International Journal of Computational Intelligence and Applications
Vol. 15, No. 2 (2016) 1650011 (23 pages)
© World Scientific Publishing Europe Ltd.
DOI: 10.1142/S1469026816500115

# A New Stochastic Optimization Approach: Dolphin Swarm Optimization Algorithm

Wang Yong

*College of Information Science and Engineering*
*Guangxi University for Nationalities*
*Nanning 530006, P. R. China*

*Key Laboratory of Guangxi High Schools Complex System and Computational Intelligence*
*Nanning 530006, P. R. China*
*wangygxnn@sina.com*

Wang Tao, Zhang Cheng-Zhi* and Huang Hua-Juan

*College of Information Science and Engineering*
*Guangxi University for Nationalities*
*Nanning 530006, P. R. China*
**zhchengzhi2007@163.com*

Received 2 June 2015
Published 6 June 2016

A novel nature-inspired swarm intelligence (SI) optimization is proposed called dolphin swarm optimization algorithm (DSOA), which is based on mimicking the mechanism of dolphins in detecting, chasing after, and preying on swarms of sardines to perform optimization. In order to test the performance, the DSOA is evaluated against the corresponding results of three existing well-known SI optimization algorithms, namely, particle swarm optimization (PSO), bat algorithm (BA), and artificial bee colony (ABC), in the terms of the ability to find the global optimum of a range of the popular benchmark functions. The experimental results show that the proposed optimization seems superior to the other three algorithms, and the proposed algorithm has the performance of fast convergence rate, and high local optimal avoidance.

*Keywords*: Intelligent computation; nature-inspired optimization; dolphin swarm optimization algorithm (DSOA).

## 1. Introduction

It is a great fillip in the nature-inspired optimization algorithm research that Holland[1] invented the genetic algorithm (GA) in 1975. GA mimics a simple picture of Darwinian natural selection in order to find a good algorithm and involves the operations of "mutation", "selection", and "evaluation of fitness" repeatedly. Since then, GA has been playing a dominant role in the domain of optimization algorithm research, and many researchers have succeeded in proposing various new nature-inspired algorithms based on mimicking the biological mechanisms or natural phenomena. Among various nature-inspired algorithms, swarm-based algorithm is one of the most representative paradigms. Over the last two decades, swarm-based algorithms have attracted more and more attention of researchers, and numerous new swarm-based algorithms have been developed (see Refs. 1–25, 27–35), such as particle swarm optimization (PSO),[2] ant colony optimization (ACO),[3] artificial bee colony (ABC),[4] artificial fish swarm algorithm (AFSA),[5] cuckoo search (CS),[11] bat algorithm (BA),[13] wolf search algorithm (WSA),[14] swallow swarm optimization algorithm (SSO),[15] kinetic energy of gas molecules (KGMO),[16] animal migration optimization (AMO),[17] magnetic optimization algorithm (MOA),[18] ions motion algorithm (IMO),[19] grey wolf optimizer (GWO),[20] ant lion optimizer (ALO),[21] firefly algorithm (FA),[22] krill herd (KH),[23] chicken swarm optimization (CSO),[25] ray optimization (RO),[32] black hole (BH),[34] etc. These algorithms mostly mimic the social and individual behavior of swarm, herds, schools, or groups of creatures in nature, or simulate the natural or physical phenomena in the universe.

Stochastic optimization algorithms have an edge over their traditional mathematical optimization techniques because they can incrementally induce a globally optimum solution by using heuristics to efficiently search a large space. The swarm-based algorithms belong to the stochastic optimization algorithms. The optimization process starts from selecting a set of random solutions, and then, these solutions are merged to create a new population. In general, in order to ensure the improvement of the whole population, the best solutions are saved for improving the quality of the solutions among poor quality. It is obvious that the swarm-based algorithms have the strong point of high local optimum avoidance. But the computational complexity of the swarm-based algorithms is usually higher than that of their traditional mathematical optimization techniques. The swarm-based algorithm is one of the most well-known paradigms in nature-inspired algorithms. The swarm-based algorithm is also called swarm intelligence (SI) method.[26] Swarm-based algorithms, such as PSO, ACO, ABC, AFSA, BA, WSA, etc., have become powerful approaches in solving global optimization problems.

Recently, Mirjalili *et al.*[20] divided the nature-inspired algorithms into three classes: swarm-inspired, evolution-inspired, and physics-inspired algorithms.[20] One of the most representative swarm-inspired algorithms is PSO. The PSO mimics the social behavior of bird flocking when searching for food. ACO is another one of the most representative swarm-inspired algorithms. The ACO algorithm was inspired from the fact that every ant can mark its own paths toward to food source and is well capable of keeping the past paths in mind by pheromone.

GA[1] is the most classical and representative evolution-inspired algorithm. The GA mimics a simple picture of Darwinian natural selection. The operators of the GA involve "mutation", "selection", and "evaluation of fitness". The GA starts with a population of genomes randomly, and then applies crossover and mutation to the individuals in the population to generate new individuals. It uses various selection criteria so that it picks up the best individuals for mating and subsequent crossover. Recently, different kinds of the evolution-inspired algorithms have been presented, such as evolution programming (EP),[27] genetic programming (GP),[28] differential evolution (DE),[29] evolution strategy (ES),[30] and biogeography-based optimizer (BBO),[18] and so forth.

The physics-inspired algorithms are mainly inspired by the natural or physical phenomena in the universe. Recently, different kinds of the physics-inspired algorithms have been proposed, such as KGMO,[16] MOA,[18] IMO,[19] gravitational search algorithm (GSA),[31] RO,[32] states of matter search (SMS),[33] BH,[34] artificial chemical reaction optimization (ACROA),[35] etc. Similar to the previous two classes, optimization is performed by a cluster of search agents (called candidate solutions). The difference is that the search agents are moved/combined based on physics-inspired concepts.[19] For example, MOA[18] mimics the electromagnetic forces between electromagnetic particles to impel the search agents to spread all over the search space. For the electromagnetic force is proportional to the fitness of particles, search agents tend to be attracted toward the fittest particles. So the search agents of the MOA are improved by moving toward the best solutions.[19] Similarly, IMO[19] imitates the attraction and repulsion of anions and cations to perform optimization. The candidate solutions for a given optimization problem in IMO are divided into two groups: anions (negative ions) and cations (positive ions). The ions represent candidate solutions for a particular problem and attraction/repulsion forces impel the ions to spread all over the search space. Anions move toward the best cation, whereas cations move toward the best anion. Their amount of movement depends on the attraction/repulsion forces between them. The ion is evaluated based on its fitness, and the fitness of ion is proportional to the value of the objective function.

Generally, different optimization algorithms usually adopt different optimization strategies, and ordinarily show different performances in application to solve the same optimization problem. On the one hand, according to no-free-lunch theorem,[36] none of the optimization algorithms can solve all optimization problems in the world, including the optimization algorithms mentioned in Refs. 1–35. Nevertheless, these optimizations mentioned above may have some shortcomings compared to each other. For example, not all of these optimization algorithms mentioned above show the same performance when they are applied to solve the same optimization problem. These optimization algorithms still have some shortcomings such as easily being trapped into local minima. Therefore, it is worthwhile to introduce a new swarm optimization algorithm if it is suitable for solving a specific optimization problem.

In this paper, getting inspiration from the behavior of dolphins' search and attacking on swarms of sardines in the sea, we propose a new nature-inspired swarm optimization algorithm, called the dolphin swarm optimization algorithm (DSOA). The remainder of this paper is organized as follows: Sec. 2 introduces the background, principle and method of the DSOA. Section 3 reports on the experiment and discuss the results. Section 4 summarizes the results and concludes the paper.

## 2. Dolphin Swarm Optimization Algorithm

### 2.1. *The major natural features of dolphins*

There are almost 40 different species of dolphins in the world. Dolphin is a kind of very clever sea mammals. Dolphin is often regarded as one of the most intelligent animals on the earth.[37] Dolphin has a streamlined body that makes it very suitable for fast swimming in the water. Hence most of dolphins have superb swimming skills and unusual diving ability. They can dynamically change their swimming modes and swimming directions, and use various swimming styles such as flip-swimming stroke and rotational-swimming stroke when they are attacking or preying on a colony of sardines or attacking sharks, avoid the attack of their natural enemies and various obstacles, and avoid being immersed into blind alley.

Dolphins are acoustic creatures, and ultrasonic is their common language. Each dolphin has a special "melon-structure" on its head.[37] The special "melon-structure" can both emit ultrasonic and receive echo. The dolphin communicates with its companions relying on the "melon-structure". Relying on its special "melon-structure", dolphin can quickly and accurately detect the distance, direction, position and shape of object. That is why the experts who engage in the research of sonar usually call the "dolphin's melon-structure" as a high sensitive sonar. The range of frequency of the ultrasonic emitted by dolphin's sonar is about in the region between 15 kHz and 350 kHz.[37] In the following, for simplicity, we call the "dolphin's melon-structure" as a sonar.

