

Swallow swarm optimization algorithm: a new method to optimization

Mehdi Neshat · Ghodrat Sepidnam · Mehdi Sargolzaei

Received: 7 November 2011 / Accepted: 2 April 2012 / Published online: 26 April 2012
© Springer-Verlag London Limited 2012

Abstract This paper presents an exposition of a new method of swarm intelligence-based algorithm for optimization. Modeling swallow swarm movement and their other behavior, this optimization method represents a new optimization method. There are three kinds of particles in this method: explorer particles, aimless particles, and leader particles. Each particle has a personal feature but all of them have a central colony of flying. Each particle exhibits an intelligent behavior and, perpetually, explores its surroundings with an adaptive radius. The situations of neighbor particles, local leader, and public leader are considered, and a move is made then. Swallow swarm optimization algorithm has proved high efficiency, such as fast move in flat areas (areas that there is no hope to find food and, derivation is equal to zero), not getting stuck in local extremum points, high convergence speed, and intelligent participation in the different groups of particles. SSO algorithm has been tested by 19 benchmark functions. It achieved good results in multimodal, rotated and shifted functions. Results of this method have been compared to standard PSO, FSO algorithm, and ten different kinds of PSO.

Keywords Computational intelligence · Swallow swarm optimization (SSO) · Benchmark function · Fish swarm optimization · Particle swarm optimization

1 Introduction

Swarm intelligence (SI) [1] as observed in natural swarms is the result of actions that individual in the swarm perform exploiting local information. Usually, the swarm behavior serves to accomplish certain complex colony-level goals. Examples include group foraging by ants, division of labor among scouts and recruits in honeybee swarms, evading predators by fish schools, flocking of birds, and group-hunting as observed in canids, herons, and several cetaceans.

The decentralized decision-making mechanisms found in the above examples, and others in the natural world, offer an insight on how to design distributed algorithms that solve complex problems related to diverse fields, such as optimization, multi-agent decision making, and collective robotics. Ant colony optimization technique [2–4], particle swarm optimization algorithm [5–7], artificial fish swarm optimization algorithm [8–10], and glowworm swarm optimization algorithm [11–14] and several swarms based collective robotic algorithms, etc are different methods of swarm intelligence [15–17].

In this paper, we present a novel algorithm called swallow swarm optimization (SSO) for the simultaneous computation of multimodal functions. The algorithm shares some common features with particle swarm optimization (PSO) with fish swarm optimization (FSO), but with several significant differences. Swallows own high swarm intelligence. Their flying speed is high, and they are able to fly long distances in order to migrate from one

M. Neshat (✉) · G. Sepidnam · M. Sargolzaei
Department of Computer Engineering, Shirvan Branch,
Islamic Azad University, Shirvan, Iran
e-mail: Neshat_mehdi@ieee.org

G. Sepidnam
e-mail: sepidnam@ferdowsi.um.ac.ir

M. Sargolzaei
e-mail: sargolzaei@yahoo.com

point to another. They fly in great colonies. Flying collectively, they mislead hunters in dangerous positions. Swallow swarm life has many particular features that are more complicated and bewildering in comparison with fish schools and ant colonies. Consequently, it has been appointed as the subject of research and algorithm simulation.

The second section is about the method. Third section is about the benchmark function. Forth section is about experimental results and examining the diverse states of PSO and FSO and then comparing them to proposed method.

2 Method

Using swarm intelligence in optimization problems has become prevalent in recent decades. Methods like PSO, GA, ACO, FSO, GSO and many compound methods which by combining these methods are trying to refine the diverse engineering problems more and more. Some examples of what have been done which are combinations of different methods of SI are shown in Table 1. Certainly, fuzzy logic plays an important role in optimization [18].

2.1 PSO and its developments

2.1.1 PSO structure

The initial ideas on particle swarms of Kennedy (a social psychologist) and Eberhart (an electrical engineer) were

basically aimed at producing computational intelligence by exploiting simple analogs of social interaction, rather than purely individual cognitive abilities. The first simulations [5] were influenced by Heppner and Grenander's work [19] and involved analogs of bird flocks searching for corn. These soon developed [5, 20, 21] into a powerful optimization method—PSO [22].

In PSO, a swarm of particles are represented as potential solutions, and each particle i is associated with two vectors, that is, the velocity vector $V_i = [v_i^1, v_i^2, \dots, v_i^D]$ and the position vector $X_i = [x_i^1, x_i^2, \dots, x_i^D]$ where D stands for the dimensions of the solution space. The velocity and the position of each particle are initialized by random vectors within the corresponding ranges. During the evolutionary process, the velocity and position of particle i on dimension d are updated as:

$$v_i^d = wv_i^d + c_1 \text{rand}_1^d (p\text{Best}_i^d - x_i^d) + c_2 \text{rand}_2^d (n\text{Best}^d - x_i^d) \quad (1)$$

$$x_i^d = x_i^d + v_i^d \quad (2)$$

where w is the inertia weight [23], c_1 and c_2 are the acceleration coefficients [20], and rand_1^d and rand_2^d are two uniformly distributed random numbers independently generated within $[0, 1]$ for the d th dimension [5]. In (1), $p\text{Best}_i$ is the position with the best fitness found so far for the i th particle, and $n\text{Best}$ is the best position in the neighborhood. In the literature, instead of using $n\text{Best}$, $g\text{Best}$ may be used in the global-version PSO, whereas $l\text{Best}$ may be used in the local-version PSO (LPSO). An

Table 1 Review to compound methods of new optimization

Method	Authors	Paper	Publisher
PSO-GA [79]	Esmín, A. A. A. Lambert-Torres, G. Alvarenga, G. B. UFLA, Brazil	Hybrid evolutionary algorithm based on PSO and GA mutation	Sixth International Conference on Hybrid Intelligent Systems, 2006. HIS '06
PSO-GA [80]	Matthew Settles and Terence Soule	Breeding swarms: a GA/PSO hybrid	In GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation (2005), pp. 161–168
ACO-PSO [81]	Yan Meng and Qlorundamilqila Kazeem	A hybrid ACO/PSO control algorithm for distributed swarm robots	Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)
PSO-ACO [82]	D. Gómez-Cabrero, D. N. Ranasinghe	Fine-tuning the ant colony system algorithm through particle swarm optimization	Proceedings of the International Conference on Information and Automation, 2005
PSO-FSO [83]	Huadong Chen, Shuzong Wang, Jingxi Li, Yunfan Li	A hybrid of artificial fish swarm algorithm and particle swarm optimization for feedforward neural network training	2007 International Conference on Intelligent Systems and Knowledge Engineering (ISKE 2007)
ACO-FSO [84]	Hongyan Shi, Zhaoyu Bei	Application of improved ant colony algorithm	Fourth International Conference on Natural Computation, 2008. ICNC '08
ACO-FSO [85]	Hong-yan Shi, Zhao-yu Bei	A mixed ant colony algorithm for function optimization	Proceedings of the 21st annual international conference on Chinese control and decision IEEE Press Piscataway, NJ, USA 3919–3923, 2009

user-specified parameter $V_{\max}^d \in \mathfrak{R}^+$ is applied to clamp the maximum velocity of each particle on the d th dimension. Thus, if the magnitude of the updated velocity $|v_i^d|$ exceeds V_{\max}^d , then v_i^d is assigned the value $sign(v_i^d)V_{\max}^d$.

2.1.2 New developments of the PSO

Given its simple concept and efficiency, the PSO has become a popular optimizer and has widely been applied in practical problem solving. Thus, theoretical studies and performance improvements of the algorithm have become important and attractive. Convergence analysis and stability studies have been reported by Clerc and Kennedy [24], Trelea [25], Yasuda et al. [26], Kadiramanathan et al. [27], and van den Bergh and Engelbrecht [28].

The inertia weight w in (1) was introduced by Shi and Eberhart [23]. They proposed a w linearly decreasing with the iterative generations as:

$$w = w_{\max} - (w_{\max} - w_{\min})g/G \tag{3}$$

where g is the generation index representing the current number of evolutionary generations and G is a predefined maximum number of generations. Here, the maximal and minimal weights w_{\max} and w_{\min} are usually set to 0.9 and 0.4, respectively [23, 29]. In addition, a fuzzy adaptive w was proposed in [30], and a random version setting w to $0.5 + \text{random}(0, 1)/2$ was experimented in [31] for dynamic system optimization. As this random w has an expectation of 0.75, it has a similar idea as Clerc’s constriction factor [32, 33]. The constriction factor has been introduced into PSO for analyzing the convergence behavior, that is, by modifying (1) to

$$v_i^d = \chi[v_i^d + c_1 \text{rand}_1^d(p\text{Best}_i^d - x_i^d) + c_2 \text{rand}_2^d(n\text{Best}^d - x_i^d)] \tag{4}$$

where the constriction factor

$$\chi = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|} \tag{5}$$

is set to 0.729 with

$$\phi = c_1 + c_2 = 4.1 \tag{6}$$

where c_1 and c_2 are both set to 2.05 [33]; mathematically, the constriction factor is equivalent to the inertia weight, as Eberhart and Shi pointed out in [34]. Besides the inertia weight and the constriction factor, the acceleration coefficients c_1 and c_2 are also important parameters in PSO. In Kennedy’s two extreme cases [35], that is, the “social-only” model and the “cognitive-only” model, experiments have shown that both acceleration coefficients are essential

to the success of PSO. Kennedy and Eberhart suggested a fixed value of 2.0, and this configuration has been adopted by many other researchers. Suganthan [36] showed that using ad hoc values of c_1 and c_2 rather than a fixed value of 2.0 for different problems could yield better performance. Ratnaweera et al. [37] proposed a PSO algorithm with linearly time-varying acceleration coefficients (HPSO-TVAC), where a larger c_1 and a smaller c_2 were set at the beginning and were gradually reversed during the search. Among these three methods, the HPSO-TVAC shows the best overall performance [37]. This may be due to the time-varying c_1 and c_2 that can balance the global and local search abilities, which implies that adaptation of c_1 and c_2 can be promising in enhancing the PSO performance. Hence, this paper will further investigate the effects of c_1 and c_2 and develop an optimal adaptation strategy according to ESE.

Another active research trend in PSO is hybrid PSO, which combines PSO with other evolutionary paradigms. Angeline [38] first introduced into PSO a selection operation similar to that in a genetic algorithm (GA). Hybridization of GA and PSO has been used in [39] for recurrent artificial neural network design. In addition to the normal GA operators, for example, selection [38], crossover [40], and mutation [41], other techniques such as local search [42] and differential evolution [43] have been used to combine with PSO. Cooperative approach [44], self-organizing hierarchical technique [45], deflection, stretching, and repulsion techniques [46] have also been hybridized with traditional PSO to enhance performance. Inspired by biology, some researchers introduced niche [47, 48] and speciation [49] techniques into PSO to prevent the swarm from crowding too closely and to locate as many optimal solutions as possible and adaptive particle swarm optimization (APSO) that features better search efficiency than classical particle swarm optimization [50]. The orthogonal PSO (OPSO) reported in [41] uses an “intelligent move mechanism” (IMM) operation to generate two temporary positions, H and R , for each particle X , according to the cognitive learning and social learning components, respectively. Then, OED is performed on H and R to obtain the best position X^* for the next move, and then, the particle velocity is obtained by calculating the difference between the new position X^* and the current position X . Such an IMM was also used in [51] to orthogonally combine the cognitive learning and social learning components to form the next position, and the velocity was determined by the difference between the new position and the current position. The OED in [52] was used to help generate the initial population evenly. Different from previous work and to go steps further, in this paper, we use the OED to form an orthogonal learning

strategy, which discovers and preserves useful information in the personal best and the neighborhood best positions in order to construct a promising and efficient exemplar. This exemplar is used to guide the particle to fly toward the global optimal region. The OL (orthogonal learning) strategy is a generic operator and can be applied to any kind of topology structure. If the OL is used for the GPSO, then P_n is P_g . If it is used for the LPSO, then P_n is P_l . For either a global or a local version, when constructing the vector of P_o , if P_i is the same as P_n (e.g., for the globally best particle, P_i and P_g are identical vectors), the OED (orthogonal experimental design) makes no contribution. In such a case, OLPSO will randomly select another particle P_r and then construct P_o by using the information of P_i and P_r through the OED. Two OLPSO versions that based on a global topology (OLPSO-G) and a local topology (OLPSO-L) are simulated [53].

In addition to research on parameter control and auxiliary techniques, PSO topological structures are also widely studied. The LPSO with a ring topological structure and the von Neumann topological structure PSO (VPSO) have been proposed by Kennedy and Mendes [54, 55] to enhance the performance in solving multimodal problems. Further, dynamically changing neighborhood structures have been proposed by Suganthan [36], Hu and Eberhart [56], and Liang and Suganthan [57] to avoid the deficiencies of fixed neighborhoods. Moreover, in the “fully informed particle swarm” (FIPS) algorithm [58], the information of the entire neighborhood is used to guide the particles. The CLPSO in [59] lets the particle use different $pBests$ to update its flying on different dimensions for improved performance in multimodal applications.

2.2 Artificial fish swarm algorithm (AFSA)

A new evolutionary computation technique, artificial fish swarm algorithm (AFSA), was first proposed in 2002 [60]. The idea of AFSA is based on the simulation of the simplified natural social behavior of fish schooling and the swarming theory. AFSA possess similar attractive features of genetic algorithm (GA) such as independence from gradient information of the objective function, the ability to solve complex non-linear high-dimensional problems. Furthermore, they can achieve faster convergence speed and require few parameters to be adjusted. The AFSA does not possess the crossover and mutation processes used in GA, so it could be performed more easily. AFSA is also an optimizer based on population. The system is initialized firstly in a set of randomly generated potential solutions and then performs the search for the optimum one iteratively [61].

Artificial fish (AF) is a fictitious entity of true fish, which is used to carry on the analysis and explanation of problem, and can be realized by using animal ecology concept. With the aid of the object-oriented analytical method, we can regard the artificial fish as an entity encapsulated with one’s own data and a series of behaviors, which can accept amazing information of environment by sense organs, and do stimulant reaction by the control of tail and fin. The environment in which the artificial fish lives is mainly the solution space and the states of other artificial fish. Its next behavior depends on its current state and its environmental state (including the quality of the question solutions at present and the states of other companions), and it influences the environment via its own activities and other companions’ activities [62].

