8–11 May 2006, p. 1782–7.

[14] Xiao JM, Zheng XM, Wang XH. A modified artificial fish-swarm algorithm. In: Proc. of the IEEE 6th world congress on intelligent control and automation, (WCICA'2006). Dalian, China; 21–23 June 2006. p. 3456–60.

[15] Krishnanand KN, Ghose D. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. In: Proceedings of IEEE swarm intelligence symposium. Piscataway: IEEE Press; 2005. p. 84–91.

[16] Krishnanand KN, Ghose D. Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications. Multiagent Grid Syst 2006;2(3):209–22.

[17] Krishnanand KN, Ghose D. Theoretical foundations for multiple rendezvous of glowworm inspired mobile agents with variable local-decision domains. In: Proceedings of American control conference. Piscataway: IEEE Press; 2006. p. 3588–93.

[18] Krishnanand KN, Ghose D. Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. Swarm Intelligence 2009;3:87–124. http://dx.doi.org/10.1007/s11721-008-0021-5.

[19] Neshat Mehdi, Sepidnam Ghodrat, Sargolzaei Mehdi. Swallow swarm optimization algorithm: a new method to optimization. Neural Comput Appl 2013;23(2):429–54.

[20] Salomon R. Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions. BioSystems 1996;39:263–78.

[21] Feng CS, Cong S, Feng XY. A new adaptive inertia weight strategy in particle swarm optimization. In: IEEE Congress on evolutionary computation, CEC 2007; 2007. p. 4186–90.

[22] Malik RF, Rahman TA, Hashim SZM, Ngah R. New particle swarm optimizer with sigmoid increasing inertia weight. Int J Comput Sci Security (IJCSS) 2007;1(2):35.

[23] Xin J, Chen G, Hai Y. A particle swarm optimizer with multistage linearly-decreasing inertia weight. Computational sciences and optimization 2009. CSO 2009. International joint conference on, vol. 1. IEEE; 2009. p. 505–8.

[24] Li HR, Gao YL. Particle swarm optimization algorithm with exponent decreasing inertia weight and stochastic mutation. 2009 Second international conference on information and computing science. IEEE; 2009. p. 66–9.

[25] Bansal JC, Singh PK, Saraswat M, Verma A, Jadon SS, Abraham A. Inertia weight strategies in particle swarm optimization, In: IEEE third world congress on nature and biologically inspired computing (NaBIC); 2011. p. 633–40.

[26] Shi Y, Eberhart RC. A modified particle swarm optimizer. In: Proc. IEEE world congr. comput. intell.; 1998. p. 69–73.

[27] Kennedy J, Mendes R. Population structure and particle swarm performance, In: Proc. IEEE congr. evol. comput.; Honolulu, HI; 2002. p. 1671–6.

[28] Kennedy J, Mendes R. Neighborhood topologies in fully informed and best-of-neighborhood particle swarms. IEEE Trans Syst, Man, Cyber – Part C: Appl Rev 2006;36(4):515–9.

[29] Mendes R, Kennedy J, Neves J. The fully informed particle swarm: simpler, maybe better. IEEE Trans Evolution Comput 2004;8(3):204–10.

[30] Ratnaweera A, Halgamuge S, Watson H. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evolution Comput 2004;8(3):240–55.

[31] Liang JJ, Suganthan PN. Dynamic multi-swarm particle swarm optimizer with local search. In: Proc. IEEE congr. evol. comput.; 2005. p. 522–8.

[32] Liang JJ, Qin AK, Suganthan PN, Baskar S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evolution Comput 2006;10():281–95.

[33] Ho S-Y, Lin H-S, Liauh W-H, Ho S-J. OPSO: orthogonal particle swarm optimization and its application to task assignment problems. IEEE Trans Syst, Man, Cybernetics – Part A 2008;38():288–98.

[34] Zhan Zhi-Hui, Zhang Jun, Li Yun, Hung Chung Henry Shu. Adaptive particle swarm optimization. IEEE Trans Syst, Man, Cyber, Part B: Cyber 2009;39(6):1362–81.

[35] Zhan Zhi-Hui, Zhang Jun, Li Yun, Shi Yu-Hui. Orthogonal learning particle swarm optimization, IEEE Trans Evolution Comput, 99. p. 1. ISSN 1089–778X.

[37] Karaboga D. An idea based on honey bee swarm for numerical optimization. Technical report. Computer engineering department. Engineering faculty. Erciyes University; 2005.

[38] Hedayatzadeh R, Akhavan Salmassi F, Keshtgari M. Termite colony optimization: a novel approach for optimizing continuous problems. IEEE ICEE 2010.

[39] Nickabadi A, Ebadzadeh MM, Safabakhsh R. A novel particle swarm optimization algorithm with adaptive inertia weight. Appl Soft Comput 2011;11(4):3658–70.

[40] Bansal JC, Singh PK, Saraswat M, Verma A, Jadon SS, Abraham A. Inertia weight strategies in particle swarm optimization. In: IEEE 2011 third world congress on nature and biologically inspired computing, vol. 640; 2011. p. 633.

[41] Liliana Dewi Yanti, Widyanto M Rahmat. Particle swarm optimization with fuzzy adaptive acceleration for human object detection. Int J Video Image Process Network Sec 2011;11(1):11.

[42] Kaveh A, Bakhshpoori T, Afshari E. An efficient hybrid particle swarm and swallow swarm optimization algorithm. Comput Struct 2014:40–59, <http://www.sciencedirect.com/science/article/pii/S0045794914001564>, alikaveh@iust.ac.ir.
[43] Neshat Mehdi. FAIPSO: fuzzy adaptive informed particle swarm optimization. Neural Comput Appl 2013;23(1):95–116.
[44] Neshat Mehdi, Rezaei Masoud. AIPSO: adaptive informed particle swarm optimization. In: Intelligent systems (IS), 2010 5th IEEE international conference. p. 438–43.
[45] Neshat M, Sargolzaei M, Masoumi A, Najaran A. A new kind of PSO: predator particle swarm optimization. Int J Smart Sens Intell Syst 2012;5:521–39.