Dolphins usually have a social life and like to live in groups. Dolphins usually live in a colony of up to dozens of individuals, especially when they are chasing after or preying on a great swarm of sardines, or attacking sharks, etc. Dolphins are very successful in search of food, and mostly they eat fish and the small marine animals such as sardine, squid, shrimp and so forth, and it is a more typical example that dolphins like to attack or prey on swarms of sardines such as seen in Figs. 1(a)

(a) (b)

Fig. 1. A colony of dolphins is preying on a swarm of sardines.

and 1(b): a colony of dolphins is preying on or attacking a great swarm of sardines using the flip-swimming mode and the rotational-swimming mode.

### 2.2. *The main idea of the DSOA*

Many behavior characteristics of dolphins remain unknown, and it is rather difficult for us to figure out all of the features that dolphins have. But from the previous description, we have known some of the characteristics of dolphins and got some useful knowledge as follows: The dolphin is one of the most intelligent animals on the earth. Dolphins are acoustic creatures. Each dolphin has a "sonar" on its head. The dolphin emits ultrasonic, receives echo, detects the position of its prey, and communicates with its companions relying on its sonar. Dolphins like to prey on or attack swarms of sardines in the sea, and usually use both flip-swimming mode and rotational-swimming mode when they are preying on or attacking a swarm of sardines. Dolphins like to have a social life, like to live in a colony, etc.

Here, in order to produce a new nature-inspired optimization algorithm, we use some natural behavior characteristics of dolphins and make the following assumptions:

(1) Dolphins only search, attack or prey on swarms of sardines in their search space.
(2) Dolphins only use their flip-swimming mode or rotational-swimming mode when they are attacking or preying on swarms of sardines in the sea.
(3) Each dolphin only uses its sonar to emit ultrasonic, receive echo, detect the position of object, and communicate with its companions.
(4) The whole search process can be divided into three scenarios: detecting the locations of targets (swarms of sardines); chasing after and approach the targets; preying on or attack the targets finally.
(5) The dolphin can dynamically change its swimming mode and can use different search strategies in accordance with its need.

Inspiration from the behavior characteristics of dolphins' chasing after, attacking or preying on swarms of sardines in the sea, we propose a new swarm optimization algorithm that we call the DSOA in this paper. In our simulations, we use virtual dolphins naturally.

### 2.3. *Basic rules of the DSOA*

In this paper, we use the following basic mathematics knowledge: for a closed interval $[a, b]$, we can use a linear transformation $\varphi(x) = x - (b + a)/2$ to map it into the interval $[-\frac{b-a}{2}, \frac{b-a}{2}]$. So in the following, without loss of generality, we suppose that the search space of dolphins is $S = [-a_1, a_1] \times [-a_2, a_2] \times \cdots \times [-a_n, a_n] \subset \mathbf{R}^n$, where $a_k > 0$, $k = 1, \ldots, n$.

### 2.3.1. *The initial positioning method*

According to the assumption that the dolphin uses its sonar to detect the positions of targets (swarms of sardines), we design the location approach in accordance with the basic principle of sonar as follows.

For each interval $[-a_k, a_k]$ in $S$, we first construct a square domain in the complex plane as follows: $F_k = \{h_R + ih_I | h_R, h_I \in [-a_k, a_k]\}$, where $i$ is the imaginary unit. For each point $h_R + ih_I \in F_k$, it can be expressed as $h_R = \rho_k \cos\theta_k$, $h_I = \rho_k \sin\theta_k$, and $\rho_k = \sqrt{h_R^2 + h_I^2}$, where $0 \le \rho_k \le a_k$, $\theta_k \in [0, 2\pi]$.

We denote the initial velocity of the $j$th dolphin to be $V_R^j(1) + iV_I^j(1)$, where $V_R^j(1) = (v_{R,1}^j(1), \ldots, v_{R,n}^j(1))$, $V_I^j(1) = (v_{I,1}^j(1), \ldots, v_{I,n}^j(1))$. Suppose, in the beginning, that the $j$th dolphin detects that there is a swarm of sardines in the location $\rho_k^j(\cos\theta_k^j + i\sin\theta_k^j)$. Making use of the point $\rho_k^j(\cos\theta_k^j + i\sin\theta_k^j)$, the $j$th dolphin gets the data of coordinate as follows:

$$x_{R,k}^j(1) = \rho_k^j \cos\theta_k^j, \quad x_{I,k}^j(1) = \rho_k^j \sin\theta_k^j, \quad \text{and} \quad X_R^j(1) + iX_I^j(1), \tag{1}$$

where $\rho_k^j$ is a random number in the range $[0, a_k]$, $\theta_k^j$ is a random angle in the interval $[0, 2\pi]$, $i$ is the imaginary unit, $X_R^j(1) = (x_{R,1}^j(1), \ldots, x_{R,n}^j(1))$, $X_I^j(1) = (x_{I,1}^j(1), \ldots, x_{I,n}^j(1))$, $k = 1, \ldots, n$, $j = 1, \ldots, M$, $M$ is the population of dolphins.

### 2.3.2. *The way of chasing after the prey*

According to the assumption mentioned above, dolphins only search, attack or prey on swarms of sardines in their search space. Therefore, if dolphins have found targets (swarms of sardines) in their search space, they will approach or chase after the targets from different directions, and will attack or prey on the targets finally. Then we design the moving rule as follows.

Suppose at the time step $t$, the position and the moving velocity of the $j$th dolphin are represented as $X_R^j(t) + iX_I^j(t)$ and $V_R^j(t) + iV_I^j(t)$ respectively, the best target (a swarm of sardines) found by the $j$th dolphin is in the position $P_R^j(t) + iP_I^j(t)$, and the best target (a swarm of sardines) detected by all dolphins is represented as $G_R(t) + iG_I(t)$ (Explanation: $G_R(t) + iG_I(t)$ is the barycenter of the best target (a swarm of sardines), and the same meaning in the following), where $i$ is the imaginary unit. Then the new position $X_R^j(t+1) + iX_I^j(t+1)$ and the velocity $V_R^j$ $(t+1) + iV_I^j(t+1)$ at the time step $t+1$ are given by the following equations:

$$\begin{aligned} V_R^j(t+1) = {} & w_1 \cdot V_R^j(t) + c_{R1}^j(t) \cdot (P_R^j(t) - X_R^j(t)) + c_{R2}^j(t) \cdot (G_R(t) - X_R^j(t)) \\ & + c_{R3}^j(t) \cdot (G_I(t) - G_R(t)), \end{aligned} \tag{2}$$

$$\begin{aligned} V_I^j(t+1) = {} & w_2 \cdot V_I^j(t) + c_{I1}^j(t) \cdot (P_I^j(t) - X_I^j(t)) + c_{I2}^j(t) \cdot (G_I(t) - X_I^j(t)) \\ & + c_{I3}^j(t) \cdot (G_R(t) - G_I(t)), \end{aligned} \tag{3}$$

$$X_R^j(t+1) = X_R^j(t) + V_R^j(t+1), \tag{4}$$

$$X_I^j(t+1) = X_I^j(t) + V_I^j(t+1), \tag{5}$$

where $X_R^j(t), X_I^j(t)$, $G_R(t), G_I(t) \in \mathbf{R}^n$, $\beta_q$ is an $n$-dimension random vector drawn from the range $[0,1]$, both $w_1$ and $w_2$ are inertia weights, $c_{R1}^j(t)$, $c_{R2}^j(t)$, $c_{R3}^j(t)$, $c_{I1}^j(t)$, $c_{I2}^j(t)$ and $c_{I3}^j(t)$ are acceleration controlling functions. In this paper, both $c_{R3}^j(t)(G_I(t) - G_R(t))$ and $c_{I3}^j(t)(G_R(t) - G_I(t))$ are called the error correction terms.

In our simulations, we have used $c_{R1}^j(t) = c_{I1}^j(t) = 2, w_1 = w_2 = 0.6, \beta_1 = \beta_2 = 2$, $c_{R2}^j(t) = c_{R3}^j(t) = \beta_1 \cdot (\text{Gen} - t)/\text{Gen}$, $c_{I2}^j(t) = c_{I3}^j(t) = \beta_2 \cdot (\text{Gen} - t)/\text{Gen}$, where Gen is the maximum number of iterations, and $t$ is the time step.

### 2.3.3. *The method of attacking or preying on the prey*

On the one hand, the aim that dolphins chase after or approach the swarm of sardines is to attack or prey on the swarm finally. Hence if these dolphins continuously chase after the swarm of sardines for sometime, they will catch up with the swarm or arrive at the advantageous positions where they can launch an attack or prey on the swarm of sardines. Nevertheless, according to our assumptions, dolphins only use their flip-swimming mode or rotational-swimming mode to move when they are attacking or preying on the swarm of sardines. In accordance with the scenarios of dolphins' attacking or preying on swarms of sardines, we design the position updating formulas as follows.