The AF realizes external perception by its vision shown in Fig. 1. X is the current state of an AF, $Visual$ is the visual distance, and X_v is the visual position at some moment. If the state at the visual position is better than the current state, it goes forward a step in this direction and arrives the X_{next} state; otherwise, it continues an inspecting tour in the vision. The greater number of inspecting tour the AF does, the more knowledge about overall states of the vision the AF obtains. Certainly, it does not need to travel throughout complex or infinite states, which is helpful to find the global optimum by allowing certain local optimum with some uncertainty.

Let $X = (x_1, x_2, \dots, x_n)$ and $X_v = (x_1^v, x_2^v, \dots, x_n^v)$, then this process can be expressed as follows:

$$x_i^v = x_i + Visual.rand(), \quad i \in (0, n] \quad (7)$$

$$X_{next} = X + \frac{X_v - X}{\|X_v - X\|} \cdot Step.rand(). \quad (8)$$

where $Rand()$ produces random numbers between 0 and 1, $Step$ is the step length, x_i is the optimizing variable, and n is

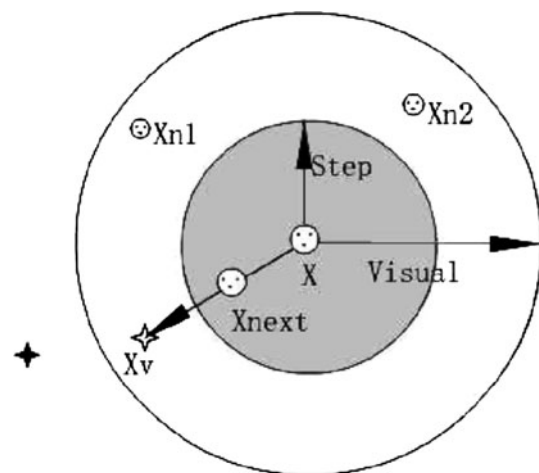


Fig. 1 Vision concept of the artificial fish

the number of variables. The AF model includes two parts (variables and functions). The variable X is the current position of the AF, $Step$ is the moving step length, $Visual$ represents the visual distance, try_number is the try number, and δ is the crowd factor ($0 < \delta < 1$). The functions include the behaviors of the AF: AF_Prey , AF_Swarm , AF_Follow , AF_Move , AF_Leap , and $AF_Evaluate$.

2.2.1 The basic functions of AFSA

Fish usually stay in the place with a lot of food, so we simulate the behaviors of fish based on this characteristic to find the global optimum, which is the basic idea of the AFSA. The basic behaviors of AF are defined [9, 10] as follows for maximum:

(1) AF_Prey : This is a basic biological behavior that tends to the food; generally, the fish perceives the concentration of food in water to determine the movement by vision or sense and then chooses the tendency. Behavior description: Let X_i be the AF current state and select a state X_j randomly in its visual distance, and Y is the food concentration (objective function value); the greater $Visual$ is, the more easily the AF finds the global extreme value and converges.

$$X_j = X_i + Visual.rand() \tag{9}$$

If $Y_i < Y_j$ in the maximum problem, it goes forward a step in this direction;

$$X_i^{(t+1)} = X_i^{(t)} + \frac{X_j - X_i^{(t)}}{\|X_j - X_i^{(t)}\|} .Step.rand(). \tag{10}$$

Otherwise, select a state X_j randomly again and judge whether it satisfies the forward condition. If it cannot satisfy after try_number times, it moves a step randomly. When the try_number is small in AF_Prey , the AF can swim randomly, which makes it flee from the local extreme value field.

$$X_i^{(t+1)} = X_i^{(t)} + Visual.rand() \tag{11}$$

(2) AF_Swarm : The fish will assemble in groups naturally in the moving process, which is a kind of living habits in order to guarantee the existence of the colony and avoid dangers. Behavior description: Let X_i be the AF current state, X_c be the center position, and n_f be the number of its companions in the current neighborhood ($d_{ij} < Visual$), and n is total fish number. If $Y_c > Y_i$ and $\frac{n_f}{n} < \delta$, which means that the companion center has more food (higher fitness function value) and is not very crowded, it goes forward a step to the companion center;

$$X_i^{(t+1)} = X_i^{(t)} + \frac{X_c - X_i^{(t)}}{\|X_c - X_i^{(t)}\|} .Step.rand(). \tag{12}$$

Otherwise, executes the preying behavior. The crowd factor limits the scale of swarms, and more AF only cluster at the optimal area, which ensures that AF move to optimum in a wide field.

(3) AF_Follow : In the moving process of the fish swarm, when a single fish or several ones find food, the neighborhood partners will trail and reach the food quickly. Behavior description: Let X_i be the AF current state, and it explores the companion X_j in the neighborhood ($d_{ij} < Visual$), which has the greatest Y_j . If $Y_j > Y_i$ and $\frac{n_f}{n} < \delta$, which means that the companion X_j state has higher food concentration (higher fitness function value) and the surroundings are not very crowded, it goes forward a step to the companion X_j ,

$$X_i^{(t+1)} = X_i^{(t)} + \frac{X_j - X_i^{(t)}}{\|X_j - X_i^{(t)}\|} .Step.rand(). \tag{13}$$

Otherwise, executes the preying behavior.

(4) AF_Move : Fish swim randomly in water; in fact, they are seeking food or companions in larger ranges.

Behavior description: Chooses a state at random in the vision; then it moves toward this state; in fact, it is a default behavior of AF_Prey .

$$X_i^{(t+1)} = X_i^{(t)} + Visual.rand() \tag{14}$$

(5) AF_Leap : Fish stop somewhere in water, every AF's behavior result will gradually be the same, the difference in objective values (food concentration, FC) becomes smaller within some iterations, it might fall into local extremum, change the parameters randomly to the still states for leaping out current state.

Behavior description: If the objective function is almost the same or difference in the objective functions is smaller than a proportion during the given ($m-n$) iterations, choose some fish randomly in the whole fish swarm and set parameters randomly to the selected AF. The β is a parameter or a function that can make some fish to have other abnormal actions (values), and eps is a smaller constant.

if $(BestFC(m) - BestFC(n)) < eps$

$$X_{some}^{(t+1)} = X_{some}^{(t)} + \beta .Visual.rand() \tag{15}$$

The detail behavior pseudo-code can be seen in [63]. AF_Swarm makes few fish confined in local extreme values move in the direction of a few fish tending to global extreme value, which results in AF fleeing from the local extreme values. AF_Follow accelerates AF moving to

better states and, at the same time, accelerates AF moving to the global extreme value field from the local extreme values.

2.3 Swallow swarm optimization (SSO)

2.3.1 Swallows natural life

There are eight different breeds of swallows. They have social life, migrate together, and find appropriate places for resting, breeding, and feeding. Swarm moving (swarm intelligence) keeps them immune against the attacks of other birds. A pair of swallows has to search a wide area in order to find food, while in collective living as soon as one swallow finds food, other birds of the group fly toward that area and enjoy it. Swallows are very intelligent birds. They fly quickly, and there is a strong interaction between members of groups [64]. They apply different sounds in different situations (warning, asking help, inviting to food, being ready to breeding) that the number of these sounds is more than other birds' and even signal of swallow's sound is used as a special model in signal processing [65]. Figure 2 is a swallow tree.

Many creatures have a social life and live in groups, including birds. A variety of birds in the small and large colonies have a social life, and each one has certain characteristics and social behaviors. By investigating these behaviors and getting inspired by them, various optimization algorithms have been presented, such as PSO. These swallows have unique characteristics and social behaviors that have attracted our attention.

Swallow is an insect-eating bird; 83 species of swallows have been so far identified [66]. Due to its high compatibility with the environment, this bird lives almost



Fig. 2 Tree male swallow photograph by Steve Berliner [88]

everywhere on Earth. This bird has special characteristics that make it distinct from the other birds. So many studies have been conducted on the life of various species of swallows and remarkable results have been obtained, which will be further discussed:

2.3.1.1 Immigration “Swallows” annually travel 17,000 km and migrate from a continent to another continent. Swallows migrate in very large groups of even hundred thousand [67]. The very social life in large groups indicates high swarm intelligence of these birds.

2.3.1.2 High-speed flying Swallows keep a record of speed among the other migrants, so that they travel 4,000 km per 24 h, that is, with a speed of 170 km/h per hour. This feature can be very effective in the particles' convergence speed and in solving the optimization problems in the least time.

2.3.1.3 Skilled hunters Swallows have adapted to hunting insects on the wing by developing a slender streamlined body and long-pointed wings, which allow great maneuverability and endurance, as well as frequent periods of gliding. Their body shape allows for very efficient flight, which costs 50–75 % less for swallows than equivalent passerines of the same size. Swallows usually forage at around 30–40 km/h. Swallows are excellent flyers and use these skills to feed and attract a mate.

The swallows generally forage for prey that is on the wing, but they will on occasion snap prey off branches or on the ground. The flight may be fast and involve a rapid succession of turns and banks when actively chasing fast-moving prey; less agile prey may be caught with a slower more leisurely flight that includes flying in circles and bursts of flapping mixed with gliding. Where several species of swallow feed together, they will be separated into different niches based on height off the ground: some species feeding closer to the ground and others feeding at higher levels. Similar separation occurs where feeding overlaps with swifts. Niche separation may also occur with the size of prey chosen.

2.3.1.4 Different calls Swallows are able to produce many different calls or songs, which are used to express excitement, to communicate with others of the same species, during courtship, or as an alarm when a predator is in the area. The songs of males are related to the body condition of the bird and are presumably used by females to judge the physical condition and suitability for mating of males. Begging calls are used by the young when soliciting food from their parents. The typical song of swallows is a simple, sometimes musical twittering [68].

Swallows are regularly correlated through various sounds, and if one of them does not find the food source, it immediately calls the other birds in the colony. These special sounds help them to have a better social life.

2.3.1.5 Information centers Cliff swallow colonies function as “information centers” in which individuals unsuccessful at finding food locate other individuals that have found food and follow them to a food source [69]. Advantages associated with information sharing on the whereabouts of food are substantial and probably represent a major reason why cliff swallows live in colonies. He also discovered that cliff swallows represent one of the few birds (and indeed non-human vertebrates) that actively communicate the presence of food to others by giving distinct signals (calls) used only in that context [70]. The evolution of such information sharing is perplexing, because the typical beneficiaries of calling are individuals unrelated to the caller.

2.3.1.6 Floating swallow When migrating, a few swallows always fly out of the colony and in the first glance; it might look like that they disturb the colony order. But these swallows that are generally young (newly matured) play an important role in the colony. *First*, the swallows have the chance to find food outside the internal areas of the colony and call the other swallows as soon as they find food. *Second*, if a hunting bird intends to attack the swallows, these floating swallows quickly notice and inform the other members of the colony. These swallows fly between the colonies and can change their colonies. This behavior has been used in this study. These particles can increase the chance to find the optimum points, and if the other particles converge on a local optimum by mistake, these particles strengthen the chance to find better points by their random and independent movement. In this paper, these particles have been called aimless particle.

2.3.1.7 The interest in social life Swallows are one of the most important species of birds that prefer living in colonies to individual life. Sometimes, thousands of swallows can be seen as a colony in the sky. However, large colonies are composed of several smaller colonies. The number of birds in the group has a critical role in more successful reproduction, fighting predators, and a better search for food [71, 72]. There is a factor in the group life of swallows called *social stimulation*, which is an important influence [72]. Swallows’ colony size can have a direct impact on the hormones level in their bodies. The larger the colony is, the higher the hormone levels will be and the more successful swallows’ life and reproduction will be [73]. During 1,000 years of

evolution, this bird has obtained a high *swarm intelligence* and has a successful social life.

2.3.1.8 Leaders Each colony is divided into several subcolonies that have side-by-side nests and live in a site. Each colony has a leader that is commonly an experienced bird. If the suitability of the group leader is reduced for any reason, another bird immediately supersedes it. Swallows always follow the leader, provided they have the necessary competencies. In this study, two types of leaders are used: *Local leaders* that conduct the internal colonies and show a local optimum point and *Head Leader* that is responsible for the leadership of the entire colony and shows the public optimum point.

2.3.1.9 Escaping from predators Because of their small body, swallows are good prey for most birds. Two of their strategies are *synchronous flying* and *namely safety in numbers*. Swallows’ intelligent behaviors against hunting birds are very interesting. Swallows can be highly adaptable to the different environments. These adaptive behaviors have been intelligent and can be well seen in feeding and nest-building architecture. Swallows are very successful in search of food and have specific strategies for finding food which are very complex [74]. Many behaviors of the swallows have remained unknown, and it is very difficult to implement all behaviors of this bird to produce a working collective intelligence algorithm, because the desired algorithm will have a high time complexity and will not have the necessary optimality, due to the high complexity of these behaviors. In this paper, only the behaviors of 8, 6, 2, and 1 have been used and good results have been obtained.

2.3.2 Algorithm SSO

Major idea of this new optimization algorithm is inspired by swallow swarm. There are three kinds of particles in this algorithm:

1. Explorer particle (e_i)
2. Aimless particle (o_i)
3. Leader particle (l_i)

These particles move parallel to each other and always are in interaction. Each particle in colony (each colony can be consisted of some subcolonies) is responsible for something that through doing it guides the colony toward a better situation.

