algorithms

MDPI

Article

Elite Opposition-Based Social Spider Optimization Algorithm for Global Function Optimization

Ruxin Zhao [1], Qifang Luo [1,2] and Yongquan Zhou [1,2,*]

[1] School of Information Science and Engineering, Guangxi University for Nationalities, Nanning 530006, China; nnzhaoms@126.com (R.Z.); luoqifang@mail.gxun.edu.cn (Q.L.)

[2] Key Laboratories of Guangxi High Schools Complex System and Computational Intelligence, Nanning 530006, China

* Correspondence: yongquanzhou@126.com; Tel.: +86-771-326-0264

Academic Editors: Yun-Chia Liang, Mehmet Fatih Tasgetiren and Quan-Ke Pan
Received: 27 November 2016; Accepted: 4 January 2017; Published: 8 January 2017

Abstract: The Social Spider Optimization algorithm (SSO) is a novel metaheuristic optimization algorithm. To enhance the convergence speed and computational accuracy of the algorithm, in this paper, an elite opposition-based Social Spider Optimization algorithm (EOSSO) is proposed; we use an elite opposition-based learning strategy to enhance the convergence speed and computational accuracy of the SSO algorithm. The 23 benchmark functions are tested, and the results show that the proposed elite opposition-based Social Spider Optimization algorithm is able to obtain an accurate solution, and it also has a fast convergence speed and a high degree of stability.

Keywords: social spider optimization; elite opposition-based learning; elite opposition-based social spider optimization; function optimization

1. Introduction

A swarm intelligence optimization algorithm originates from the simulation of various types of biological behavior in nature and has the characteristics of simple operation, good optimization performance and strong robustness. Inspired by this idea, many bio-inspired swarm intelligent optimization algorithms are proposed, such as, ant colony optimization (ACO) [1], differential evolution (DE) [2], particle swarm optimization (PSO) [3], firefly algorithm (FA) [4], glowworm swarm optimization (GSO) [5], monkey search (MS) [6], harmony search (HS) [7], cuckoo search (CS) [8,9], bat algorithm (BA) [10], krill herd algorithm (KH) [11–13], Swarm intelligence optimization algorithm can solve problems which traditional methods cannot handle effectively and have shown excellent performance in many respects, and its application scope has been greatly expanded.

The Social Spider Optimization algorithm (SSO), proposed by Erik Cuevas in 2013 [14], is a novel metaheuristic optimization algorithm that simulates social-spider behavior. Although SSO has obtained good performance on many optimization problems, it still falls easily into a local optimal solution. In order to enhance the performance of SSO on optimization problems, this paper presents a novel SSO algorithm called EOSSO by using OBL and elite selection mechanism. Opposition-based Learning (OBL) is a new concept in computational intelligence, many algorithms have used the OBL mechanism [15,16], and it has been proven to be an effective strategy to improve performance of various optimization algorithms. The main idea behind OBL is to transform solutions in the current search space to a new search space. By simultaneously considering the solutions in the current search space and the transformed search space, OBL can provide a higher chance of finding solutions which are closer to the global optimum. However, OBL could not be suitable for all kinds of optimization problems. For instance, the transformed candidate may jump away from the global optimum when solving multimodal problems. To avoid this case, a new elite selection mechanism

based on the population is used after the transformation. The proposed elite opposition-based Social Spider Optimization algorithm is validated by 23 benchmark functions. The results show that the proposed algorithm is able to obtained accurate solution, and it also has a fast convergence speed and a high degree of stability.

The rest of the paper is organized as follows. Section 2 introduces the original Social Spider Optimization algorithm (SSO). Section 3 proposes a new elite opposition-based Social Spider Optimization algorithm (EOSSO). A series of comparison experiments on various benchmarks in Section 4. The results analysis is described in Section 5. Finally, conclusion and future works can be found and discussed in Section 6.