Suppose the group of dolphins continuously chase after a swarm of sardines for $\omega$ time steps ($\omega <$ Gen), some of these dolphins have arrived at the advantageous positions where they can attack or prey on the swarm of sardines. According to this case, we design the position updating formulas (6) and (7) to depict the scene such as seen in Fig. 1, that there are some of these dolphins attacking or preying on a swarm of sardines, as follows:

$$x_{R,k}^j(t+1) = r_{R,l}, \tag{6}$$

$$x_{I,k}^j(t+1) = r_{I,l}. \tag{7}$$

where $r_{R,l}$ is a uniformly distributed random number in the interval $[g_{R,l}(t) - \varepsilon(t), g_{R,l}(t) + \varepsilon(t)]$, $r_{I,l}$ is a uniformly distributed random number in the interval $[g_{I,l}(t) - \varepsilon(t), g_{I,l}(t) + \varepsilon(t)]$, $l$ is a random number belonging to the set $\{1, \cdots, n\}$, $\varepsilon(t)$ is a decreasing positive function and satisfied $|\varepsilon(t)| \ll 1$ (that is to say, the absolute value of the $\varepsilon(t)$ is much less than 1), $t$ is the time step, $\omega$ is a positive integer, and $t \geq \omega$, $\omega <$ Gen, Gen is the maximum number of iterations, $k = 1, \ldots, n$, $X_l^j(t) = (x_{l,1}^j(t), \ldots, x_{l,n}^j(t))$, $G_l(t) = (g_{l,1}(t), \ldots, g_{l,n}(t))$ $(l = I, R)$.

In our simulations, we have used $\omega = 20$ and $\varepsilon(t) = 1/1000t^{100}$.

**Explanation:** Equations (6) and (7) describe the scenarios that some of these dolphins are moving to the vicinity of the position $G_R(t) + iG_I(t)$ to attack or prey on the swarm of sardines ($G_R(t) + iG_I(t)$ is the barycenter of the swarm of sardines actually) from different directions, and few of them arrive at the barycenter to catch these sardines.

### 2.4. *The switch of swimming modes*

Since most of dolphins can dynamically change their swimming modes and can use different swimming modes according to their need. So in this paper, we design the position updating rules as follows.

Let $M$ be the population size of dolphins, $f(X)$ be the optimization problem we discuss (we only discuss the minimum problem in this paper), $X = (x_1, \ldots, x_n) \in S$, and $X_R^j(t) + iX_I^j(t)$ be the position of the $j$th dolphin at the time step $t$. We mark

$$Y^j = \frac{1}{2}(f(X_R^j(t)) + f(X_I^j(t))), \quad j = 1, \ldots, M. \tag{8}$$

We first sort all these $Y^j (j = 1, \ldots, M)$ according to the ascending sort. Suppose $Y^j$ is ranked at $O^j(t)$ among all these $Y^l (l = 1, \ldots, M)$ according to the ascending sort.

```
Begin
   Giving the objective function $f(X)$
 Step 1: Initialization Initialize the population of dolphins (the population size is $M$)
         randomly; Set the maximum generation Gen, dimension $n$, parameter:
         $w_1, w_2, c_{R1}, c_{I1}, \beta_1, \beta_2, \omega$ ; Set the generation counter $t = 1$.
 Step 2 : Fitness evaluation. Evaluate each dolphin according to its position.
 Step 3 : $t = t+1$.
 Step 4 : While the best solution is not found or t<Gen do
    If $t \le \omega$
       Each dolphin uses Eqs. (2)–(5) to update its position;
       Evaluation the population according to the newly update positions;
    Else
       For $j$ = 1 to $M$ (for all dolphins) do
             The $j$th dolphin uses Eq. (8) to calculate $Y^j$.
       End for $j$
       Sort all the dolphin individuals according to these $Y^j$.
       For $j$ = 1 to M (for all dolphins) do
          Each dolphin uses Eq. (9) to calculate its switching factor $\mu^j$.
          Select a random number rand $\in [0,1]$.
          If $\mu^j \le$ rand
             The $j$th dolphin uses Eqs. (6) and (7) to update its position
          Else
             The $j$th dolphin uses Eqs. (2)–(5) to update its position
          End if
       End for $j$
    End if
       Evaluation the population according to the newly update positions;
    $t = t+1$.
 Step 5: End while
 Step 6: Output the best solution.
End
```

Fig. 2. Pseudo-codes of the DSOA algorithm.

We denote

$$\mu^j(t) = (O^j(t) - 1)/M. \tag{9}$$

Here, we call $\mu^j$ as a switching factor.

**Explanation:** It is obvious that $0 \le \mu^j(t) < 1$. If $Y^j = \min\{Y^l | l = 1, \ldots, M\}$, then $O^j(t) = 1$. If $Y^j = \max\{Y^l | l = 1, \ldots, M\}$, then $O^l(t) \le O^j(t) = M' \le M$, $l = 1, \ldots, M$.

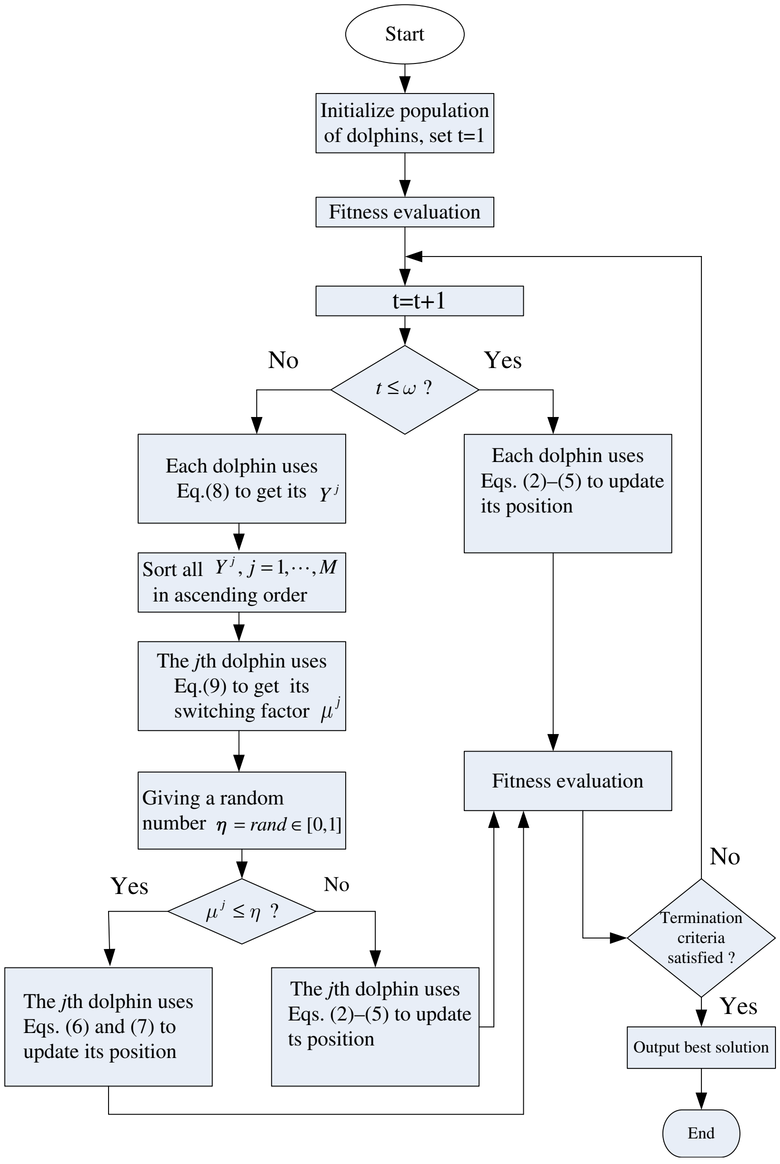

Start
Initialize population of dolphins, set t=1
Fitness evaluation
t=t+1
No
Yes
$t \le \omega$ ?
Each dolphin uses Eq.(8) to get its $Y^j$
Each dolphin uses Eqs. (2)–(5) to update its position
Sort all $Y^j, j = 1, \cdots, M$ in ascending order
The *j*th dolphin uses Eq.(9) to get its switching factor $\mu^j$
Fitness evaluation
Giving a random number $\eta = rand \in [0,1]$
No
Yes
No
$\mu^j \le \eta$ ?
Termination criteria satisfied ?
Yes
The *j*th dolphin uses Eqs. (6) and (7) to update its position
The *j*th dolphin uses Eqs. (2)–(5) to update ts position
Output best solution
End

Fig. 3. Flowchart of the DSOA algorithm.