2.3.2.1 Explorer particle These particles encompass the major population of colony. Their primary responsibility is to explore in problem space. With just arriving at an

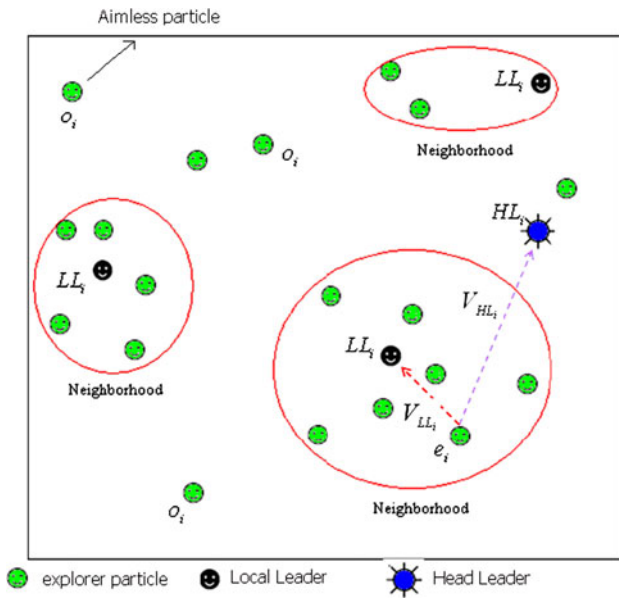


Fig. 3 Kinds of particles and how explorer particles move

extreme point (swallow), using a special sound guides the group toward there, and if this place is the best one in problem space, this particle plays role as a Head Leader (HL_i). But if the particle is in a good (not the best) situation in comparison with its neighboring particles, it is chosen as a local leader LL_i ; otherwise, each particle e_i regarding V_{HL_i} (velocity vector of particle toward HL), V_{LL_i} (velocity vector of particle toward LL), and competence of resultant of these two vector makes a random move. Figure 3 shows how a particle moves in problem space.

$$V_{HL_{i+1}} = V_{HL_i} + \alpha_{HL}rand()(e_{best} - e_i) + \beta_{HL}rand()(HL_i - e_i) \tag{16}$$

$$\alpha_{HL} = \{if(e_i = 0 || e_{best} = 0) \rightarrow 1.5\} \tag{17}$$

$$\alpha_{HL} = \begin{cases} if(e_i < e_{best}) \& \& (e_i < HL_i) \rightarrow \frac{rand().e_i}{e_i \cdot e_{best}} \\ if(e_i < e_{best}) \& \& (e_i > HL_i) \rightarrow \frac{2rand().e_{best}}{1/(2 \cdot e_i)} \\ if(e_i > e_{best}) \rightarrow \frac{e_{best}}{1/(2 \cdot rand())} \end{cases} \tag{18}$$

$$\beta_{HL} = \{if(e_i = 0 || e_{best} = 0) \rightarrow 1.5\} \tag{19}$$

$$\beta_{HL} = \begin{cases} if(e_i < e_{best}) \& \& (e_i < HL_i) \rightarrow \frac{rand().e_i}{e_i \cdot HL_i} \\ if(e_i < e_{best}) \& \& (e_i > HL_i) \rightarrow \frac{2rand().HL_i}{1/(2 \cdot e_i)} \\ if(e_i > e_{best}) \rightarrow \frac{HL_i}{1/(2 \cdot rand())} \end{cases} \tag{20}$$

Vector V_{HL_i} has a significant effect on explorer particle behavior. e_i is the particle current position in problem

space. e_{best} is the best position that particle remembers from the beginning up to now. HL_i is a leader particle that has the best possible response in current position. α_{HL} and β_{HL} are the control of acceleration coefficients that are defined adaptively. These two parameters change during the particle movements, and this change depends on the particle’s position. If the particle is a minimum point (minimizing problem) and is in a better position than the e_{best} and HL_i , probability of being a global minimum for that particle should be considered and control coefficients estimate a small amount to decrease the particle movement to the least. If the particle is in a better situation than e_{best} but is in worse situation than HL_i , it should move toward HL_i with an average amount. If the particle position is worse than e_{best} , consequently it is worse than HL_i , too, so it can move toward HL_i with a larger amount. Remember that the vector of V_{LL_i} affects this movement.

$$V_{LL_{i+1}} = V_{LL_i} + \alpha_{LL}rand()(e_{best} - e_i) + \beta_{LL}rand()(LL_i - e_i) \tag{21}$$

$$\alpha_{LL} = \{if(e_i = 0 || e_{best} = 0) \rightarrow 2\} \tag{22}$$

$$\alpha_{LL} = \begin{cases} if(e_i < e_{best}) \& \& (e_i < LL_i) \rightarrow \frac{rand().e_i}{e_i \cdot e_{best}} \\ if(e_i < e_{best}) \& \& (e_i > LL_i) \rightarrow \frac{2rand().e_{best}}{1/(2 \cdot e_i)} \\ if(e_i > e_{best}) \rightarrow \frac{e_{best}}{1/(2 \cdot rand())} \end{cases} \tag{23}$$

$$\beta_{LL} = \{if(e_i = 0 || e_{best} = 0) \rightarrow 2\} \tag{24}$$

$$\beta_{LL} = \begin{cases} if(e_i < e_{best}) \& \& (e_i < LL_i) \rightarrow \frac{rand().e_i}{e_i \cdot LL_i} \\ if(e_i < e_{best}) \& \& (e_i > LL_i) \rightarrow \frac{2rand().LL_i}{1/(2 \cdot e_i)} \\ if(e_i > e_{best}) \rightarrow \frac{LL_i}{1/(2 \cdot rand())} \end{cases} \tag{25}$$

$$V_{i+1} = V_{HL_{i+1}} + V_{LL_{i+1}} \tag{26}$$

$$e_{i+1} = e_i + V_{i+1} \tag{27}$$

Every particle e_i utilizes nearest particle LL_i in order to compute the vector of V_{LL_i} .

2.3.2.2 Aimless particle These particles in the beginning of exploring the situation do not have a good position in comparison with other particles, and the amount of their $f(o_i)$ is bad. These particles, after being recognized, are differentiated from explorer particles e_i , so a new responsibility in group is defined for them (o_i). Their duty is an exploratory and random search. They start moving randomly and do not have anything to do with the position of HL_i and LL_i . They are swallows that explore remote areas as the scout of colony and inform the group if they find a

good point. In a lot of optimization problems because of inappropriate distribution of particles in position space, the optimum response is kept hidden from the group's eyes and the group converges in a local optimum. This is the greatest difficulty with optimization problems (early convergence in local optimum points). Particles o_i , apparently, may have an aimless and useless behavior but examine the probability of neglecting the global optimum response and go around the diverse surrounding points with their long jumps and examine the situation of optimization. The particle o_i compares its position with the local optimum points LL_i and HL_i .

If this particle finds an optimum point while it is searching, it will replace its position with the nearest explorer particle e_i and then keep searching.

$$o_{i+1} = o_i + \left[\text{rand}(\{-1, 1\}) * \frac{\text{rand}(\text{min}_s, \text{max}_s)}{1 + \text{rand}()} \right] \quad (28)$$

New position of each particle of o_i is equal to its previous position plus a random amount between the minimum and the maximum of position space, divided by an amount between one and two. The division answer is added to or subtracted from the previous position of particle o_i randomly.

For example, the range of the function Rosenbrock has been defined between $(-50, 50)$. If the function $\text{rand}(\text{min}, \text{max})$ produces a random number (25) and the function $\text{rand}()$ produces 0.5, the fraction result would be 12.5. Now this amount may be added to or subtracted from the position of o_i . This will increase the chance of examining the different areas of environment.

2.3.2.3 Leader particle There are particles in SSO algorithm named *Leader*. These particles have the best amount of $f(l_i)$ in the beginning of position space searching. Their place and their amount may change in each level. There is just one leader particle in PSO method (g_{best}), while in this new method there may be n_l leader particle. These particles may be distributed or gathered in space. The best leader is named *Leader Head*, which is recognized as the major leader in colony, also there are some particles named *Local Leader*. They are candidates for quite good answers which we preserve. In real world of swallows, each thousands member colony is divided into a number of subcolonies. These subcolonies have a local leader; though, this leadership may be changed repeatedly by other wiser and stronger swallows. In the population of swallows, leader is a bird which is in a better location which is near to food and a resting place. The duty of this leader is to guide other members of colony to this area. This issue is simulated in SSO algorithm. In each repetition, leader particles whether head or local may be changed or an aimless particle may discover an area that is the best response of problem up to

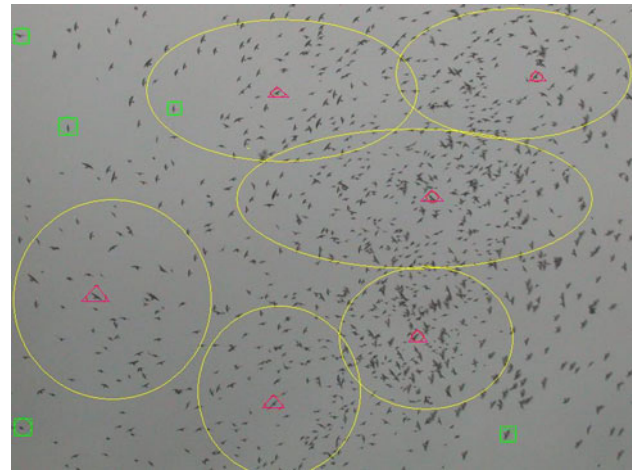


Fig. 4 Swallow swarm flying. Circles are subcolonies, and triangles are local leaders (choosing that swallow is surely a hard job but that swallow is generally in the center of subcolony and there is a high density around it. Never forget that this swallow may be changed every second. Squares are aimless swallows o_i

then; consequently, this swallows acts as a leader. Figure 4 shows an example of swallow swarm flying.

Real boundaries between subcolonies may never be marked because swallow movements happen in high speed and high dynamic. The diagram of swallows and their numbers changes according to the size of subcolonies.

All of these three particles (e_i , o_i , and l_i) interact with each other continuously, and each particle (swallow) can play each of these three roles. During searching period, these particles may frequently change their role but primary goal (finding optimum point) is such a more important task.

2.3.2.4 Pseudocode SSO Programming the SSO algorithm is easy. In this research, MATLAB software is used for above simulation (Figs. 5, 6).

Swallow swarm optimization algorithm:

1. Initialize population (random all particle e_i)
2. Iter = 1
3. While (iter < max_iter)
4. For every particle (e_i) Calculate $f(e_i)$
5. Sort $f(e_1, e_2, \dots, e_n) \rightarrow \text{min to max}$
6. All particles $e_i = f(e_i)$
7. HL = e_{min}
8. For ($j = 2$ to $j < m$) $LL_j = e_j$
9. For ($j = 1$ to $j < b$) $o_j = e_{n-j+1}$
10. $i = 1$
11. While ($k < \text{iter}$)
12. While ($j < N$)
13. if ($e_{\text{best}} > e_i$) $e_{\text{best}} = e_i$
14. While ($i < N$)

continued

15. Search (nearest LL_i to e_i)
16. $\alpha_{HL} = \{\text{if}(e_i = 0|e_{best} = 0) \rightarrow 1.5\}$
17. $\alpha_{HL} = \begin{cases} \text{if}(e_i < e_{best}) \& \& (e_i < HL_i) \rightarrow \frac{rand().e_i}{e_i \cdot e_{best}} & e_i, e_{best} \neq 0 \\ \text{if}(e_i < e_{best}) \& \& (e_i > HL_i) \rightarrow \frac{2rand().e_{best}}{1/(2 \cdot e_i)} & e_i \neq 0 \\ \text{if}(e_i > e_{best}) \rightarrow \frac{e_{best}}{1/(2 \cdot rand())} \end{cases}$
18. $\beta_{HL} = \{\text{if}(e_i = 0|e_{best} = 0) \rightarrow 1.5\}$
19. $\beta_{HL} = \begin{cases} \text{if}(e_i < e_{best}) \& \& (e_i < HL_i) \rightarrow \frac{rand().e_i}{e_i \cdot HL_i} & e_i, HL_i \neq 0 \\ \text{if}(e_i < e_{best}) \& \& (e_i > HL_i) \rightarrow \frac{2rand().HL_i}{1/(2 \cdot e_i)} & e_i \neq 0 \\ \text{if}(e_i > e_{best}) \rightarrow \frac{HL_i}{1/(2 \cdot rand())} \end{cases}$
20. $V_{HL_{i+1}} = V_{HL_i} + \alpha_{HL}rand()(e_{best} - e_i) + \beta_{HL}rand()(HL_i - e_i)$
21. $\alpha_{LL} = \{\text{if}(e_i = 0|e_{best} = 0) \rightarrow 2\}$
22. $\alpha_{LL} = \begin{cases} \text{if}(e_i < e_{best}) \& \& (e_i < LL_i) \rightarrow \frac{rand().e_i}{e_i \cdot e_{best}} & e_i, e_{best} \neq 0 \\ \text{if}(e_i < e_{best}) \& \& (e_i > LL_i) \rightarrow \frac{2rand().e_{best}}{1/(2 \cdot e_i)} & e_i \neq 0 \\ \text{if}(e_i > e_{best}) \rightarrow \frac{e_{best}}{1/(2 \cdot rand())} \end{cases}$
23. $\beta_{LL} = \{\text{if}(e_i = 0|e_{best} = 0) \rightarrow 2\}$
24. $\beta_{LL} = \begin{cases} \text{if}(e_i < e_{best}) \& \& (e_i < LL_i) \rightarrow \frac{rand().e_i}{e_i \cdot LL_i} & e_i, LL_i \neq 0 \\ \text{if}(e_i < e_{best}) \& \& (e_i > LL_i) \rightarrow \frac{2rand().LL_i}{1/(2 \cdot e_i)} & e_i \neq 0 \\ \text{if}(e_i > e_{best}) \rightarrow \frac{LL_i}{1/(2 \cdot rand())} \end{cases}$
25. $V_{LL_{i+1}} = V_{LL_i} + \alpha_{LL}rand()(e_{best} - e_i) + \beta_{LL}rand()(LL_i - e_i)$
26. $V_{i+1} = V_{HL_{i+1}} + V_{LL_{i+1}}$
27. $e_{i+1} = e_i + V_{i+1}$
28. $o_{i+1} = o_i + rand(\{-1, 1\}) * \frac{rand(\min_e, \max_e)}{1+rand()}$
29. if $(f(o_{i+1}) > f(HL_i))$ $e_{nearest} = O_{i+1} \rightarrow$ go to 30
30. While $(k_o \leq b)$
31. While $(l \leq n_i)$
32. if $(f(o_{k_o}) > f(LL_i))$
33. $e_{nearest} = O_{i+1}$
34. Loop// $i < N$
35. Loop//iter $< \text{max_iter}$
36. End.

e_i is that $f(e_i)$ in above algorithm (Table 2).