2. The Social Spider Optimization Algorithm

The SSO algorithm is divided into two different search agents (spiders): males and females. Depending on gender, each individual is conducted by a set of different evolutionary operators which mimic different cooperative behaviors that are commonly assumed within the colony. The SSO algorithm starts by defining the number of female and male spiders that will be characterized as individuals in the search space. The number of females (N_f) is randomly selected within the range of 65%–90% of the entire population (N). Therefore, N_f is calculated by the following equation:

$$N_f = floor[(0.9 - rand * 0.25) * N] \tag{1}$$

where, *rand* is a random number in the range [0, 1], whereas $floor(.)$ maps a real number to an integer number. The number of male spiders (N_m) is computed as the complement between N and N_f. It is calculated as follows:

$$N_m = N - N_f \tag{2}$$

where, the population (S) is composed by N elements and is divided into sub-groups F and M. The group (F) include the set of female individuals ($F = \left\{f_1, f_2, \ldots, f_{N_f}\right\}$) and the group ($M$) include the set of male individuals ($M = \{m_1, m_2, \ldots, m_{N_m}\}$), where $S = F \cup M$ ($S = \{s_1, s_2, \ldots, s_N\}$), such that $S = \left\{s_1 = f_1, s_2 = f_2, \ldots, s_{N_f} = f_{N_f}, s_{N_f+1} = m_1, s_{N_f+2} = m_2, \ldots, S_N = m_{N_m}\right\}$.

2.1. Fitness Assignation

In natural metaphor, the spider size is the characteristic that evaluates the individual capacity, every spider receives a weight (w_i) which represents the solution quality that corresponds to the spider (i) of the population (S). The weight of every spider is defined:

$$w_i = \frac{J(s_i) - worst_s}{best_s - worst_s} \tag{3}$$

where, $J(s_i)$ is the fitness value obtained by the evaluation of the spider position (s_i) with regard to the objective function ($J(.)$). The values of $worst_s$ and $best_s$ are defined as follows (considering a maximization problem):

$$best_s = \max_{k \in \{1,2,\ldots,N\}}(J(s_k)) \text{ and } worst_s = \min_{k \in \{1,2,\ldots,N\}}(J(s_k)) \tag{4}$$

2.2. Modeling of the Vibrations through the Communal Web

The communal web is used as a mechanism to transmit information among the colony members. The vibrations depend on the weight and distance of the spider which has generated them. The vibrations perceived by the individual (i) as a result of the information transmitted by the member (j) are modeled as the following equation:

$$Vib_{i,j} = w_j * e^{-d_{i,j}^2} \tag{5}$$

where, $d_{i,j}$ is the Euclidian distance between the spiders i and j, such that $d_{i,j} = ||s_i - s_j||$.

Although it is virtually possible to compute perceived vibrations by considering any pair of individuals, but only three special relationships are considered within the SSO approach:

(1) Vibrations ($Vibc_i$) are perceived by the individual (i) as a result of the information transmitted by the member (c) who is an individual that has two characteristics: it is the nearest member to individual (i) and possesses a higher weight in comparison to individual (i) ($w_c > w_i$).

$$Vibc_i = w_c * e^{-d_{i,c}^2} \tag{6}$$

(2) The vibrations ($Vibb_i$) perceived by the individual (i) as a result of the information transmitted by the member (b), with individual (b) being the individual holding the best weight (best fitness value) of the entire population (S), such that $w_b = \max_{k \in \{1,2,\ldots,N\}}(w_k)$.

$$Vibb_i = w_b * e^{-d_{i,b}^2} \tag{7}$$

(3) The vibrations ($Vibf_i$) perceived by the individual (i) as a result of the information transmitted by the member (f), with individual (f) being the nearest female individual to individual (i).

$$Vibf_i = w_f * e^{-d_{i,f}^2} \tag{8}$$

2.3. Initializing the Population

The SSO algorithm begins by initializing the set (S) of N social-spider positions. Each social-spider position, f_i or m_i, is an n-dimensional vector containing the parameter values to be optimized. Such values are randomly and uniformly distributed between the pre-specified lower initial parameter bound (p_j^{low}) and the upper initial parameter bound (p_j^{high}), just as it described by the following expressions:

$$f_{k,j}^0 = p_j^{low} + \underset{k=1,2,\ldots,N_f; j=1,2,\ldots,n}{rand(0,1)} * (p_j^{high} - p_j^{low}) m_{k,j}^0 = p_j^{low} + \underset{k=1,2,\ldots,N_m; j=1,2,\ldots,n}{rand(0,1)} * (p_j^{high} - p_j^{low}) \tag{9}$$

where k and j are the individual indexes, whereas zero signals the initial population. The function ($rand(0,1)$) generates a random number between 0 and 1.

2.4. Cooperative Operators

2.4.1. Female Cooperative Operator

To emulate the cooperative behavior of the female spider, the operator considers the position change of the female spider (i) at each iteration process. The position change is computed as a combination of three different elements. The first one involves the change in regard to the nearest member to individual (i) that holds a higher weight and produces the vibration ($Vibc_i$). The second one considers the change regarding the best individual of the entire population (S) who produces the vibration ($Vibb_i$). Finally, the third one incorporates a random movement.