In our paper, we design the switching rules of dolphins' swimming modes in accordance with the following Rule 1 and Rule 2:

Rule 1: If $t \leq \omega$ or $\eta < \mu^j$, then the $j$th dolphin will use the formulas (2)–(5) to update its position and velocity.

Rule 2: If $t > \omega$ and $\mu^j \leq \eta$, then the $j$th dolphin will use the formulas (6) and (7) to attack or prey on the swarm of sardines.

Here, $\eta$ is a random number in the domain $[0, 1]$.

According to the description above, the pseudo-codes of the DSOA algorithm are provided in Fig. 2.

Correspondingly, a brief presentation of the DSOA algorithm is shown in Fig. 3.

## 3. Experimental Results and Discussions

### 3.1. *Benchmark functions*

To examine the performance of optimization algorithms, benchmark functions are usually employed. In this paper, in order to show the performance of the proposed new algorithm (DSOA), we implement the DSOA in MATLAB and run with diverse famous benchmark functions so as to compare its performance with other well-established nature-inspired optimization algorithms. Though there are many test functions, there is no standard set or official number of test functions that one has to follow. In our simulations, we only choose some of these test functions to examine the efficiency and performance of the proposed new algorithm (DSOA). Each of these functions tests the DSOA in diverse or special conditions. So the drawbacks of the DSOA will usually be revealed in the numerical experiments, and the experimental results can mainly reflect the performance of the DSOA. The properties and the formulas of these functions are presented below.

(a) Schwefel's function:

$$f_1(X) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|, \quad -10 \leq x_i \leq 10, n = 30.$$

This function is a unimodal function, and has a global minimum $f_{\min} = 0$ at $(0, \ldots, 0)$. 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.

(b) Rastrigin's function:

$$f_2(X) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)], \quad -5.12 \leq x_i \leq 5.12, n = 30.$$

Rastrigin's function is a complex multimodal problem with a large number of local optima, and the number of its local minima shows an exponential increase with the problem dimension and its peak shape is up and down in jumping. When attempting to solve this function, algorithms may easily fall into a local optimum. Therefore, an algorithm capable of maintaining a larger diversity is likely to yield better results, so this function is viewed as a typical function used for testing global search performance of algorithm. It has a global minimum $f_{\min} = 0$ at $(0, \ldots, 0)$.

(c) Step function:

$$f_3(X) = \sum_{i=1}^{n} (\lfloor 0.5 + x_i \rfloor)^2, \quad -100 \le x_i \le 100, n = 30.$$

Step function is a discontinuous function and has a global minimum $f_{\min} = 0$ at $(-0.5, \ldots, -0.5)$.

(d) Ackley's function:

$$f_4(X) = -20\exp\left[-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right] - \exp\left[\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right] + 20 + e,$$
$$-30 \le x_i \le 30, n = 30.$$

Ackley's function is a multimodal problem and has one narrow global optimum basin and many minor local optima. The function has a global optimum $f_{\min} = 0$ at $(0, \ldots, 0)$.

(e) Sphere function:

$$f_5(X) = \sum_{i=1}^{n} x_i^2, \quad -5.12 \le x_i \le 5.12, \;\; n = 30.$$

This function is a unimodal function, and its global minimum is $f_{\min} = 0$ at $(0, \ldots, 0)$.

(f) Griewank's function:

$$f_6(X) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \quad -600 \le x_i \le 600, \;\; n = 30.$$

The function is a multimodal problem. In the test function, the search space is relatively large, and there are a lot of local optima. On the other hand, the test function has a $\prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}})$ component causing linkages among variables, thereby making it difficult to reach the global optimum. Therefore, it is generally regarded as a complex multimodal problem that is hard to optimize. The function has a global minimum $f_{\min} = 0$ at $(0, \ldots, 0)$.

(g) Rotated hyper-ellipsoid function:

$$f_7(X) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2, \quad -100 \le x_i \le 100, \;\; n = 30.$$

This function is a unimodal function, and the global minimum is $f_{\min} = 0$ at $(0, \ldots, 0)$.

(h) Schwefel's function:

$$f_8(X) = -\sum_{i=1}^{n}(x_i \sin(\sqrt{|x_i|})), \quad -500 \leq x_i \leq 500, \ \ n = 30.$$

The complexity of Schwefel's function is due to its deep local optima 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. Its global minimum is $f_{\min} \approx -418.9829n$ and the optimum point is at the point $(420.9687, \ldots, 420.9687)$ in the $n$-dimensional search space.

(i) Zakharov's function:

$$f_9(X) = \sum_{i=1}^{n} x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{n} ix_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{n} ix_i\right)^4, \quad -5 \leq x_i \leq 10, \ \ n = 30.$$

This function is a unimodal function, and the global minimum is $f_{\min} = 0$ at $(0, \ldots, 0)$.

(j) Rosenbrock's function:

$$f_{10}(X) = \sum_{i=1}^{n-1}[(x_i - 1)^2 + 100(x_{i+1} - x_i^2)^2], \quad -5 \leq x_i \leq 10, \ \ n = 30.$$

Rosenbrock's function is a multimodal problem. The function has several local minima, and its global extreme point is located in a narrow and flat parabolic shape valley. So it is very difficult for the algorithm to search the global extreme point. Therefore, the function is often used to evaluate the optimal performance of algorithm. The function has a global minimum $f_{\min} = 0$ at $(1, \ldots, 1)$.

(k) Quartic's function:

$$f_{11}(x) = \sum_{i=1}^{n} ix_i^4 + \text{random}[0, 1), \quad x_i \in [-1.28, 1.28], \ \ n = 30.$$

The global minimum of this function is $f_{\min} = 0$ at $(0, \ldots, 0)$.

(l) Generalized Penalized 1 functions:

$$f_{12}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4),$$

where

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a < x_i < a, \\ k(-x_i - a)^m, & x_i < -a, \end{cases} \quad -50 \leq x_i \leq 50, \ \ n = 30.$$

This function is a multi-peak function, and its global minimum is $f_{\min} = 0$ at $(1, 1, \ldots, 1)$.

(m) Schwefel's Problem 2.21:

$$f_{13}(x) = \max_i\{|x_i|, 1 \le i \le n\}, \quad -100 \le x_i \le 100, \quad n = 30.$$

This function is a unimodal function, and its global minimum is $f_{\min} = 0$ at $(0, \dots, 0)$.

(n) Generalized Penalized two functions:

$$\begin{aligned} f_{14}(x) = {} & \frac{\pi}{n}\left\{10\sin(\pi y_1) + (y_n - 1)^2 + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})]\right\} \\ & + \sum_{i=1}^{n} u(x_i, 10, 100, 4), \end{aligned}$$

where

$$y_i = 1 + \frac{x_i + 1}{4}, \quad u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a < x_i < a,, \\ k(-x_i - a)^m, & x_i < -a, \end{cases}$$
$$-50 \le x_i \le 50, n = 30.$$

This function is a multimodal problem, and its global minimum is $f_{\min} = 0$ at $(1, 1, \dots, 1)$.

(o) Shekel's Foxholes function:

$$\begin{aligned} f_{15}(x) &= \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}, \quad -65.536 \le x_i \le 65.536, \\ a_{ij} &= (-32, -16, 0, 16, 32, -32, L, 0, 16, 32; -32, -32, -32, -32, -32, -16, \\ & \quad L, 32, 32, 32). \end{aligned}$$

This function is a two-fixed-dimensional multimodal function, and its global minimum is $\min f((X^*)) = f(-32, -32) = 0.998$.

(p) Kowalik's function:

$$\begin{aligned} f_{16}(x) &= \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 x_4}\right]^2, \quad -5 \le x_i \le 5, \\ a &= [0.1957, 0.1947, 0.1735, 0.16, 0.0844, 0.0627, 0.0456, 0.0342, 0.0323, \\ & \quad 0.0235, 0.0246], \\ (1/b_i) &= [0.25, 0.5, 1, 2, 4, 6, 8, 10, 12, 14, 16]. \end{aligned}$$

This function is a four-fixed-dimensional multi-peak function, and its global minimum is $\min(f(X^*)) \approx f(0.1928, 0.1928, 0.1231, 0.1358) \approx 0.00030754$ in the range $[-5, 5]$.

(q) Shekel's family function ($m = 5$):

$$f_{17}(x) = -\sum_{i=1}^{5} [(X-a_i)(X-a_i)^T + c_i]^{-1}, \quad 0 \le x_i \le 10,$$
$$a = [4,4,4,4;1,1,1,1;8,8,8,8;6,6,6,6;3,7,3,7;2,9,2,9;5,5,3,3;8,1,8,1;6,2,6,2;7,3.6,7,3.6],$$
$$c = [0.1\ 0.2\ 0.2\ 0.4\ 0.4\ 0.6\ 0.3\ 0.7\ 0.5\ 0.5].$$

This function is a four-fixed-dimensional multi-peak function, and its global minimum is $f_{\min} \approx 10.1532$ in the range $[0, 10]$.