3 Benchmark functions

To test the ability of the proposed algorithm and to compare it to conventional Variant PSO and FSO, some benchmarks are considered. Each of these functions tests the optimization algorithm in special conditions. So, weakness points of the optimization algorithm will be obvious. These functions are described in Table 3 and Fig. 7.

The first problem is the sphere function, and it is easy to solve. The second problem is the Rosenbrock function. It can be treated as a multimodal problem. It has a narrow



Fig. 5 Swallow swarm flying

valley from the perceived local optima to the global optimum. In the experiments below, we find that the algorithms that perform well on sphere function perform well on Rosenbrock function too. Ackley’s function has one narrow global optimum basin and many minor local optima. It is probably the easiest problem among the six as its local optima are shallow. Griewank’s function has a $\prod_{i=1}^D \cos(x_i/\sqrt{i})$ component causing linkages among variables, thereby making it difficult to reach the global optimum. An interesting phenomenon of Griewank’s function is that it is more difficult for lower dimensions than for higher dimensions [75].

Rastrigin’s function is a complex multimodal problem with a large number of local optima. When attempting to solve Rastrigin’s function, algorithms may easily fall into a local optimum. Hence, an algorithm capable of maintaining a larger diversity is likely to yield better results. Non-continuous Rastrigin’s function is constructed based on the Rastrigin’s function and has the same number of local optima as the continuous Rastrigin’s function. The complexity of Schwefel’s function is due to its deep local optima being far from the global optimum. It will be hard to find the global optimum if many particles fall into one of the deep local optima.

To rotate a function, first an orthogonal matrix M should be generated. The original variable is left multiplied by the orthogonal matrix M to get the new rotated variable $y = M*x$, and this variable y is used to calculate the fitness value f .

$$\text{if } M = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1D} \\ m_{21} & m_{22} & \dots & m_{2D} \\ \dots & \dots & \dots & \dots \\ m_{D1} & m_{D2} & \dots & m_{DD} \end{bmatrix} x = [x_1, x_2, \dots, x_D]^T \text{ and } y = [y_1, y_2, \dots, y_D]^T \quad (29)$$

Fig. 6 Swallow swarm flying and food searching



Table 2 Parameters of SSO algorithm

#	Parameter	Description
1	Iter	Algorithm iteration
2	e_i	Explorer particle
3	o_i	Aimless particle
4	HL	Head leader
5	LL_i	Local leader
6	n_l	The number of local leader
7	e_{best}	The best position the particle has ever had
8	$\alpha_{HL} \beta_{HL}$	Control parameters of convergence speed toward particle HL
9	$\alpha_{LL} \beta_{LL}$	Control parameters of convergence speed toward particle LL
10	V_{HL}	Velocity vector of particle toward HL
11	V_{LL}	Velocity vector of particle toward LL
12	min_s, max_s	The minimum and the maximum amount of benchmark function
13	b	The number of aimless particle

$$\text{then } y_i = m_{i1}x_1 + m_{i2}x_2 + \dots + m_{iD}x_D \quad i = 1, 2, \dots, D \tag{30}$$

$(y_i - 500)^2$, that is, z_i is set in portion to the square distance between y_i and the bound.

When one dimension in x vector is changed, all dimensions in vector y will be affected. Hence, the rotated function cannot be solved by just D one-dimensional searches. The orthogonal rotation matrix does not affect the shape of the functions. In this paper, we used Salomon’s method to generate the orthogonal matrix [76].

In rotated Schwefel’s function, in order to keep the global optimum in the search range after rotation, noting that the original global optimum of Schwefel’s function is at $[420.96, 420.96, \dots, 420.96]$, $y' = M * (x - 420.96)$ and $y = y' + 420.96$ are used instead of $y = M * x$. Since Schwefel’s function has better solutions out of the search range $[500, -500]^D$, when $|y_i| > 500$, $z_i = 0.001$

4 Experiments

A number of benchmarks have been used to test the proposed algorithm. Choosing each of these functions for testing had a particular reason, for instance Ackley function is used to *aimless* particles testing. These particles are expected to find one of the minimum areas with their random movements and guide the group toward there. Each of these functions has particular features that generally have been applied in numerous articles to examine the optimization methods. The SSO method with different iterations of particles and different iterations has been

Table 3 Benchmarks that are used to test the ability of the proposed algorithm

#	Function	Equation	Domain	F_{\min}	D
f_1	Sphere	$\sum_{i=1}^D x_i^2$	± 5.12	0	30 Unimodal
f_2	Rosenbrock	$\sum_{i=1}^{D-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	± 50	0	30
f_3	Schwefel's P2.22	$\sum_{i=1}^D x_i + \prod_{i=1}^D x_i $	± 10	0	30
f_4	Quadratic	$\sum_{i=1}^D (\sum_{j=1}^i x_j)^2$	± 100	0	30
f_5	Step	$\sum_{i=1}^D (x_i + 0.5)^2$	± 100	0	30
f_6	Quadratic noise	$\sum_{i=1}^D ix_i^4 + \text{rand}[0, 1)$	± 1.28	0	30
f_7	Ackley	$20 + e - 20e^{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^D x_i^2}} - e^{0.5\cos(2\pi x_i)}$	± 32	0	30 Multimodal
f_8	Griewank	$\sum_{i=1}^D (\frac{x_i^2}{4000}) - \prod_{i=1}^D \cos(\frac{x_i}{\sqrt{i}}) + 1$	± 600	0	30
f_9	Rastrigin	$\sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10)$	± 5.12	0	30
f_{10}	Perm #1 [86]	$\sum_{x=1}^4 [\sum_{i=1}^4 (t^k + \beta)((x_i/t)^k - 1)]^2$	± 4 $\beta = 50$	0	30
f_{11}	Schwefel	$\sum_{i=1}^D -x \sin(\sqrt{ x_i })$	± 500	-12569.5	30
f_{12}	Non-continuous Rastrigin	$\sum_{i=1}^D (y_i^2 - 10 \cos(2\pi x_i) + 10)$ where $y_i = \begin{cases} x_i & x_i < 0.5 \\ \text{round}(\frac{2x_i}{\pi}) & x_i \geq 0.5 \end{cases}$	± 5.12	0	30
f_{13}	Generalized penalized	$\pi/D \left\{ 10 \sin^2(\pi y_i) + \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_D - 1)^2 \right\} + \sum_{i=1}^D u(x_i, 10, 100, 4)$ where $y_i = 1 + \frac{1}{4}(x_i + 1), u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \leq x_i \leq a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	± 50	0	30
f_{14}	Rotated	$418.9829 * D - \sum_{i=1}^D z_i$ $z_i = \begin{cases} y_i \sin(\sqrt{ y_i }) & \text{if } y_i \leq 500 \\ 0 & \text{otherwise} \end{cases}$ $y_i = y_i' + 420.96$ $y' = M * (x - 420.96)$ M is an orthogonal matrix	± 500	0	30 Rotated and shifted
f_{15}	Rotated Rastrigin	$\sum_{i=1}^D (y_i^2 - 10 \cos(2\pi y_i) + 10)$ $y = M * x$	± 5.12	0	30
f_{16}	Rotated Ackley	$-20 \exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^D y_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^D \cos 2\pi y_i\right) + 20 + e$ $y = M * x$	± 32	0	30

Table 3 continued

#	Function	Equation	Domain	F_{\min}	D
f_{17}	Rotated Griewank	$\sum_{i=1}^D \left(\frac{y_i^2}{4,000} \right) - \prod_{i=1}^D \cos \left(\frac{y_i}{\sqrt{i}} \right) + 1$ $y = M * x$	± 600	0	30
f_{18}	Shifted Rosenbrock	$\sum_{i=1}^D \left(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2 \right) + f_bias_6$ $z = x - o + 1, \quad x = [x_1, x_2, \dots, x_D]$ $o = [o_1, o_2, \dots, o_D] : \text{the shifted global optimum}$	± 100	390	30
f_{19}	Shifted Rastrigin	$\sum_{i=1}^D \left(z_i^2 - 10 \cos(2\pi z_i) + 10 \right) + f_bias_9$ $o = [o_1, o_2, \dots, o_D] : \text{the shifted global optimum}$	± 5	-330	30

tested by every function of bench and then has been compared to FSO and PSO.

In SSO algorithm, first amount of parameters are $try_number = 3, S = 3, \delta = 16, visual_{\min} = 0.001, dimension = 30,$ and $step_{\min} = 0.0002$. Proposed idea besides standard FSO and standard PSO simulated in the MATLAB software and then functions of section three have been used to test the accuracy of the method performance. Goal of this simulation is to compare the performance of these three methods in speed of convergence and escaping from minimum local points. First, the Ackley function is applied to test. Table 4 shows the results of performance of two optimization methods (PSO and FSO) and the proposed method (SSO).

Regarding the Table 4, proposed method (SSO) does not have good performance when the number of particles is low, and in comparison with other two methods, it does not achieve good results. But by increasing the number of particles, performance of proposed method begins being refined and shows a better performance than PSO's and FSO's. This is a fact that when there are a few swallows in a colony, there is not a big success. This shows why swallows choose bigger colonies to live in and to migrate in (Fig. 8).

It is seen clearly in Fig. 9 when exploring particles' number is high, the convergence speed of the SSO method heightens, and in iterations that are more than 100, the process of finding the global minimum point is satisfying. FSO has a better convergence speed than PSO, but unfortunately both methods got stuck in a local minimum point in the function Ackley and they have not optimized after 600 repetitions. Another point that is seen in Fig. 9 is that in iterations less than 50, the proposed method (SSO) has a same performance as the FSO.

Function Sphere is a good function to test every optimization method because it is a simple structure one with a global minimum equal to zero. Generally, different optimization methods perform quite well in this function. Table 5 compares the proposed method with PSO and FSO.

Performance of the proposed method stays the same in comparison with FSO when there are 100 particles and 1,000 unchanging iterations. This is correct for 50 particles too. But when the number of particles decreases and comes to 30, the SSO performs a better performance, and if the number of particles keeps on decreasing and comes to (5, 10), the SSO performs a better performance again comparing to other two methods. But a steep decrease in the number of particles (2, 3) changes the result. Then, FSO and PSO make a better performance, respectively (Fig. 10).

Performance of three optimization methods with a low number of particles (2) and 1,000 iterations is shown in Fig. 11. The FSO and PSO had a quite good performance

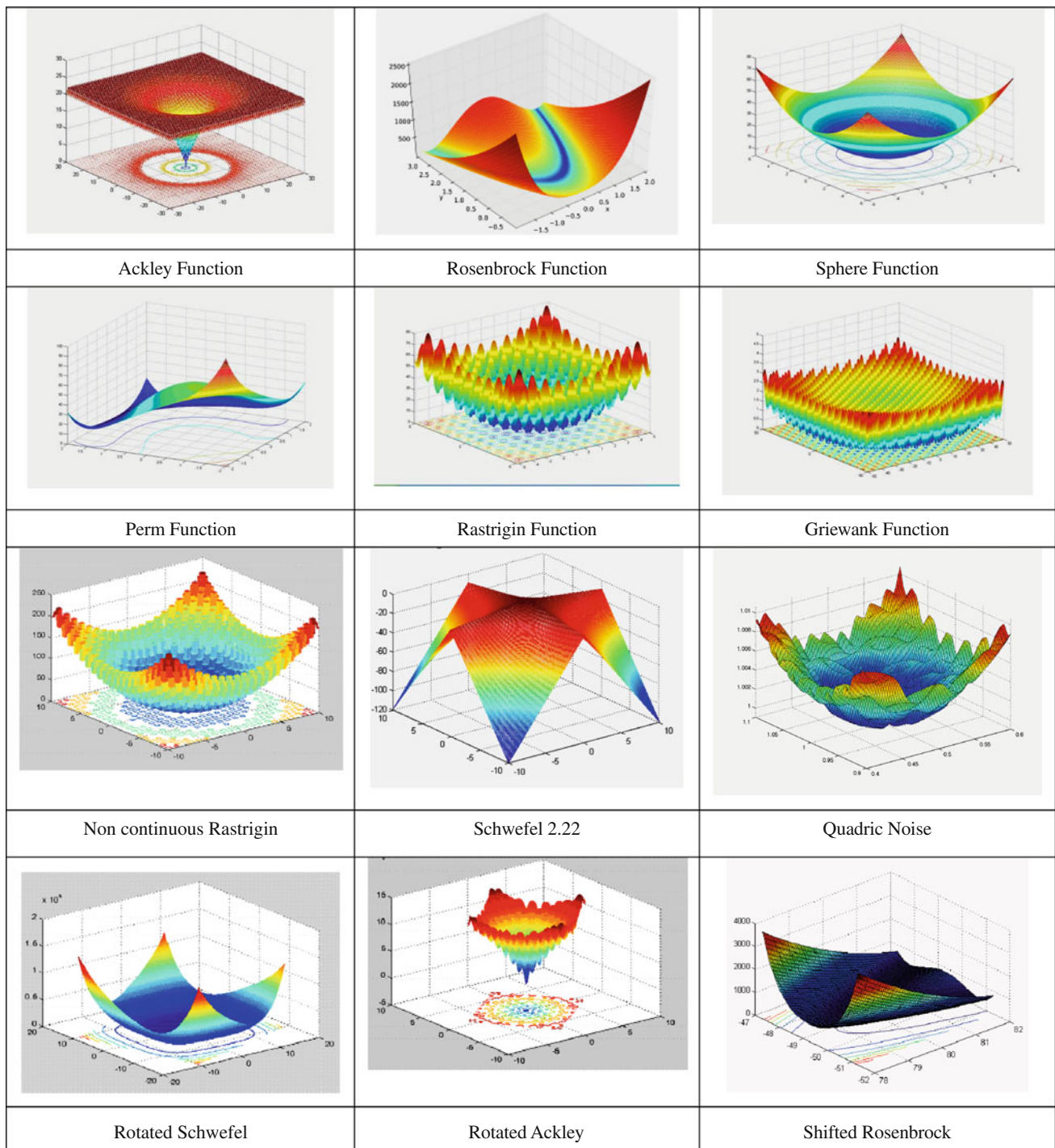


Fig. 7 Benchmark tests and its view in 2-dimensional axes

and could find small minimum amounts. Since the proposed method is heavily relied on swarm intelligence knowledge, when the population is small, it confronts a fundamental deficiency. Actually, this problem is not a weak point of the SSO but it is its particular feature. The number of particles should not be less than a distinct

amount in order to have a good performance. However, in iterations less than 200, this method has a better performance comparing to FSO and PSO, and in iterations 400–600, it is better than PSO again but in iterations more than 600 has not found optimum response and got stuck in local minimum points.