For this operation, a uniform random number (r_m) is generated within the range [0, 1]. If random number (r_m) is smaller than a threshold (PF), an attraction movement is generated; otherwise, a repulsion movement is produced. Therefore, such operator can be modeled as follows:

$$f_i^{k+1} = \begin{cases} f_i^k + \alpha * Vibc_i * (s_c - f_i^k) + \beta * Vibb_i * (s_b - f_i^k) + \delta * (rand - \frac{1}{2}); r_m < PF \\ f_i^k - \alpha * Vibc_i * (s_c - f_i^k) - \beta * Vibb_i * (s_b - f_i^k) + \delta * (rand - \frac{1}{2}); r_m > PF \end{cases} \quad (10)$$

where, α, β, δ and $rand$ are random numbers between 0 and 1, whereas k represents the iteration number. The individual (s_c) represent the nearest and holds a higher weight of member to individual (i), the individual (s_s) represent that the social-spider with a highest weight in the entire population (S).

2.4.2. Male Cooperative Operator

In the natural, male population of social-spider is divided into two classes: dominant members (D) and non-dominant members (ND) of male spiders. Male members, with a weight value above the median value within the male population, are considered the dominant individuals (D). On the other hand, those under the median value are labeled as non-dominant (ND) males.

In order to implement this computation, the male population ($M = \{m_1, m_2, \ldots, m_{N_m}\}$) is arranged according to their weight value in decreasing order. Thus, the individual whose weight (w_{N_f+m}) is located in the middle is considered the median male member. Since indexes of the male population (M) in regard to the entire population (S) are increased by the number of female members (N_f), the median weight is indexed by $N_f + m$. According to this, the male spider positions change as follows:

$$m_i^{k+1} = \begin{cases} m_i^k + \alpha * Vibf_i * (s_f - m_i^k) + \delta * (rand - \frac{1}{2}); w_{N_f+i} > w_{N_f+m} \\ m_i^k + \alpha * (\frac{\sum_{h=1}^{N_m} m_h^k * w_{N_f+h}}{\sum_{h=1}^{N_m} w_{N_f+h}} - m_i^k); w_{N_f+i} \leq w_{N_f+m} \end{cases} \quad (11)$$

where, the individual (s_f) represents the nearest female individual to the male member (i), whereas median ($\sum_{h=1}^{N_m} m_h^k * w_{N_f+h} \Big/ \sum_{h=1}^{N_m} w_{N_f+h}$) correspond to the weighted mean of the male population (M).

2.5. Mating Operator

Mating in a social-spider colony is performed by dominant males and the female members. Under such circumstances, when a dominant male (m_g) spider ($g \in D$) locates a set of female members (E^g) within range of mating (r), it mates, forming a new brood (s_{new}) which is generated considering all the elements of the set (T^g), which has been generated by the union ($E^g \cup m_g$). It is emphasized that if the set (E^g) is empty, the mating operation is canceled. The range (r) is defined as a radius which depends on the size of the search space. Such radius (r) is computed according to the following model:

$$r = \frac{\sum_{j=1}^{n} p_j^{high} - p_j^{low}}{2n} \quad (12)$$

In the mating process, the weight of each involved social-spider (elements of T^g) defines the probability of influence for each individual into the new brood. The social-spider holding a heavier weight is more likely to influence the new product, while elements with lighter weight have a lower probability. The influence probability (Ps_i) of each member is assigned by the roulette method, which is defined as follows:

$$Ps_i = \frac{w_i}{\sum_{j \in T^g} w_j} \quad (13)$$

where $i \in T^g$.

Once the new spider is formed, it is compared to holding the worst spider (s_{wo}) of the colony, according to their weight values (where $w_{wo} = \min_{l \in \{1,2,\ldots,N\}}(w_l)$). If the new spider is better than the worst spider, the worst spider is replaced by the new one. Otherwise, the new spider is discarded and the population does not suffer changes.

2.6. Computational Procedure

The computational procedure for the Algorithm 1 can be summarized as follow:

Algorithm 1: The Social spider optimization algorithm

Step 1: Think N as the total number of n-dimensional colony members, define the number of male (N_m) and females spiders (N_f) in the entire population (S).

$$N_f = floor[(0.9 - rand * 0.25) * N],\ N_m = N - N_f$$

Where $rand$ is a random number in the range [0, 1], whereas $floor(.)$ maps a real number to an integer number.