(r) Shekel's family function ($m = 7$):

$$f_{18}(x) = -\sum_{i=1}^{7} [(X-a_i)(X-a_i)^T + c_i]^{-1}, \quad 0 \le x_i \le 10,$$
$$a = [4,4,4,4;1,1,1,1;8,8,8,8;6,6,6,6;3,7,3,7;2,9,2,9;5,5,3,3;8,1,8,1;6,2,6,2;7,3.6,7,3.6],$$
$$c = [0.1\ 0.2\ 0.2\ 0.4\ 0.4\ 0.6\ 0.3\ 0.7\ 0.5\ 0.5].$$

This function is a four-fixed-dimensional multi-peak function, and its global minimum is $f_{\min} \approx 10.403$ in the range $[0, 10]$.

(s) Shekel's family function ($m = 10$):

$$f_{19}(x) = -\sum_{i=1}^{10} [(X-a_i)(X-a_i)^T + c_i]^{-1}, \quad 0 \le x_i \le 10,$$
$$a = [4,4,4,4;1,1,1,1;8,8,8,8;6,6,6,6;3,7,3,7;2,9,2,9;5,5,3,3;8,1,8,1;6,2,6,2;7,3.6,7,3.6],$$
$$c = [0.1\ 0.2\ 0.2\ 0.4\ 0.4\ 0.6\ 0.3\ 0.7\ 0.5\ 0.5].$$

This function is a four-fixed-dimensional multi-peak function, and its global minimum is $f_{\min} \approx 10.536$ in the range $[0, 10]$.

(t) Hartman's function:

$$f_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right), \quad 0 \le x_i \le 1,$$
$$c = [1, 1.2, 3, 3.2],$$
$$a = [10,3,17,3.5,1.7,8;.05,10,17,1,8,14;3,3.5,1.7,10,17,8;17,8,0.5,10,1,14],$$
$$p = [1312,1696,5569,0124,8283,5886;2329,4135,8307,3736,1004,9991;2348,1415,3522,2883,3047,6650;4047,8828,8732,5743,1091,0381].$$

This function is a six-fixed-dimensional multi-peak function, and its global minimum is $\min f(X^*) = f(0.201, 0.15, 0.477, 0.275, 0.311, 0.657) \approx -3.3194$ in the range $[0, 1]$.

## 3.2. *Comparison of experimental results and discussions*

In order to test the performance of the proposed new algorithm (DSOA), we have compared it with other three nature-inspired optimization algorithms, including PSO, ABC, and BA. There are many means to carry out the comparison of algorithm performance. We can use such ways: to compare the number of function evaluations for a given accuracy, or to compare their accuracies for a fixed number of function evaluations, and so forth. Here, we use the second approach.

In order to ensure the comparability of the test results, these methods should be tested in the same software and hardware, and the parameters settings are set as consistent as possible for the DSOA, PSO, ABC, and BA. In this test, the population size is set at 50 and the number of iterations is set at 2000, applied software is MATLAB version 7.0.4 service pack2, current hardware is Intel® Core™ i5-2430M CPU @ 2.40 GHz, and windows 7 operating system.

In our simulation experiments, we have performed 30 independent runs for each algorithm, so that we can do reasonable statistical analysis. We choose the best results (**Best**), the worst results (**Worst**), the mean results (**Mean**), the standard deviation (**Std.**) of the results, and the success rate (**SR**) of algorithm in finding the global optima as the evaluation indicators of optimization ability for the four algorithms mentioned above. These evaluation indicators usually reflect the searching capability and the robustness of algorithm. The experimental results of the best fitness value, the worst fitness value, the mean value, the standard deviation of the results, and the success rate of algorithm in finding the global optima are presented in Table 1. The **Best**, **Worst**, **Mean**, **Std.**, and **SR** respectively represent the best fitness value, the worst fitness value, the mean value, the standard deviation of the results, and the success rate of algorithm in finding the global optima. The experimental results and the comparison of the experimental results are recorded in Table 1, and the comparisons of the convergence speed of the four algorithms are shown in Fig. 4.

The best results among the four algorithms are shown in boldface in Table 1, and the convergence characteristics of the four algorithms when solving these test functions are illustrated in Fig. 4.

As may be seen in Table 1 and Fig. 4, the DSOA is compared to the normal PSO, normal ABC, and normal BA. For the best solution (**Best)** and the success rate (**SR**) in Table 1, we see that the proposed new algorithm (DSOA) has successfully found the optimal solution, and the success rate (**SR**) is 100% for the functions $f_2$, $f_3$, $f_6$, $f_{10}$ and $f_{15}$. Meanwhile, the other three algorithms (PSO, ABC, and BA) do not find the optimal solution in the 30 trials for the five functions. We also note that $f_2$, $f_6$, $f_{10}$ and $f_{15}$ are multimodal problem and $f_3$ is a discontinuous function among of these five test functions. The optimal solution respectively in the functions $f_1$, $f_4$, $f_5$, $f_7$, $f_9$,$f_{16}$, $f_{17}$, $f_{18}$, $f_{19}$ and $f_{20}$ has been found by the proposed new algorithm (DSOA) in the 30 trials, and the success rate (**SR**) is respectively 70%, 3.3%, 86%, 20%, 20%, 3.3%, 3.3%, 3.3%, 3.3% and 3.3%. Meanwhile, all of the other three algorithms

Table 1. The comparison of experimental results.

| Function | Algorithm | **Best** | **Worst** | **Mean** | **Std.** | **SR** (%) |
|---|---|---|---|---|---|---|
| $f_1$ | **DSOA** | **0** | **9.8154E−77** | **3.2718E−78** | **1.7920E−77** | **70** |
| | PSO | 8.2415E−14 | 10.0000 | 0.5262 | 2.2329 | 0 |
| | BA | 106.6090 | 55833831.83 | 5271401.67 | 13462425.94 | 0 |
| | ABC | 0.8303 | 1.6981 | 1.3327 | 0.2518 | 0 |
| $f_2$ | **DSOA** | **0** | **0** | **0** | **0** | **100** |
| | PSO | 51.7378 | 1.1640E+02 | 76.6755 | 19.6463 | 0 |
| | BA | 125.5769 | 253.8909 | 188.4617 | 37.2492 | 0 |
| | ABC | 1.7478 | 10.2658 | 5.1245 | 2.3441 | 0 |
| $f_3$ | **DSOA** | **0** | **0** | **0** | **0** | **100** |
| | PSO | 4 | 176 | 33.60000 | 42.14810 | 0 |
| | BA | 34100 | 56520 | 48039.85 | 5839.602 | 0 |
| | ABC | 2 | 11 | 5.8000 | 2.3078 | 0 |
| $f_4$ | **DSOA** | **0** | **1.6843** | **0.0561** | **0.3075** | **3.3** |
| | PSO | 3.2862E−14 | 5.1580 | 2.6395 | 1.1256 | 0 |
| | BA | 18.5512 | 19.1996 | 18.8465 | 0.1487 | 0 |
| | ABC | 1.6844 | 1.7047 | 1.6893 | 0.0051 | 0 |
| $f_5$ | **DSOA** | **0** | **1.9712E−119** | **6.5709E−121** | **3.5990E−120** | **86** |
| | PSO | 1.0213E−12 | 1.8746E−10 | 4.1062E−11 | 4.9873E−11 | 0 |
| | BA | 7.9130E−04 | 1.3728E−03 | 1.0781E−03 | 1.5513E−4 | 0 |
| | ABC | 2.3812E−04 | 0.0044 | 0.0017 | 0.0010 | 0 |
| $f_6$ | **DSOA** | **0** | **0** | **0** | **0** | **100** |
| | PSO | 1.5142E−08 | 2.7651E−06 | 6.0061E−07 | 7.6616E−07 | 0 |
| | BA | 259.2022 | 377.0474 | 327.6945 | 29.45 | 0 |
| | ABC | 0.1534 | 0.5137 | 0.3397 | 0.0885 | 0 |
| $f_7$ | **DSOA** | **0** | **5.0678E−11** | **1.6892E−12** | **9.2526E−12** | **20** |
| | PSO | 1.4607E−07 | 4.0000E+04 | 4.0000E+03 | 1.2311E+04 | 0 |
| | BA | 0.0066 | 434.9910 | 29.6945 | 99.5714 | 0 |
| | ABC | 8.6357E−10 | 3.1342E−06 | 5.5644E−07 | 8.9392E−07 | 0 |
| $f_8$ | **DSOA** | **−1.2672E+04** | **0** | **−4.2241E+02** | **2.3136E+02** | **0** |
| | PSO | −9.6843E+03 | −7.1347E+03 | −8.5491E+03 | 6.8226E+03 | 0 |
| | BA | −10357.3703 | −6982.8810 | −7932.8280 | 860.0639 | 0 |
| | ABC | −9.4489E+03 | −7.8104E+03 | −8.3656E+03 | 3.6003E+02 | 0 |
| $f_9$ | **DSOA** | **0** | **8.1300E−22** | **2.7100E−23** | **1.4843E−22** | **20** |
| | PSO | 5.6696E−10 | 2.1008E+02 | 35.0501 | 55.5440 | 0 |
| | BA | 0.0012 | 0.0018 | 0.0015 | 0.0001 | 0 |
| | ABC | 2.7002E+02 | 3.6010E+02 | 3.1315E+02 | 22.6625 | 0 |
| $f_{10}$ | **DSOA** | **0** | **0** | **0** | **0** | **100** |
| | PSO | 0.0303 | 5.5521E+04 | 1.4177E+04 | 2.4170E+04 | 0 |
| | BA | 17.19 | 170.8199 | 40.318 | 36.8743 | 0 |
| | ABC | 48.0856 | 1.2072E+02 | 90.2872 | 19.9189 | 0 |
| $f_{11}$ | **DSOA** | **8.5164E−04** | **5.6823E−03** | **3.5416E−03** | **2.4617E−03** | 0 |
| | PSO | 1.3985E−01 | 8.6579E−01 | 3.7489E−01 | 1.9652E−01 | 0 |
| | BA | 0.0022 | 0.0106 | 0.0049 | 0.0020 | 0 |
| | ABC | 1.0072E−01 | 3.2423E−01 | 1.9354E−01 | 6.3320E−02 | 0 |
| $f_{12}$ | **DSOA** | **3.3238E−12** | **2.8579E−11** | **1.3134E−11** | **4.8303E−12** | 0 |
| | PSO | 2.7854E+02 | 4.5887E+02 | 3.5254E+02 | 3.7307E+01 | 0 |
| | BA | 8.1455E+01 | 1.2245E+02 | 1.0152E+02 | 1.0889E+01 | 0 |
| | ABC | 2.4836E+00 | 4.2465E+00 | 3.5531E+00 | 4.2125E−01 | 0 |
| $f_{13}$ | **DSOA** | **5.6092E−32** | **3.2645E−04** | **1.2890E−09** | **1.4978E−05** | 0 |
| | PSO | 1.0379E+01 | 2.7585E+01 | 1.7531E+01 | 3.8118E+00 | 0 |
| | BA | 3.7614E+01 | 6.3158E+01 | 5.0658E+01 | 5.8564E+00 | 0 |
| | ABC | 4.9856E−01 | 9.8461E−01 | 6.3578E−01 | 1.7997E−01 | 0 |