Table 4 Comparing the performance results between two optimization methods (PSO and FSO) and proposed method (SSO) in the function Ackley

#	Particle	Iteration	PSO _{gbest}	FSO _{gbest}	SSO _{gbest}
1	10	100	1.8703e-005	7.1776e-007	3.8612e-009
2	20	100	1.8703e-005	3.5021e-007	4.7025e-012
3	30	100	1.8703e-005	2.8182e-008	1.2436e-014
4	40	100	1.8703e-005	2.5565e-009	3.7155e-015
5	5	100	5.2714e-004	9.4081e-006	2.1381e-006
6	3	100	2.5799	3.6774e-001	5.1442e-001
7	2	100	14.4370	1.0895	7.0286
8	50	1,000	1.003e-015	1.003e-015	5.0842e-22

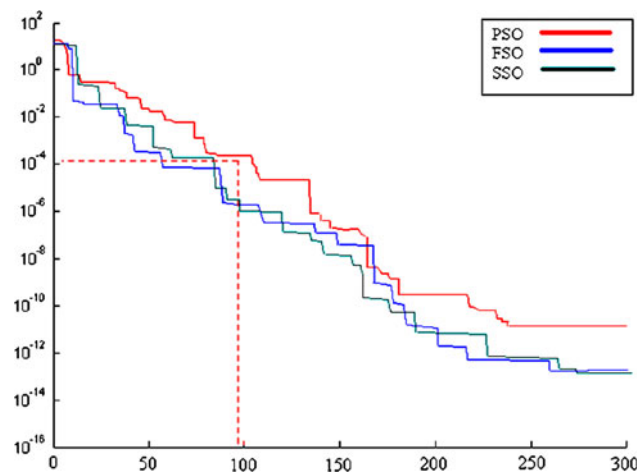


Fig. 8 Comparing the performance of SSO to the FSO and PSO. In the function Ackley with 5 particles and 300 repetition

Function Griewank has a complex structure. It is like a hedgehog full of ups and downs and a collection of local extremum points. It is a hard function for testing the optimization methods. Searching would be stopped, falling down one of the local optimum points. Performance of the SSO in this function is good. Table 6 shows the result of comparing three considered methods with each other.

The SSO has shown a good performance in the function Griewank. Because of the interaction between the particles e_i and aimless movements of particles o_i , the probability of finding the minimum points is well examined. This method comparing to PSO has a better performance with a less number of particles in function Griewank (Fig. 12).

Convergence speed in SSO is more than other two methods. Step-like movements in diagram of SSO performance exhibit how it escapes from local minimum points in order to find a better optimum point (Fig. 13).

SSO with two particles in function Griewank has a good performance. It shows more proper behavior comparing to other two methods in the iterations less than 100, but in

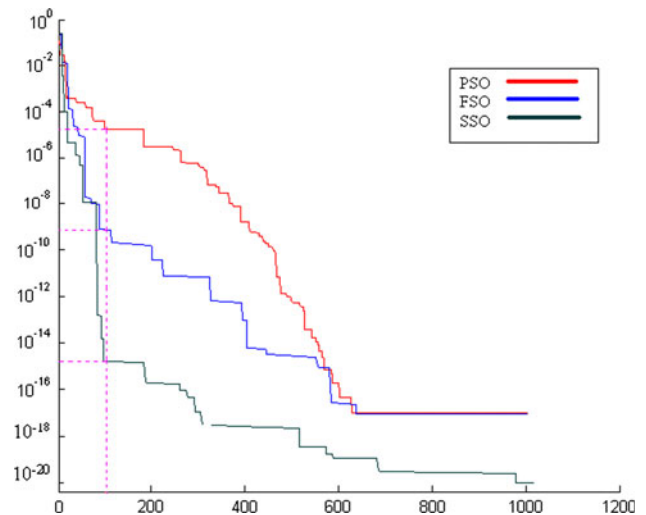


Fig. 9 Comparing the performance of SSO with the FSO and PSO in the function Ackley with 50 particles and 1,000 repetition

Table 5 Comparison between SSO, PSO, and FSO in the function Sphere

Particle	Iteration	PSO _{gbest}	FSO _{gbest}	SSO _{gbest}
100	1,000	1.9240e-082	0	0
50	1,000	7.5243e-078	0	0
30	1,000	3.4892e-064	2.2378e-091	0
10	1,000	4.9240e-041	6.4182e-087	8.0016e-095
5	1,000	2.2926e-038	2.5051e-076	6.0971e-085
3	1,000	6.5324e-034	1.0036e-061	4.0176e-032
2	1,000	4.5519e-028	7.0584e-041	7.4218e-012

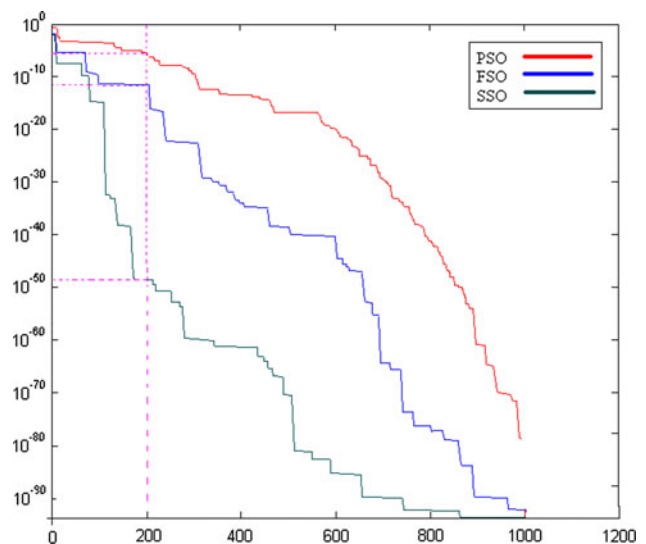


Fig. 10 Comparison between SSO, PSO, and FSO in the function Sphere with 100 particles and 1,000 repetitions

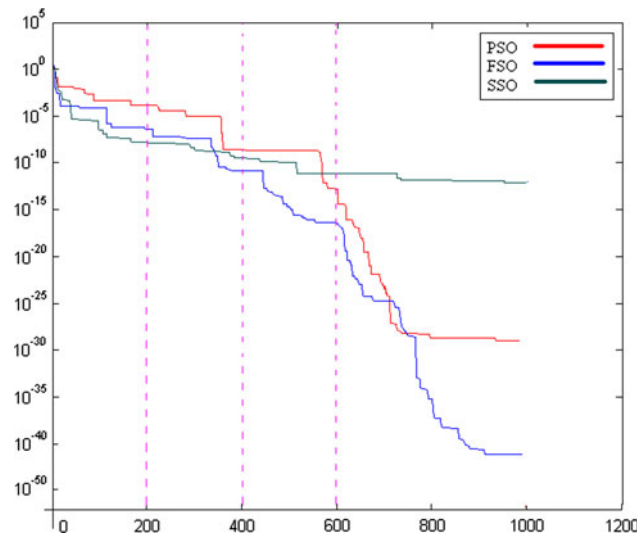


Fig. 11 Comparison between SSO and PSO, FSO in the function Sphere with 2 particles and 1,000 repetitions

Table 6 Comparison between SSO and PSO, FSO in the function Griewank

Particle	Iteration	PSO _{gbest}	FSO _{gbest}	SSO _{gbest}
100	1,000	2.4192e-003	2.0008e-009	7.5301e-012
50	1,000	7.7481e-002	1.7812e-007	1.4704e-011
30	1,000	3.4521e-002	3.2304e-006	3.3501e-009
10	1,000	1.0081e-002	1.4102e-005	4.8516e-008
5	1,000	1.2047e-001	2.5051e-005	4.0072e-003
3	1,000	2.0489e-001	6.1099e-004	1.0051e-002
2	1,000	2.1105e-001	4.0529e-004	1.0063e-002

iterations more than 100, FSO escapes better from local minimum points.

Rastrigin is a hard and difficult function. It, like Griewank, has many local minimum points and, different from Griewank, its apexes (peaks) are too higher and too deeper (Table 7).

Performance of PSO has not been good at all in this function. PSO gets stuck very quickly in one of the local minimum points and cannot find a more proper response. FSO's behavior is better and more intelligently has found more optimum points comparing to PSO. The performance of the SSO is much better than that of other two methods and in all cases with any number of particles has survived from local minimum points (Fig. 14).

Function Rosenbrock has a complex and particular structure. This function has different areas that distract any optimization method. This special function is used in this research to observe the performance of proposed method. Table 8 shows the comparison between proposed method and other two methods.

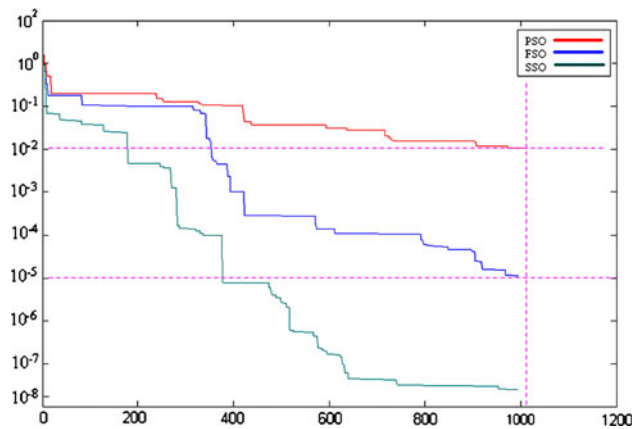


Fig. 12 Comparison between SSO and PSO, FSO in the function Griewank with 10 particles and 1,000 repetitions

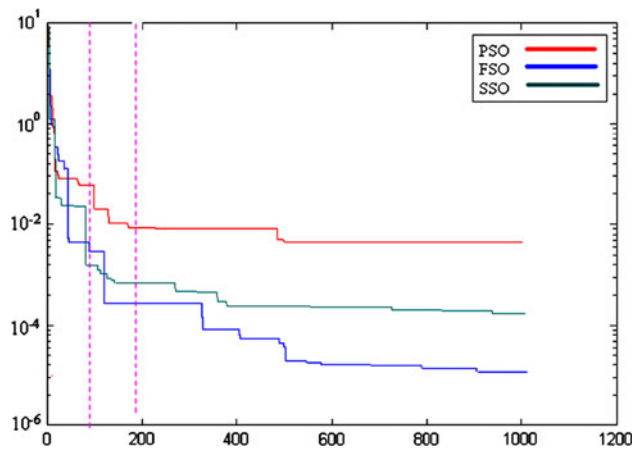


Fig. 13 Comparison between SSO and PSO, FSO in the function Griewank with 2 particles and 1,000 iterations

Table 7 Comparison between SSO and PSO, FSO in the function Rastrigin

Particle	Iteration	PSO _{gbest}	FSO _{gbest}	SSO _{gbest}
100	1,000	1.08254e-004	7.3841e-009	3.3557e-015
50	1,000	2.3961e-002	1.0014e-006	7.2471e-013
30	1,000	8.73	4.5106e-005	8.4519e-012
10	1,000	10.26	9.0045e-005	1.0041e-009
5	1,000	10.57	1.5067e-002	6.7581e-004
3	1,000	14.81	2.15	2.0076e-002
2	1,000	14.73	4.78	2.12

As shown in Table 8, the PSO represents inappropriate responses. But FSO with 100 particles has a quite good response. The SSO has been able to pass different areas of this function and recognizes different minimum points. Appointing some local leaders and using aimless particles

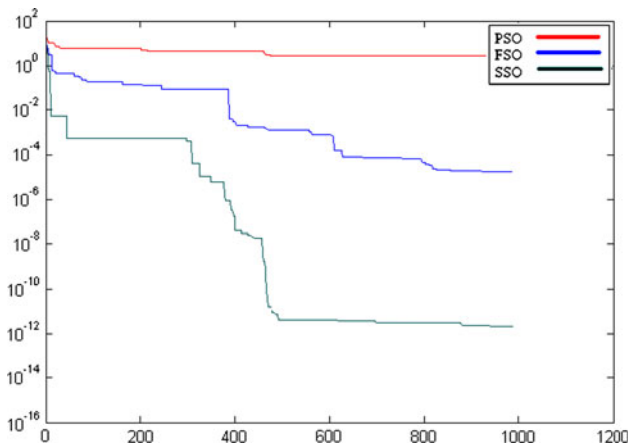


Fig. 14 Comparison between SSO, PSO, and FSO in the function Rastrigin with 30 particles and 1,000 repetitions

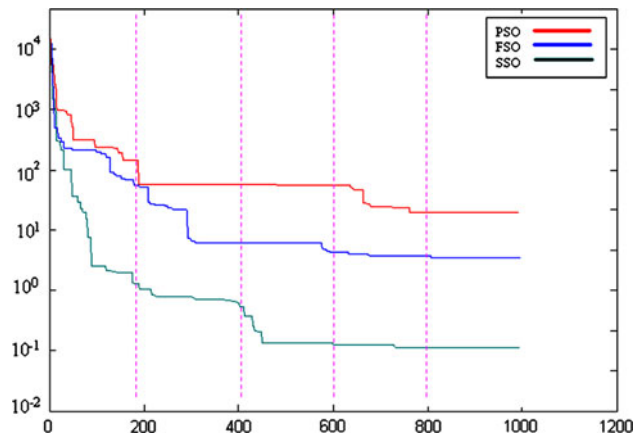


Fig. 15 Comparison between SSO, PSO, and FSO in the function Rosenbrock with 10 particles and 1,000 repetitions

Table 8 Between SSO, PSO, and FSO in the function Rosenbrock

Particle	Iteration	PSO _{gbest}	FSO _{gbest}	SSO _{gbest}
100	1,000	5.0158	8.4573e-001	4.1217e-004
50	1,000	11.2104	1.381	9.2483e-003
30	1,000	18.0025	3.4502	2.1172e-003
10	1,000	26.5891	4.8604	1.0043e-001
5	1,000	29.4508	8.955	1.2541
3	1,000	38.25	13.104	3.402
2	1,000	38.6271	13.8004	4.815

Table 9 Comparison between SSO, PSO, and FSO in the function perm

Particle	Iteration	PSO _{gbest}	FSO _{gbest}	SSO _{gbest}
100	1,000	3.4573e-003	2.0792e-004	1.0102e-004
50	1,000	9.1824e-002	4.0482e-004	2.5724e-004
30	1,000	3.0045e-001	1.6614e-003	7.8254e-004
20	1,000	8.4793e-001	5.3554e-003	4.1047e-003
15	1,000	0.14	7.0008e-002	8.4471e-003
10	1,000	1.004	9.7329e-002	2.5711e-003
5	1,000	3.024	1.3381e-001	6.1049e-002

o_i have been a great help to intensify the feasibility of finding the minimum points.