Step 2: Initialize randomly the female members ($F = \left\{f_1, f_2, \ldots, f_{N_f}\right\}$), male members ($M = \{m_1, m_2, \ldots, m_{N_m}\}$) and calculate the radius of mating.

$$r = \frac{\sum_{j=1}^{n} p_j^{high} - p_j^{low}}{2n}$$

Step 3: Calculation the weight of every spider of S.

For (i =1; $i < N$ +1; i++)

$w_i = \frac{J(s_i) - worst_s}{best_s - worst_s}$, where $best_s = \max_{k \in \{1,2,\ldots,N\}}(J(s_k))$ and $worst_s = \min_{k \in \{1,2,\ldots,N\}}(J(s_k))$

End For

Step 4: Female spider's movement according to the female cooperative operator.

For (i =1; $i < N_f$ +1; i++)

Calculate $Vibc_i$ and $Vibb_i$

If ($r_m < PF$), where $r_m \in rand(0, 1)$

$f_i^{k+1} = f_i^k + \alpha * Vibc_i * (s_c - f_i^k) + \beta * Vibb_i * (s_b - f_i^k) + \delta * (rand - \frac{1}{2})$

Else If

$f_i^{k+1} = f_i^k - \alpha * Vibc_i * (s_c - f_i^k) - \beta * Vibb_i * (s_b - f_i^k) + \delta * (rand - \frac{1}{2})$

End If

End For

Step 5: Move the male spiders according to the male cooperative operator

Find the median male individual (w_{N_f+m}) from M.

For (i =1; $i < N_m$ +1; i++)

Calculate $Vibf_i$

If ($w_{N_f+i} > w_{N_f+m}$)

$m_i^{k+1} = m_i^k + \alpha * Vibf_i * (s_f - m_i^k) + \delta * (rand - \frac{1}{2})$

Else If

$m_i^{k+1} = m_i^k + \alpha * (\frac{\sum_{h=1}^{N_m} m_h^k * w_{N_f+h}}{\sum_{h=1}^{N_m} w_{N_f+h}} - m_i^k)$

End If

End For

Step 6: Perform the mating operation

For (i =1; $i < N_m$ +1; i++)

If ($m_i \in D$)

Find E^i

If (E^i is not empty)

From s_{new} using the roulette method

If ($w_{new} > w_{wo}$)

$s_{wo} = s_{new}$

End If

End If

End If

End For

Step 7: if the stop criteria is met, the process is finished; otherwise, go back to Step 3.

3. Elite Opposition-Based Social Spider Optimization Algorithm (EOSSO)

When the SSO calculates unimodal function and multimodal function, we can clearly discover that the solutions obtained by SSO are not good enough. In order to enhance accuracy of the algorithm, we append one optimization strategy to SSO. There is global elite opposition-based learning strategy (GEOLS). In this part, we introduce opposition-based learning (OBL) and global elite opposition based learning strategy (GEOLS).

3.1. Opposition Based Learning (OBL)

OBL is, basically, a machine intelligence strategy which was proposed by Tizhoosh in [17]. It considers the current individual and its opposite individual simultaneously in order to get a better approximation at the same time for a current candidate solution. It has been also proved that an opposite candidate solution has a greater opportunity to be closer to the global optimal solution than a random candidate solution [17]. Therefore, the concept of OBL has been utilized to enhance population based algorithms in [18–21]. The general, OBL concept has been, successfully, applied in some areas of research work such as in reinforcement learning [22], window memorization for morphological algorithms [23], image processing using the opposite fuzzy sets [24,25] and also in some popular optimization techniques like ant colony optimization [26–28], GA [29], artificial neural networks with opposite transfer function and back propagation [30,31], DE, PSO with Cauchy mutation [32], gravitational search algorithm [33], harmonic search algorithm [34], and BBO [35,36]. In proposing this technique, some definitions are clearly defined below:

A. **Opposite number**:

Let $p \in [x, y]$ be a real number. The opposite number of $p(p^*)$ is defined by:

$$p^* = x + y - p \tag{14}$$

B. **Opposite point**:

Let $p = (s_1, s_2, \ldots, s_n)$ be a point in n-dimensional search space, where $p_r \in [x_r, y_r]$ and $r = \{1, 2, \ldots, n\}$. The opposite point is defined by

$$p_r^* = x_r + y_r - p_r \tag{15}$$

C. **Opposition based population initialization**:

By utilizing opposite points, a suitable starting candidate solution may be obtained even when there is not a priori knowledge about the solution. The main steps of the proposed approach are listed as follows:

Step 1 Initialize the population set in a random manner.