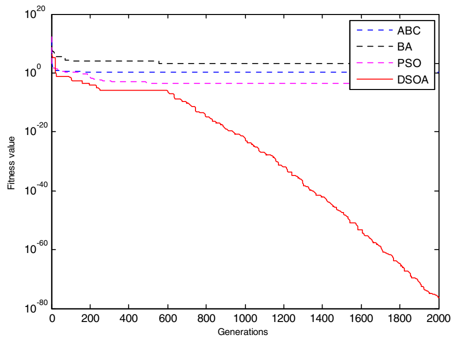
Table 1. (*Continued*)

| Function | Algorithm | **Best** | **Worst** | **Mean** | **Std.** | **SR** (%) |
|---|---|---|---|---|---|---|
| $f_{14}$ | **DSOA** | **2.2754E−06** | **1.8297E−02** | **6.4998E−04** | **3.3341E−04** | 0 |
| | PSO | 4.6620E+00 | 3.4065E+01 | 1.5764E+01 | 7.3964E+00 | 0 |
| | BA | 2.4164E+01 | 5.6412E+01 | 3.4791E+01 | 8.4030E+00 | 0 |
| | ABC | 1.2064E−03 | 9.1020E−03 | 3.4503E−03 | 2.1519E−03 | 0 |
| $f_{15}$ | **DSOA** | **9.9800E−01** | **9.9800E−01** | **9.9800E−01** | **0** | **100** |
| | PSO | 9.9811E−01 | 5.9288E+00 | 1.9881E+00 | 1.4208E+00 | 0 |
| | BA | 2.9821E+00 | 2.1073E+01 | 1.2266E+01 | 5.3019E+00 | 0 |
| | ABC | 1.0071E+01 | 1.1024E+01 | 1.1015E+01 | 3.4750E−5 | 0 |
| $f_{16}$ | **DSOA** | **3.0054E−04** | **8.6836E−04** | **5.6925E−04** | **2.9403E−04** | **3.3** |
| | PSO | 3.1869E−04 | 2.2853E−02 | 4.2643E−03 | 7.8306E−03 | 0 |
| | BA | 3.1784E−04 | 1.1289E−03 | 6.4587E−04 | 3.4902E−04 | 0 |
| | ABC | 3.3732E−04 | 1.8298E−03 | 6.3537E−04 | 3.8417E−04 | 0 |
| $f_{17}$ | **DSOA** | **−1.0153E+01** | **−7.6126E+00** | **−9.1561E+00** | **2.2973E+00** | **3.3** |
| | PSO | −3.8006E+00 | −3.2706E−01 | −1.0138E+00 | 8.0728E−01 | 0 |
| | BA | −1.0013E+01 | −2.6305E+00 | −4.8537E+00 | 2.6728E+00 | 0 |
| | ABC | −5.0537E+00 | −5.0414E+00 | −5.0503E+00 | 3.1115E−03 | 0 |
| $f_{18}$ | **DSOA** | **−1.0403E+01** | **−2.7659E+00** | **−1.0021E+01** | **1.7077E+00** | **3.3** |
| | PSO | −3.4175E+00 | −4.7227E−01 | −1.0652E+00 | 6.3916E−01 | 0 |
| | BA | −1.0364E+01 | −1.8376E+00 | −4.4590E+00 | 3.1423E+00 | 0 |
| | ABC | −5.0866E+00 | −5.0780E+00 | −5.0826E+00 | 2.1034E−03 | 0 |
| $f_{19}$ | **DSOA** | **−1.0536E+01** | **−2.8066E+00** | **−9.8819E+00** | **2.0509E+00** | **3.3** |
| | PSO | −2.0606E+00 | −6.2921E−01 | −1.0820E+00 | 0.376188689 | 0 |
| | BA | −5.1261E+00 | −5.1146E+00 | −5.1214E+00 | 2.8841E−03 | 0 |
| | ABC | −1.0143E+01 | −2.4216E+0 | −5.7129E+00 | 3.7127E+00 | 0 |
| | **DSOA** | **−3.3194E+00** | **−3.2031E+00** | **−3.2744E+00** | **5.9759E−02** | **3.3** |
| $f_{20}$ | PSO | −2.6983E+00 | −9.1020E−01 | −1.9266E+00 | 4.5145E−01 | 0 |
| | BA | −3.3189E+00 | −3.1032E+00 | −3.2504E+00 | 5.8134E−02 | 0 |
| | ABC | −3.3102E+00 | −1.1670E+00 | −3.0779E+00 | 4.9603E−01 | 0 |

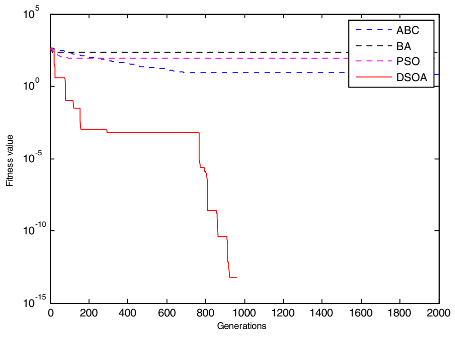
(PSO, ABC, and BA) do not find the optimal solution in the 30 trials for the 10 functions. On the functions $f_8$, $f_{11}$, $f_{12}$, $f_{13}$ and $f_{14}$, all of the four algorithms cannot find the best solution in the 30 runs, but the DSOA is apparently superior to that of the other three algorithms in terms of accuracy. For the worst solution (**Worst**) in Table 1, the DSOA is significantly surpasses the other three algorithms (PSO, ABC, and BA). From the mean results (**Mean**) and the standard deviation (**Std.**) in Table 1, we can see that the DSOA has the characteristics of strong robustness and good stability.

In summary, the comparisons of the experimental results in Table 1 show that the proposed new algorithm (DSOA) has better search ability than that of any other three algorithms. From the experimental results of the **Best**, **Worst**, **Mean**, **Std.**, and **SR** in Table 1, we observe that the DSOA obviously outperforms all the other three algorithms (PSO, ABC, and BA) on the 20 test functions.