Performance of the three methods is shown in Fig. 15. All the three methods have had a quite good performance in iterations less than 200 and a descending movement toward minimum points. In iterations between 200 and 400, the PSO has not optimized but other two methods especially PSO have behaved well. In iterations between 400 and 600, the PSO has not optimized again but the FSO has had a small optimization and the SSO has found better minimum points again. In iterations between 600 and 800, the PSO suddenly performs a good movement and jumps over a local minimum point but other two methods have not had any noticeable optimization. At the end (iterations between 800 and 1,000), the three methods have not had any noticeable optimization. Totally, the SSO method has had a very good function and in lower numbers of iterations, has found optimum points with high convergence speed.

Perm is a hard function. Its outline is like a flatfish keeping its wings up and has a curving back. In this function, particle distribution in goal space is so crucial in order to let the particles have the chance of finding the global optimum point. Table 9 shows the results of these three methods' performance.

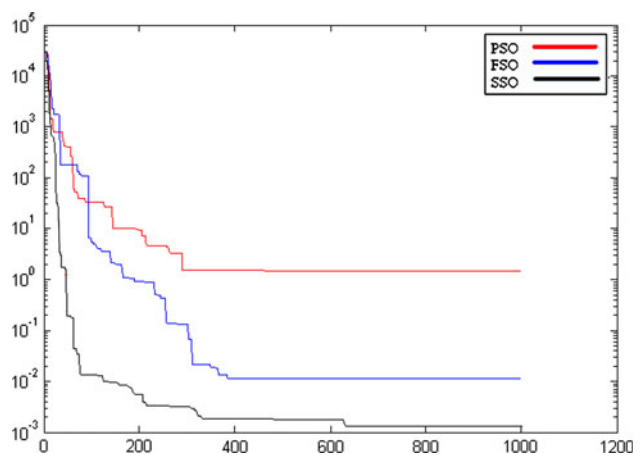


Fig. 16 Comparison between SSO, PSO, and FSO in the function Perm with 10 particles and 1,000 repetitions

The SSO has a quite good performance in this function. Its responses are close to FSO's. But what is really clear is that its results are better than PSO's (Fig. 16).

Convergence speed in the function Perm is very good. In iterations less than 200, it has been able to achieve considerable minimum points. Common aspect between these three

methods is in iterations more than 400. All three methods have got stuck in local minimum points and could not release themselves up to the end. The FSO did not have a good commencement. It means that in iterations less than 100, it had a bad behavior comparing to the PSO. But in iterations more than 100, it has found proper minimum points.

Global best has been used for comparing the proposed method to other optimization methods up to now; however, this topology is bad for multimodal functions [77].

Local best PSO, or lbest PSO, uses a ring social network topology where smaller neighborhoods are defined for each

particle. The social component reflects the information exchanged within the neighborhood of the particle, reflecting local knowledge of the environment. With reference to the velocity equation, the social contribution to particle velocity is proportional to the distance between a particle and the best position found by the neighborhood of particles [78].

In this comparison, the number of particles 20, the number of iterations 1000, the size of neighborhood 2, ring topology, and the parameters $c_1 = c_2 = 1.5$, $\omega = 0.73$ have been used for lbest PSO. $try_number = 3$, $S = 3$, $\delta = 16$, $visual_{min} = 0.001$, $dimension = 30$, and $step\ visual_{min} = 0.002$ have been used for FSO. Comparison of the proposed method to the other two methods PSO and FSO with the size of neighborhood 2 is shown in Table 10.

According to the results of the Table 10, the SSO method, considering the lbest, is better than the other two methods FSO and PSO. The reason why it is better is local searching of function Sphere as well as having a local leader, which causes the particles not to converge prematurely and not to get stuck in local optimum points. The SSO has found more optimum points in multimodal functions and has achieved better results. A considering thing about the SSO is that the results of local best have been improved in comparison with the global best. This is an attribute of the SSO. Besides, the results of the FSO have been noticeable.

Certainly, it is not claimed in this research that the SSO is the best optimization method since this is a new method and should be optimized by other researchers as the PSO has been optimized by other researchers and, now, there are different kinds of this good optimization method. Many methods of PSO algorithm have been innovated. Table 11 shows different methods of PSO.

It is represented that the comparison among the performance of new optimization method SSO and a number of current optimization methods. This is done to assess the significance of the SSO among other current methods.

Table 10 Comparison between SSO and PSO, FSO in the local best

Benchmark		PSO _{Lbest}	FSO _{Lbest}	SSO _{Lbest}
Ackley	Mean	7.547e-002	2.84e-004	4.841e-007
	Min	7.99e-015	3.045e-019	7.172e-024
	Max	1.5017	5.21e-001	1.842
Griewank	Mean	9.399e-002	2.874e-002	1.895e-005
	Min	0	0	0
	Max	5.407e-001	1.472e-001	2.647e-001
Quadric	Mean	1.039e-011	3.542e-014	1.521e-015
	Min	8.2216e-016	5.157e-021	2.708e-023
	Max	1.6906e-010	2.657e-012	1.952e-011
Quadric noise	Mean	1.325e-002	2.238e-004	1.411e-004
	Min	4.8734e-004	2.416-e006	5.274e-007
	Max	2.9155e-002	1.286e-002	1.561e-002
Rastrigin	Mean	54.2849	7.124e-003	1.604e-005
	Min	25.8689	2.206e-005	4.522e-010
	Max	85.5663	2.071e-001	1.055e-002
Rosenbrock	Mean	3.2552	5.602e-004	8.741e-005
	Min	4.128e-005	1.0477e-008	8.525e-010
	Max	19.091	2.255e-002	2.581e-002
Sphere	Mean	5.514e-160	4.117e-171	2.11e-185
	Min	1.3621e-177	2.554e-180	0
	Max	2.757e-158	1.478e-142	3.344e-161

Bold values indicate the best values in the benchmark functions

Table 11 Shows different methods of PSO

Algorithm	Year	Topology	Parameters setting	Reference
GPSO	1998	Global star	$w: 0.9-0.4, c_1, c_2 = 2$	[23]
LPSO	2002	Local ring	$w: 0.9-0.4, c_1, c_2 = 2$	[54]
VPSO	2002	Local von Neumann	$w: 0.9-0.4, c_1, c_2 = 2$	[55]
FIPS	2004	Local URing	$\chi = 0.729, \sum c_i = 4.1$	[58]
HPSO-TVAC	2004	Global star	$w: 0.9-0.4, c_1 = 2.5 - 0.5, c_2 = 0.5 - 2.5$	[37]
DMS-PSO	2005	Dynamic multiswarm	$w: 0.9-0.2, c_1 = c_2 = 2, m = 3, R = 5$	[42]
CLPSO	2006	Comprehensive learning	$w: 0.9-0.2, c = 1.49445, m = 7$	[59]
OPSO	2008	Orthogonal particle swarm	$w: 0.9-0.4, c_1 = c_2 = 0.2, V_{max} = 0.5*rang$	[87]
APSO	2009	Adaptive swarm	Adaptation of the inertia weight	[50]
OLPSO	2010	Orthogonal learning particle swarm	$w: 0.9-0.4, c = 2, G = 5, V_{max} = 0.2*rang$	[53]

Table 12 Comparison between SSO and optimization several methods

Algorithm	Sphere	Rosenbrock	Ackley	Griewank	Rastrigin	Schwefel
GPSO	1.98e-053	28.1	1.15e-014	2.37e-002	8.68	-10,090.16
LPSO	4.77e-029	21.8627	1.85e-014	1.10e-02	7.25	-9,628.35
VPSO	5.11e-038	37.6469	1.40e-014	1.31e-02	8.07	-9,845.27
FIPS	3.21e-030	22.5387	7.69e-015	9.04e-04	10.92	-10,113.8
HPSO-TVAC	3.38e-041	13	2.06e-010	1.07e-02	3.71	-10,868.57
DMS-PSO	3.85e-054	32.3	8.52e-015	1.31e-02	6.42	-9,593.33
CLPSO	1.89e-019	11	2.01e-012	6.45e-13	6.64e-011	-12,557.65
OPSO	6.45e-018	49.61	6.23e-009	2.29e-03	6.97	-8,402.53
APSO	1.45e-150	2.84	1.11e-014	1.67e-02	1.01e-14	-12,569.5
OLPSO-G	4.12e-054	21.52	7.98e-015	4.83e-03	1.07	-9,821.74
OLPSO-L	1.11e-038	1.26	4.14-e015	0	0	-12,150.63
SSO	0	2.4373e-001	4.7025e-012	4.8516e-008	1.8104e-010	-12,569.5
Best method	SSO	SSO	OLPSO-L	OLPSO-L	OLPSO-L	SSO&APSO

Algorithm	Schwefel's P2.22	Quadric	Step	Quadric noise	Perm	N_Rastrigin	Generalized Penalized
GPSO	2.51e-034	6.45e-002	0	7.77e-003	1.02e-001	15.5	1.04e-002
LPSO	2.03e-020	18.6	0	1.49e-002	1.41e-002	30.4	2.18e-030
VPSO	6.29e-027	1.44	0	1.08e-002	12.5	21.33	3.46e-003
FIPS	1.32e-017	0.77	0	2.55e-003	5.68e-001	35.91	1.22e-031
HPSO-TVAC	6.90e-023	2.89e-007	0	5.54e-002	2.02e-002	1.83	7.07e-030
DMS-PSO	2.61e-029	47.5	0	1.10e-002	2.78	32.8	2.05e-032
CLPSO	1.01e-013	395	0	3.92e-003	4.05e-001	1.67e-002	1.59e-021
OPSO	1.26e-010	2.44e-002	0	4.87e-002	2.33e-002	2.49e-006	1.56e-019
APSO	5.15e-084	1.13e-010	0	4.66e-003	2.94e-003	4.14e-016	3.27e-031
OLPSO-G	9.85e-030	5.59e-006	0	6.21e-003	1.28	1.05e-011	1.59e-032
OLPSO-L	7.67e-022	1.56e-001	0	1.32e-002	5.31e-002	6.32e-009	1.57e-032
SSO	1.58e-078	4.16e-015	0	2.86e-003	1.01e-004	6.04e-019	1.84e-031
Best method	APSO	SSO	#	FIPS	SSO	SSO	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	1.50e-02	3.16e-007	1.28e-008	424.83	-245.77
HPSO-TVAC	5.32e-003	52.9	9.29	9.26e-003	494.2	-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	2.45e007	-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-08	415.94	-330
SSO	3.11e-003	41.02	1.08e-14	1.93e-011	403.48	-330
Best method	APSO	SSO	OLPSO-L	SSO	SSO	CLPSO&OLPSO-L&SSO

Bold values indicate the best values in the benchmark functions

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 1,000. Table 12 shows the performance of the methods of Table 11 and the SSO method.

Table 13 The ranking of different methods in every benchmark function

f_1	f_2	f_3	f_4	f_5	f_6
SSO	SSO	OLPSO_L	OLPSO_L	OLPSO_L	SSO(1)
APSO	OLPSO_L	FIPS	CLPSO	APSO	APSO(1)
DMS-PSO	APSO	OLPSO_G	SSO	CLPSO	CLPSO
OLPSO_G	CLPSO	DMS_PSO	FIPS	SSO	OLPSO_L
GPSO	HPSO	APSO	OPSO	OLPSO_G	HPSO
HPSO	OLPSO_G	GPSO	OLPSO_G	HPSO	FIPS
OLPSO_L	LPSO	VPSO	HPSO	DMS-PSO	GPSO
VPSO	FIPS	LPSO	LPSO	OPSO	VPSO
FIPS	GPSO	CLPSO	VPSO(8)	LPSO	OLPSO_G
LPSO	DMS_PSO	SSO	DMS_PSO(8)	VPSO	LPSO
CLPSO	VPSO	HPSO	APSO	GPSO	DMS_PSO
OPSO	OPSO	OPSO	GPSO	FIPS	OPSO
f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
APSO	SSO	All equals	FIPS	SSO	SSO
SSO	APSO	...	SSO	APSO	APSO
GPSO	HPSO	...	CLPSO	LPSO	OLPSO_G
OLPSO_G	OLPSO_G	...	APSO	HPSO	OLPSO_L
DMS_PSO	OPSO	...	OLPSO_G	OPSO	OPSO
VPSO	GPSO	...	GPSO	OLPSO_L	CLPSO
HPSO	OLPSO_L	...	VPSO	GPSO	HPSO
OLPSO_L	FIPS	...	DMS_PSO	CLPSO	GPSO
LPSO	VPSO	...	OPSO	FIPS	VPSO
FIPS	LPSO	...	OLPSO_L	OLPSO_G	LPSO
CLPSO	DMS_PSO	...	LPSO	DMS_PSO	DMS_PSO
OPSO	CLPSO	...	HPSO	VPSO	FIPS
f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}
APSO	SSO	OLPSO_L	SSO	CLPSO	SSO(1)
SSO	DMS_PSO	OLPSO_G	FIPS	SSO	CLPSO(1)
OLPSO_L	OLPSO_G	SSO	APSO	OLPSO_L	OLPSO_L(1)
OLPSO_G	APSO	DMS_PSO	OLPSO_L	OLPSO_G	OLPSO_G
DMS_PSO	HPSO	APSO	CLPSO	FIPS	HPSO
VPSO	OLPSO_L	OPSO	OPSO	GPSO	APSO
CLPSO	LPSO	FIPS	OLPSO_G(7)	APSO	DMS_PSO
FIPS	GPSO	CLPSO	LPSO(7)	LPSO	VPSO
OPSO	OPSO	VPSO	VPSO	HPSO	OPSO
LPSO	VPSO	LPSO	HPSO	VPSO	FIPS
GPSO	CLPSO	GPSO	DMS_PSO	DMS_PSO	LPSO
HPSO	FIPS	HPSO	GPSO	OPSO	GPSO

According to the results of Table 12, in the Sphere function, the proposed method achieved the best result and the total optimum point. The APSO with a proper result is placed second, and the DMS-PSO with a rather good result is placed third. In the Rosenbrock function, the proposed method, the OLPSO-L, and the APSO are placed first to

third, respectively. Table 13 shows the ranking of different methods in every benchmark function.