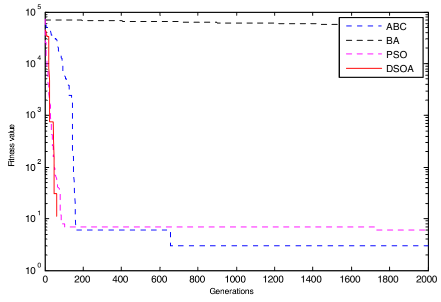
In order to investigate the convergence characteristics of the DSOA, Fig. 4 provides the convergence curves of the four algorithms (DSOA, PSO, ABC and BA) when solving these test functions, respectively. From Fig. 4, we can see that the



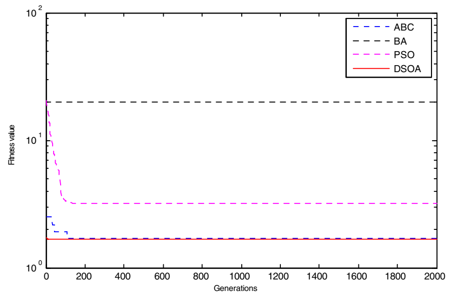
(a) $f_1$



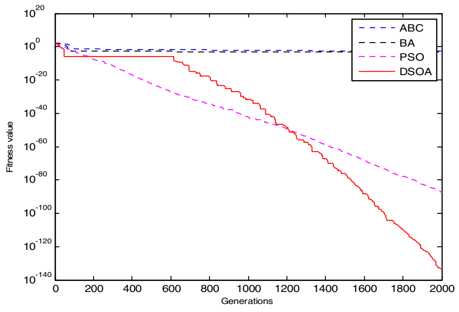
(b) $f_2$



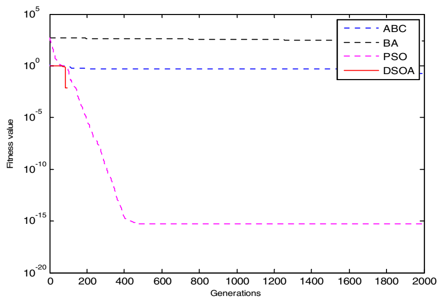
(c) $f_3$



(d) $f_4$

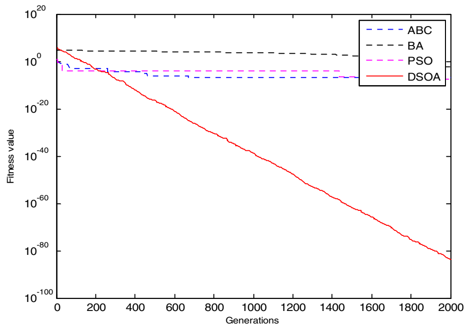


(e) $f_5$

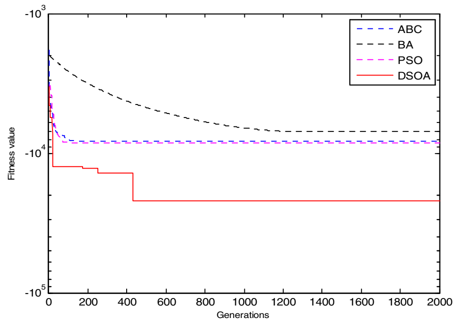


(f) $f_6$

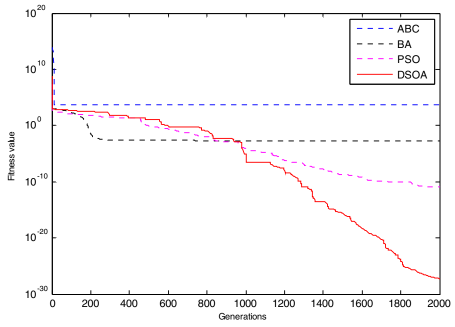
Fig. 4. Convergence characteristics of test functions.



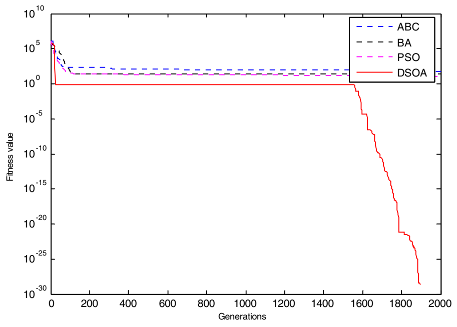
(g) $f_7$



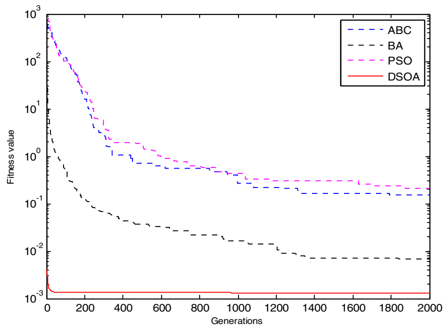
(h) $f_8$



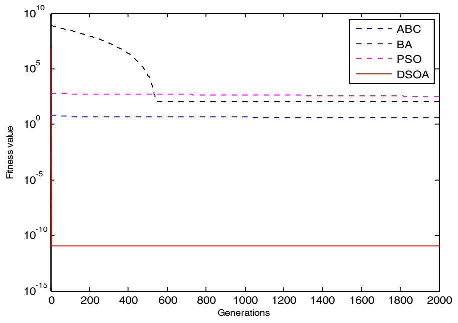
(i) $f_9$



(j) $f_{10}$

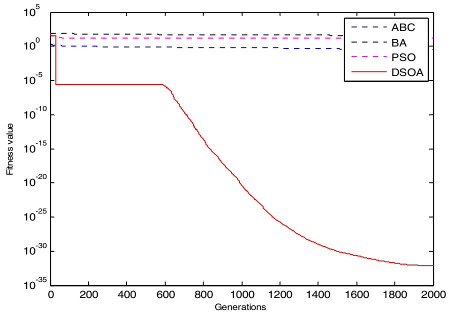


(k) $f_{11}$

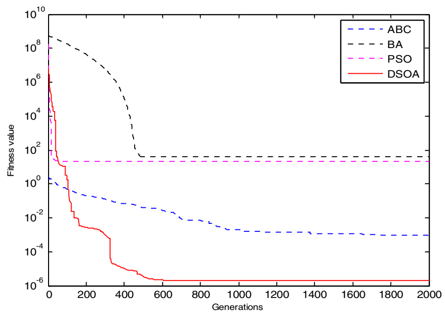


(l) $f_{12}$

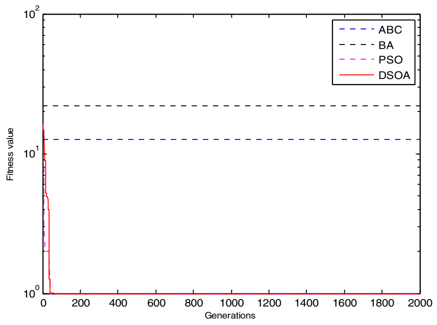
Fig. 4. (*Continued*)



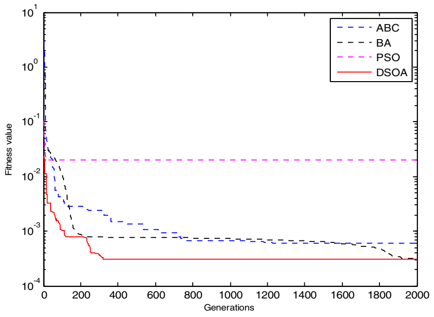
(m) $f_{13}$



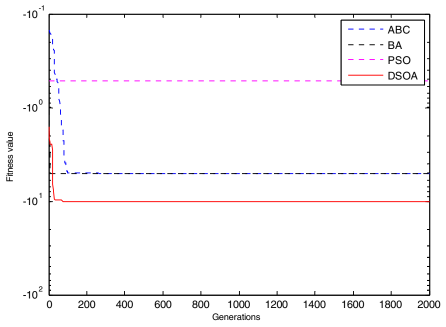
(n) $f_{14}$



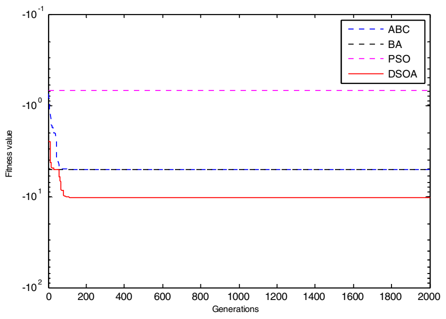
(o) $f_{15}$



(p) $f_{16}$

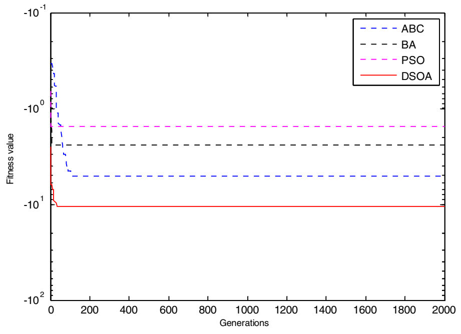


(q) $f_{17}$

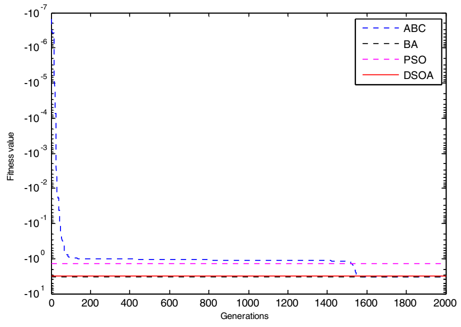


(r) $f_{18}$

Fig. 4. (*Continued*)



(s) $f_{19}$



(t) $f_{20}$

Fig. 4. (*Continued*)

DSOA has a very fast convergence rate, and converges faster than that of any other three algorithms (PSO, ABC, and BA).

In short, it is obvious that the proposed new algorithm (DSOA) is distinctly superior to the normal PSO, the original ABC, and the normal BA in terms of global search ability, convergence rate, accuracy, effectiveness and robustness. Moreover, from the comparisons of the experimental results and the comparisons of convergence rate, we also see that the DSOA is a very promising optimization algorithm, a powerful approach in solving global optimization problems, and is potentially more powerful than PSO algorithm and ABC algorithm as well as BA algorithm.

## 4. Conclusions

In this paper, we have successfully proposed a new nature-inspired optimization algorithm that we call the DSOA, which imitates the behavior of dolphins in chasing after, attacking or preying on swarms of sardines. In this new optimization algorithm, each dolphin cooperates with its companions and independently carries out itself search activities in the search space. Some dolphins are chasing after a swarm of sardines, and some are attacking or preying on a swarm of sardines meanwhile. Each dolphin can use two different swimming-modes when it searching food in its own search space. In order to validate the performance of the DSOA, the DSOA is tested against other classical optimization algorithms such as PSO, ABC, and BA. The test results show that the DSOA outperforms the normal PSO, the normal ABC and the normal BA in all these testing cases, that the DSOA has the characteristics of global search ability, fast convergence speed, strong robustness and good stability. The current studies indicated that the DSOA is a kind of potentially powerful optimization approaches and can be applied to solve various engineering optimization problems, combinatorial optimization problems, and so forth, although the DSOA is not necessarily the best choice for solving these optimization problems.

## Acknowledgments

This work is supported by the project of high school scientific research, Guangxi, China, under Grant (KY2015YB078).

## References

1. H. J. Holland, *Adaptation in Natural and Artificial Systems* (University of Michigan Press, Ann Arbor, MI, 1975).
2. J. Kennedy and R. C. Eberhart, Particle swarm optimization, in *Proc. IEEE Int. Conf. Neural Networks*, IV (IEEE Service Center, Piscataway, NJ, 1995), pp. 1942–1948.
3. M. Dorigo, V. Maniezzo and A. Colorni, Ant system: Optimization by a colony of cooperating agents, *IEEE Trans. Syst. Man Cybern. B Cybern.* **26** (1996) 29–41.
4. D. Karabora and B. Basturk, A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm, *J. Glob. Optim.* **39** (2007) 459–471.
5. X.-L. Li, Z.-J. Shao and J.-X. Qian, An optimizing method based on autonomous animats: Fish-swarm algorithm, *Syst. Eng. Theory Pract.* **22** (2002) 32–38 (in Chinese).
6. G. Theraulaz, E. Bonabeau and J. L. Deneubourg, Self-organization of hierarchies in animal societies: The case of the primitively eusocial wasp polistes dominulus Christ, *J. Theor. Biol.* **174** (1995) 313–323.
7. G. Theraulaz, E. Bonabeau and J. L. Deneubourg, Response threshold reinforcement and division of labour in insect societies, in *Proc. Roy. Soc. Lond. B* **265** (1998) 327–332.
8. M. M. Eusuff and K. E. Lansey, Optimization of water distribution network design using shuffled frog leaping algorithm, *J. Water Res. Plan. Mgnt.* **129** (2003) 210–225.
9. L. C. Jiao and L. Wang, Anovel genetic algorithm based on immunity, *IEEE Trans. Syst. Man Cybern.* **30** (2000) 552–561.
10. K. Krishnanand and D. Ghose, Glowworm swarm based optimization algorithm for multimodel functions with collective robotics applications, *Multiagent Grid Syst.* **2** (2006) 209–222.
11. X.-S. Yang and S. Deb, Cuckoo search via Levy flights, in *Proc. World Congr. Nature and Biologically Inspired Computing*, eds. A. Abraham, A. Carvalho, F. Herrera and V. Pai (IEEE Publications, USA, 2009), pp. 210–214.
12. R. Oftadeh, M. J. Mahjoob and M. Sharitatpanahi, A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search, *Comput. Math. Appl.* **60** (2010) 2087–2098.
13. X.-S. Yang, A new metaheuristic bat-inspired algorithm, in *Nature Inspired Cooperative Strategies for Optimization*, Vol. 284 (Springer, Studies in Computational Intelligence, 2010), pp. 65–74.
14. R. Tang, S. Fong, X.-S. Yang and S. Deb, Wolf search algorithm with ephemeral memory, in *Int. Conf. Digi. Inf. Mgnt.* 2012 IEEE, pp. 165–172. DOI: 10.1109/ICDIM.2012.6360147.
15. M. N. G. Sepidnam and M. Sargolzaei, Swallow swarm optimization algorithm: A new method to optimization, *Neural Comput. Appl.* **23** (2013) 429–454.
16. S. Moein and R. Logeswaran, KGMO: A swarm optimization algorithm based on the kinetic energy of gas molecules, *Inf. Sci.* **275** (2014) 127–144.
17. X. Li, J. Zhang and M. Yin, Animal migration optimization: An optimization algorithm inspired by animal migration behavior, *Neural Comput. Appl.* **24** (2014) 1867–1877.
18. D. Simon, Biogeography-based optimization, *IEEE Trans. Evol. Comput.* **12** (2008) 702–713.
19. B. Javidy, A. Hatamlou and S. Mirjalili, Ions motion algorithm for solving optimization problems, *Appl. Soft Comput.* **32** (2015) 72–79.

20. S. Mirjalili, M. S. Mirjalili and A. Lewis, Grey wolf optimizer, *Adv. Eng. Softw.* **69** (2014) 46–61.
21. S. Mirjalili, The ant lion optimizer, *Adv. Eng. Softw.* **83** (2015) 80–98.
22. X.-S. Yang, Firefly algorithm, stochastic test functions and design optimization, *Int. J. Bio-Inspired Comput.* **2** (2010) 78–84.
23. A. H. Gandomi and A. H. Alavi, Krill herd: A new bio-inspired optimization algorithm, *Commun. Nonlinear Sci. Numer. Simul.* **17** (2012) 4831–4845.
24. A. Hatamlou, Heart: A novel optimization algorithm for cluster analysis, *Prog. Artif. Intell.* **2** (2014) 167–173.
25. X. Meng, Y. Liu, X. Gao and H. Zhang, A new bio-inspired algorithm: Chicken swarm optimization, in *Advances in Swarm Intelligence*, Lecture notes in Computer Science, Vol. 8794 (Springer, New York, 2014), pp. 86–94.
26. Z. Cui and X. Gao, Theory and applications of swarm intelligence, *Neural Comput. Appl.* **21** (2012) 205–206.
27. T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms* (Oxford University Press, Oxford, 1996).
28. D. Hand, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press, Cambridge, 1992).
29. R. Storn and K. Price, Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optim.* **11** (1997) 341–359.
30. X. Yao, Y. Liu and G. Lin, Evolutionary programming made faster, *IEEE Trans. Evol. Comput.* **3** (1999) 82–102.
31. E. Rashedi, H. Nezamabadi-Pour and S. Saryazdi, GSA: A gravitational search algorithm, *Inf. Sci.* **179** (2009) 2232–2248.
32. A. Kaveh and M. Khayatazad, A new meta-heuristic method: Ray optimization, *Comput. Struct.* **112** (2012) 283–294.
33. E. Cuevas, A. Echavarría and M. A. Ramírez-Ortegón, An optimization algorithm inspired by the states of matter that improves the balance between exploration and exploitation, *Appl. Intell.* **40** (2014) 256–272.
34. A. Hatamlou, Black hole: A new heuristic optimization approach for data clustering, *Inf. Sci.* **222** (2013) 175–184.
35. B. Alatas, ACROA: Artificial chemical reaction optimization algorithm for global optimization, *Expert Syst. Appl.* **38** (2011) 13170–13180.
36. D. H. Wolpert and W. G. Macready, No free lunch theorems for optimization, *IEEE Trans. Evol. Comput.* **1** (1997) 67–82.
37. X.-H. Tian, *The Mystery of Animal World* (Guangming Daily Press, 2015) (in Chinese).