The best performance in the sphere function was achieved by SSO and after that APSO. GPSO, DMS-PSO, and OLPSO-G have achieved almost the same results, but the speed of convergence of OLPSO-G is more. In the

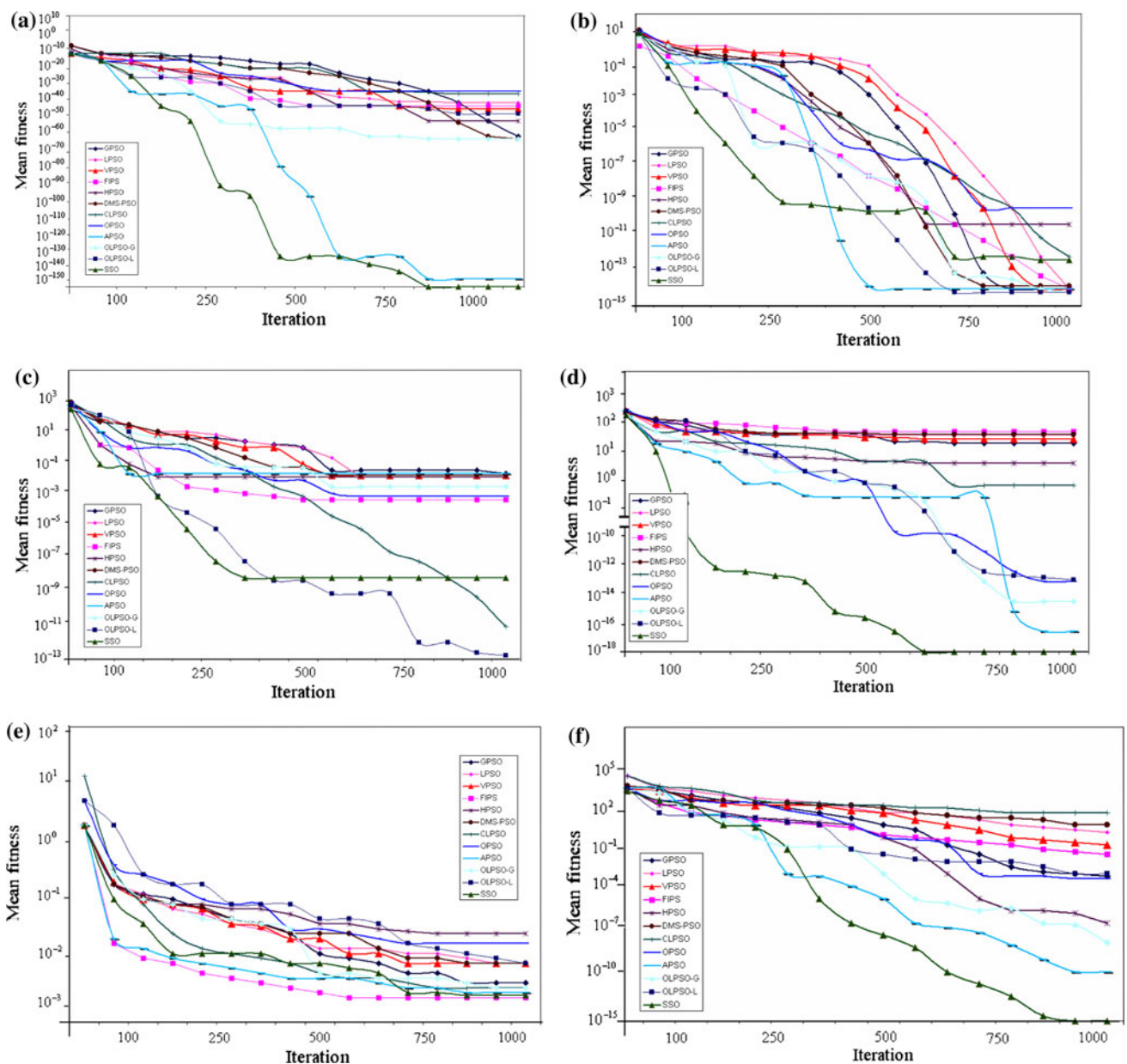


Fig. 17 Convergence performance of the eleven different PSOs and SSO on the six test functions. **a** Sphere, **b** Ackley, **c** Griewank, **d** N_Rastrigin, **e** Quadric noise, **f** Quadric

iterations less than 100, the speed of convergence of APSO is high, but SSO has had a better convergence after the iteration of 200. In all PSO methods except for SSO, getting trapped in local optima points is visible (Fig. 17).

Various behaviors of the eleven PSO methods are seen in the Ackley function, and the best result is achieved by OLPSO-L, and OLPSO-G, FIPS, and DMS-PSO have achieved almost the same results. HPSO and OPSO proved to have gotten trapped in local optima points and to obtain the worst results. In the iterations less than 250, SSO had the fastest convergence among the other methods, but the particles were trapped in local optima and could not escape

from this point until the iteration of 600. In the iteration of 600, the particles escaped from the local optima point and reached a better result, but in the iteration of 700, they got trapped in the other local optima again and failed to provide better results up to the end of iterations. However, some good methods such as APSO have also similar behaviors toward SSO, but they have shown better performance.

The best performance in the Griewank function has been achieved by OLPSO-L, CLPSO and SSO, respectively. The other methods have nearly identical performance, and getting trapped in local optima points is a serious problem

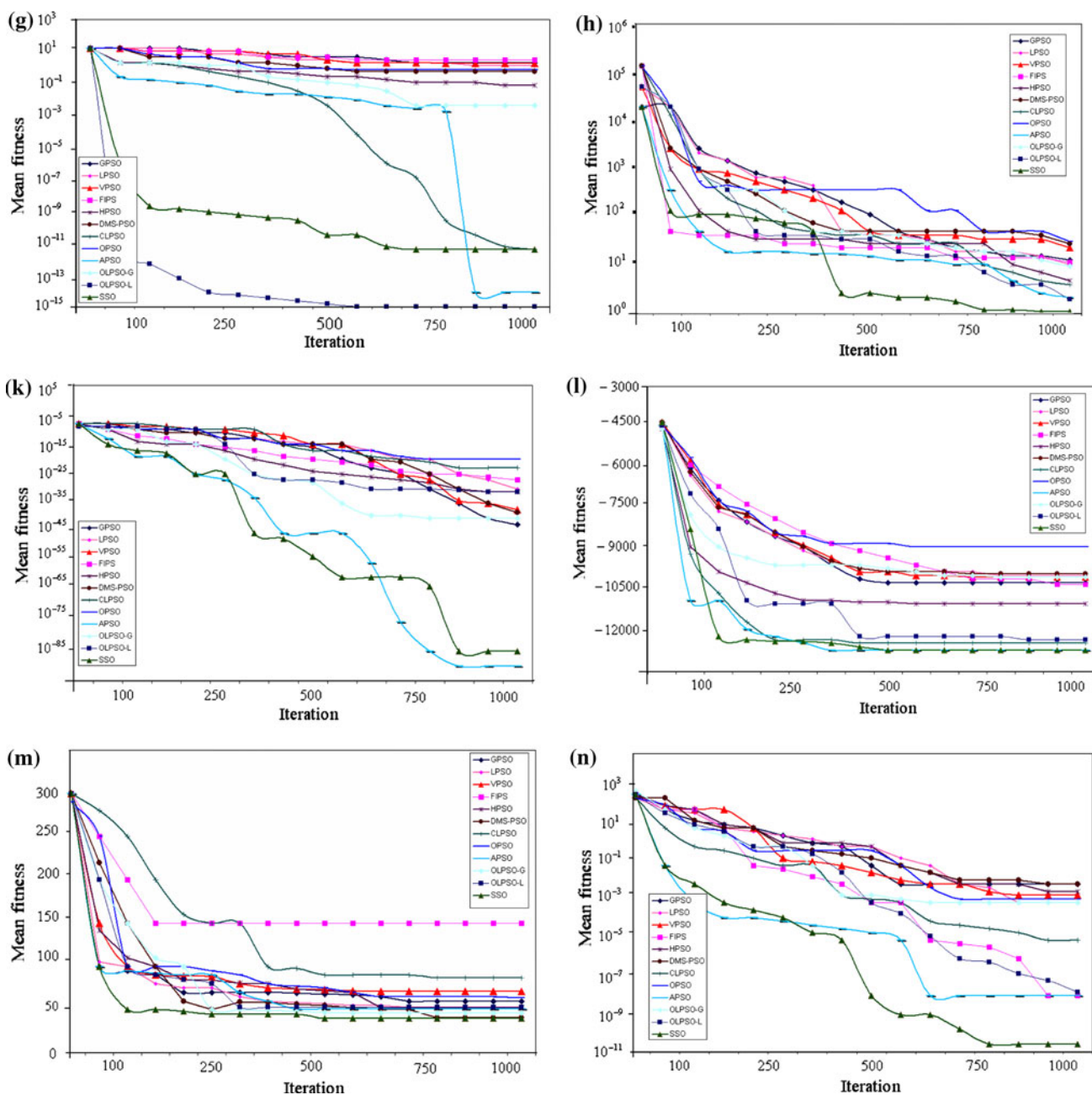


Fig. 18 Convergence performance of the eleven different PSOs and SSO on the six test functions. **g** Rastrigin, **h** Rosenbrock, **k** Schwefel's P2.22, **l** Schwefel, **m** Rotated Rastrigin, **n** Rotated Griewank

in all of them and has been unable to achieve good results. In the iterations less than 300, the speed of convergence of SSO is better than that of the other PSO methods.

The best results in the non-continuous Rastrigin function were obtained by SSO. The speed of convergence and escaping from local optima points in the low iterations can be seen well by the SSO. The function is one of the difficult functions for the various PSO methods. The APSO and OLPSO-G methods could also achieve good results.

The best result in the Quadric function was obtained by SSO, and after that, APSO, OLPSO-G, and HPSO, respectively, achieved relatively good results. SSO has presented a very good behavior in this function, and the role of particles is seen well here. These particles prevent the other particles from getting trapped in the local optima points, and the chance to find some better points will be always investigated in Fig. 18.

The best result in the Rosenbrock function was obtained by SSO; however, in the initial iterations (less than 100),

Table 14 The ranking of different methods in three groups of benchmark functions

	1	2	3	4	5	6	7	8	9	10	11	12
Unimodal	OLPSO_L	SSO	APSO	OLPSO_G	CLPSO	HPSO-TVAC	FIPS	DMS-PSO	GPSO	VPSO	LPSO	OPSO
Multimodal	SSO	APSO	OLPSO_G	OLPSO_L	GPSO	FIPS	OPSO	DMS-PSO	CLPSO	LPSO	VPSO	OPSO
Rotated_shifted	SSO	OLPSO_L	OLPSO_G	APSO	CLPSO	DMS-PSO	FIPS	VPSO	OPSO	LPSO	HPSO_TVAC	GPSO

FIPS has managed to reach a good optima point very quickly.

Between 100 and 400 iterations, the performance of APSO has been better than that of the other methods, but unfortunately it has been trapped in a local optima point and has been able to be optimized. After iteration of 400, SSO has managed to escape from a local optima point and present the best performance.

SSO has had the best performance in the rotated Griewank function. APSO, FIPS, and OLPSO-L methods have also achieved acceptable results. In the iterations less than 250, APSO has converged faster than SSO and has had better performance, but after the iteration of 250, SSO method has managed to become more efficient.

Now, a general comparison between the different methods of PSO and the proposed method is made. Table 14 shows a comparison of the results of different methods in 6 unimodal functions, 7 multimodal functions, and 6 rotated and shifted functions.

According to the results of the Table 14, in unimodal functions, the OLPSO_L, the proposed method, and the OLPSO_L are placed first to third, respectively. In the multimodal functions, the proposed method, the APSO, and the OLPSO_G are placed first to third, respectively. In the rotated and shifted functions, the proposed method, the OLPSO_L, and the OLPSO_G are placed first to third, respectively.

5 Conclusion

In this paper, the method of SSO is a new optimization method that is inspired by the PSO and FSO methods. This method has represented a new optimization method with examining the swallow swarms and their behavioral features. In this method, the particles can have three distinct duties during the searching period: explorer particles, aimless particles, and leader particles. Every particle shows a different behavior regarding its position. They interact with each other continuously. This new method has particular features such as high convergence speed in different functions and not getting stuck in local minimum points. If a particle gets stuck in one of these points, assistance offered by local leader particles and/or aimless particles give hope to it to flee. Different optimization methods such as FSO and kinds of PSO were performed in MATLAB. Different benchmark functions have been used. These methods have been compared to SSO and have been tested. In some functions, the SSO has represented a more optimized response comparing to the other methods, and in some functions and positions, it is rated in second place or third place. It is not claimed in this research that the SSO is the best optimization method but it can be applied in different

engineering problems for optimizing the functions and problems. Regarding the results achieved, it can be claimed that it is one of the best optimization methods in swarm intelligence. This method should be examined and refined by other researchers. Using fuzzy logic for increasing the flexibility of this method is a part of future activities.

References

- Bonabeau E, Dorigo M, Theraulaz G (1999) *Swarm intelligence: from natural to artificial systems*. Oxford University Press, New York
- Dorigo M, Maniezzo V, Colnari A (1996) The ant system: optimization by a colony of cooperating agents. *IEEE Trans Syst Man Cybern Part B* 26(1):29–41
- Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans Evol Comput* 1(1):53–66
- Dorigo M, Stützle T (2004) *Ant colony optimization*. MIT Press, Cambridge
- Kennedy J, Eberhart RC (1995) Particle swarm optimization. *Proceedings of the IEEE international conference on neural networks*. IEEE Press, Piscataway, pp 1942–1948
- Clerc M (2007) Particle swarm optimization. ISTE Ltd., London
- Poli R, Kennedy J, Blackwell T (2007) Particle swarm optimization: an overview. *Swarm Intell* 1(1):33–57
- Li XL (2003) A new intelligent optimization-artificial fish swarm algorithm. PhD thesis, Zhejiang University, China, June, 2003
- Jiang MY, Yuan DF (2006) Artificial fish swarm algorithm and its applications. In: *Proceedings of the international conference on sensing, computing and automation, (ICSCA'2006)*. Chongqing, China, 8–11 May, 2006, pp 1782–1787
- Xiao JM, Zheng XM, Wang XH (2006) 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, pp 3456–3460
- Krishnanand KN, Ghose D (2005) Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. *Proceedings of IEEE swarm intelligence symposium*. IEEE Press, Piscataway, pp 84–91
- Krishnanand KN, Ghose D (2006) Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications. *Multiagent Grid Syst* 2(3):209–222
- Krishnanand KN, Ghose D (2006) Theoretical foundations for multiple rendezvous of glowworm inspired mobile agents with variable local-decision domains. *Proceedings of American control conference*. IEEE Press, Piscataway, pp 3588–3593
- Krishnanand KN, Ghose D (2009) Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. *Swarm Intell* 3:87–124. doi:10.1007/s11721-008-0021-5
- Dorigo M, Trianni V, Sahin E, Gross R, Labella TH, Baldassarre G, Nolfi S, Deneubourg J-L, Mondada F, Floreano D, Gambardella LM (2004) Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots* 17(2–3):223–245
- Fronczek JW, Prasad NR (2005) Bio-inspired sensor swarms to detect leaks in pressurized systems. In: *Proceedings of IEEE international conference on systems, man and cybernetics*. IEEE Press, Piscataway, pp 1967–1972
- Zarzhitsky D, Spears DF, Spears WM (2005) Swarms for chemical plume tracing. *Proceedings of IEEE Swarm intelligence symposium*. IEEE Press, Piscataway, pp 249–256
- Zadeh LA (1965) Fuzzy sets. *Inf Control* 8:338–353
- Heppner H, Grenander U (1990) A stochastic non-linear model for coordinated bird flocks. In: Krasner S (ed) *The ubiquity of chaos*. AAAS, Washington, pp 233–238
- Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: *Proceedings of the sixth international symposium on micro machine and human science*. IEEE, Nagoya, Japan, Piscataway, pp 39–43
- Eberhart RC, Simpson PK, Dobbins RW (1996) *Computational intelligence PC tools*. Academic Press, Boston
- Poli R, Kennedy J, Blackwell T (2007) Particle swarm optimization an overview. *Swarm Intell* 1:33–57. doi:10.1007/s11721-007-0002-0
- Shi Y, Eberhart RC (1998) A modified particle swarm optimizer. In: *Proceedings of IEEE world congress on computational intelligence*, pp 69–73
- Clerc M, Kennedy J (2002) The particle swarm-explosion, stability and convergence in a multidimensional complex space. *IEEE Trans Evol Comput* 6(1):58–73
- Trelea IC (2003) The particle swarm optimization algorithm: convergence analysis and parameter selection. *Inf Process Lett* 85(6):317–325
- Yasuda K, Ide A, Iwasaki N (2003) Stability analysis of particle swarm optimization. In: *Proceedings of the 5th metaheuristics international conference*, pp 341–346
- Kadiramanathan V, Selvarajah K, Fleming PJ (2006) Stability analysis of the particle dynamics in particle swarm optimizer. *IEEE Trans Evol Comput* 10(3):245–255
- van den Bergh F, Engelbrecht AP (2006) A study of particle optimization particle trajectories. *Inf Sci* 176(8):937–971
- Shi Y, Eberhart RC (1999) Empirical study of particle swarm optimization. In: *Proceedings of IEEE congress on evolution and computation*, pp 1945–1950
- Shi Y, Eberhart RC (2001) Fuzzy adaptive particle swarm optimization. *IEEE Congr Evol Comput* 1:101–106
- Eberhart RC, Shi Y (2001) Tracking and optimizing dynamic systems with particle swarms. In: *Proceedings of IEEE congress on evolution and computation*, Seoul, Korea, pp 94–97
- Clerc M (1999) The swarm and the queen: toward a deterministic and adaptive particle swarm optimization. In: *Proceedings of IEEE Congress on Evolution and Computation*, pp 1951–1957
- Clerc M, Kennedy J (2002) The particle swarm-explosion, stability and convergence in a multidimensional complex space. *IEEE Trans Evol Comput* 6(1):58–73
- Eberhart RC, Shi Y (2000) Comparing inertia weights and constriction factors in particle swarm optimization. In: *Proceeding of IEEE Congress on Evolution and Computation*, pp 84–88
- Kennedy J (1997) The particle swarm social adaptation of knowledge. In: *Proceedings of IEEE international conference on Evolution and computation*. Indianapolis, IN, pp 303–308
- Suganthan PN (1999) Particle swarm optimizer with neighborhood operator. In: *Proceedings of IEEE congress on evolution and computation*. Washington DC, pp 1958–1962
- Ratnaweera A, Halgamuge S, Watson H (2004) Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Trans Evol Comput* 8(3):240–255
- Angeline PJ (1998) Using selection to improve particle swarm optimization. In: *Proceedings of IEEE congress on evolution and computation*. Anchorage, AK, pp 84–89
- Juang CF (2004) A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Trans Syst Man Cybern B Cybern* 34(2):997–1006
- Chen YP, Peng WC, Jian MC (2007) Particle swarm optimization with recombination and dynamic linkage discovery. *IEEE Trans Syst Man Cybern B Cybern* 37(6):1460–1470
- Andrews PS (2006) An investigation into mutation operators for particle swarm optimization. In: *Proceedings of IEEE congress*

- on evolution and computation. Vancouver, BC, Canada, pp 1044–1051
42. Liang JJ, Suganthan PN (2005) Dynamic multi-swarm particle swarm optimizer with local search. In: Proceedings of IEEE congress on evolution and computation, pp 522–528
 43. Zhang WJ, Xie XF (2003) DEPSO: hybrid particle swarm with differential evolution operator. In: Proceedings of IEEE conference on systems, man, cybernetics, pp 3816–3821
 44. van den Bergh F, Engelbrecht AP (2004) A cooperative approach to particle swarm optimization. *IEEE Trans Evol Comput* 8(3):225–239
 45. Ratnaweera A, Halgamuge S, Watson H (2004) Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Trans Evol Comput* 8(3):240–255
 46. Parsopoulos KE, Vrahatis MN (2004) On the computation of all global minimizers through particle swarm optimization. *IEEE Trans Evol Comput* 8(3):211–224
 47. Brits R, Engelbrecht AP, van den Bergh F (2002) A niching particle swarm optimizer. In: Proceedings of 4th Asia-Pacific conference on simulation and evolution and learning, pp. 692–696
 48. Brits R, Engelbrecht AP, van den Bergh F (2007) Locating multiple optima using particle swarm optimization. *Appl Math Comput* 189(2):1859–1883
 49. Parrott D, Li XD (2006) Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Trans Evol Comput* 10(4):440–458
 50. Zhan Z, Zhang J, Li Y, Shu-Hung Chung H (2009) Adaptive particle swarm optimization. *IEEE Trans Syst Man Cybern B Cybern* 39(6):1362–1381
 51. Liu J-L, Chang C-C (2008) Novel orthogonal momentum-type particle swarm optimization applied to solve large parameter optimization problems. *J Artif Evol Appl* 1:1–9
 52. Sivanandam SN, Visalakshi P (2009) Dynamic task scheduling with load balancing using parallel orthogonal particle swarm optimization. *Int J Bio Inspired Comput* 1(4):276–286
 53. Zhan Z-H, Zhang J, Li Y, Shi Y-H (2011) Orthogonal learning particle swarm optimization. *IEEE Trans Evol Comput* 15(6):832–847
 54. Kennedy J, Mendes R (2002) Population structure and particle swarm performance. In: Proceedings of IEEE congress on evolution and computation. Honolulu, HI, pp 1671–1676
 55. Kennedy J, Mendes R (2006) Neighborhood topologies in fully informed and best-of-neighborhood particle swarms. *IEEE Trans Syst Man Cyber Part C Appl Rev* 36(4):515–519
 56. Hu X, Eberhart RC (2002) Multiobjective optimization using dynamic neighborhood particle swarm optimization. In: Proceedings of IEEE congress on evolution and computation. Honolulu, HI, pp 1677–1681
 57. Liang JJ, Suganthan PN (2005) Dynamic multi-swarm particle swarm optimizer. In: Proceedings of swarm intelligence symposium, pp 124–129
 58. Mendes R, Kennedy J, Neves J (2004) The fully informed particle swarm: Simpler, maybe better. *IEEE Trans Evol Comput* 8(3):204–210
 59. Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans Evol Comput* 10(3):281–295
 60. Li LX, Shao ZJ, Qian JX (2002) An Optimizing method based on autonomous animals: fish-swarm algorithm. *Syst Eng Theory Pract* 22(11):32–38
 61. Zhang M, Shao C, Li F, Gan Y, Sun J (2006) Evolving neural network classifiers and feature subset using artificial fish swarm. In: Proceedings of the 2006 IEEE international conference on mechatronics and automation, June 25–28. Luoyang, China
 62. Jiang M, Wang Y, Rubio F, Yuan D (2007) Spread spectrum code estimation by artificial fish swarm algorithm. In: IEEE international symposium on intelligent signal processing (WISP)
 63. Jiang MY, Yuan DF (2005) Wavelet threshold optimization with artificial fish swarm algorithm. In: Proceedings of the IEEE international conference on neural networks and brain, (ICNN&B'2005), Beijing, China, 13–15, pp 569–572
 64. Paul Gorenzel W, Salmon TP (1994) Swallows, prevention and control of wildlife damage
 65. Lazareck LJ, Moussavi Z Adaptive swallowing sound segmentation by variance dimension
 66. Angela T, Chris R (1989) Swallows and martins: an identification guide and handbook. Houghton-Mifflin. ISBN 0-395-51174-7
 67. Bijlsma RG, van den Brink B (2005) A Barn Swallow *Hirundo rustica* roost under attack: timing and risks in the presence of African Hobbies *Falco cuvieri*. *Ardea* 93(1):37–48
 68. Saino N, Galeotti P, Sacchi R, Møller A (1997) Song and immunological condition in male barn swallows (*Hirundo rustica*). *Behav Ecol* 8(94):364–371. doi:10.1093/beheco/8.4.364 (<http://dx.doi.org/10.1093%2Fbeheco%2F8.4.364>)
 69. Brown CR (1986) Cliff swallow colonies as information centers. *Science* 234:83–85
 70. Brown CR, Brown M, Shaffer ML (1991) food sharing signals among socially foraging cliff swallows. *Anim Behav* 42:551–564
 71. Safran R (2010) Barn swallows: sexual and social behavior. *Encycl Animal Behav* 1:139–144 (Elsevier)
 72. Snapp BD (1976) Colonial breeding in the barn swallow (*hirundo rustica*) and its adaptive significance. *Condor* 78:3471480
 73. Smith LC, Raouf SA, Brown MB, Wingfield JC, Brown CR (2005) Testosterone and group size in cliff swallows: testing the “challenge hypothesis” in a colonial bird. *Horm Behav* 47:76–82
 74. Mccarty JP, Winkler DW (1999) Foraging ecology and diet tree swallows feeding selectivity of nestlings. *The Condor* 101:246–254. The cooper ornithological society
 75. Whitley D, Rana D, Dzuberka J, Mathias E (1996) Evaluating evolutionary algorithms. *Artif Intell* 85(1–2):245–276
 76. Salomon R (1996) Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions. *BioSystems* 39:263–278
 77. Esquivel SC, Coello CAC (2003) On the use of particle swarm optimization with multimodal functions. *IEEE Congr Evol Comput* 2:1130–1136
 78. Engelbrecht AP (2005) Fundamentals of computational swarm intelligence. Wiley, New York
 79. Esmín AAA, Lambert-Torres G, Alvarenga GB (2006) UFLA, Brazil, hybrid evolutionary algorithm based on PSO and GA mutation, sixth international conference on hybrid intelligent systems. HIS '06
 80. Settles M, Soule T (2005) Breeding swarms: A GA/PSO Hybrid. In: GECCO '05: proceedings of the 2005 conference on genetic and evolutionary computation, pp 161–168
 81. Meng Y, Kazeem O (2007) A hybrid ACO/PSO control algorithm for distributed swarm robots. In: Proceedings of the 2007 IEEE swarm intelligence symposium (SIS 2007)
 82. Gomez-Cabrero D, Ranasinghe DN (2005) Fine-tuning the ant colony system algorithm through particle swarm optimization, technical report TR07-2005. Departamento de Estadística e Investigació Operativa, Universitat de Valencia, Burjassot, Spain
 83. Chen H, Wang S, Li J, Li Y (2007) A hybrid of artificial fish swarm algorithm and particle swarm optimization for feed forward neural network training, 2007 international conference on intelligent systems and knowledge engineering (ISKE 2007)
 84. Shi H, Bei Z (2008) Application of improved ant colony algorithm. In: 4th International conference on natural computation. ICNC '08

85. Shi H, Bei Z (2009) A mixed ant colony algorithm for function optimization. In: Proceedings of the 21st annual international conference on Chinese control and decision IEEE Press Piscataway, NJ, USA, pp 3919–3923
86. Mishra SK (2006) Performance of differential evolution and particle swarm methods on some relatively harder multi-modal benchmark functions. Available at SSRN: <http://ssrn.com/abstract=937147>
87. Ho S-Y, Lin H-S, Liauh W-H, Ho S-J (2008) OPSO: Orthogonal particle swarm optimization and its application to task assignment problems. *IEEE Trans Syst Man Cybern Part A* 38(2): 288–298
88. Berliner S (2004) The Birders Report. <http://home.earthlink.net/~s.berliner/>