Step 2 Calculate opposite population by:

$$op_{a,b} = x_b + y_b - p_{a,b} \tag{16}$$

where $a = 1, 2, \ldots, N$, $b = 1, 2, \ldots, n$ and $p_{a,b}$ and $op_{a,b}$ denote the bth variable of the ath vector of the population and opposite population, respectively.

Step 3 Select the fittest N individuals from $\{p \cup op\}$ as initial population.

D. **Opposition based generation jumping**

If we apply similar approach to the current population, the whole evolutionary process can be forced to jump to a new solution candidate who is more suitable than the current one. From this

comparison, the fittest N individuals are selected. In each generation, search space is reduced to calculate the opposite points, i.e.,

$$op_{a,b} = Min_b^{gn} + Max_b^{gn} - p_{a,b} \tag{17}$$

where $a = 1, 2, \ldots, N$ and $b = 1, 2, \ldots, n$. $[Min_b^{gn}, Max_b^{gn}]$ is the current interval in the population which is becoming increasingly smaller than the corresponding initial range $[x_b, y_b]$.

3.2. Global Elite Opposition-Based Learning Strategy (GEOLS)

The Social Spider Optimization algorithm use cooperation among female groups in global search process. It is simulated by changes in the position of female spiders. As we know that it is a stochastic process, the probability of getting a good solution is relatively low. For increasing the probability of obtained a better solution to the problem in global search process and expand the searching space, this strategy is applied to the proposed EOSSO.

Elite opposition-based Learning is a new technique in the field of intelligence computation. Its main ideology is: for a feasible solution, calculate and evaluate the opposite solution at the same time, and choose the better one as the individual of next generation. In this paper, individual with the best fitness value in the population is viewed as the elite individual. For explaining the definition of elite opposition-based solution, an example should be exhibited. If we suppose that the elite individual of the population is $X_e = (x_{e,1}, x_{e,2}, \ldots, x_{e,n})$. For the individual $X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n})$, the elite opposition-based solution of X_i which can be defined as $X_i^* = (x_{i,1}^*, x_{i,2}^*, \ldots, x_{i,n}^*)$. In addition, it can be obtained by following equation:

$$x_{i,j}^* = k(da_j + db_j) - x_{e,j}, i = 1, 2, \ldots, N; j = 1, 2, \ldots, n. \tag{18}$$

where N is the population size, n is the dimension of X, $k \in U(0, 1)$ and $(da_j db_j)$ is the dynamic bound of jth decision variable. da_j, db_j can be obtained by following equation:

$$da_j = \min(x_{i,j}), db_j = \max(x_{i,j}) \tag{19}$$

As we know that the shrink of searching space may cause algorithm stuck in local minimal. Thus, in this proposed algorithm, we will update da_j and db_j every 50 generations. Dynamic bound is good at restoring searching experience. However, it can make $x_{i,j}^*$ jump out of $(da_j db_j)$, if that happened, equation below should be used to reset $x_{i,j}^*$.

$$x_{i,j}^* = rand(da_j, db_j), \text{ if } x_{i,j}^* < da_j \text{ or } x_{i,j}^* > db_j \tag{20}$$

In global searching process, this strategy expands the searching space of algorithm and it can strengthen the diversity of the population, thus the proposed global searching ability can be enhanced by this optimization strategy.

4. Simulation Experiments

In this section, 23 benchmark test functions [37] are applied to evaluate the optimal performance of EOSSO. The space dimension, scope and optimal value of 23 functions are shown in Tables 1–3. The rest of this section is organized as follows: experimental setup is given in Section 4.1 and the comparison of each algorithm performance is shown in Section 4.2.

Table 1. Unimodal benchmark function.

Function	Dim	Range	$f_{\min}$
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, 10]$	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j\right)^2$	30	$[-100, 100]$	0
$f_4(x) = \max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	30	$[-100, 100]$	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]$	0
$f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	$[-100, 100]$	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0.1)$	30	$[-1.28, 1.28]$	0

Table 2. Multimodal benchmark function.

Function	Dim	Range	$f_{\min}$
$f_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	$[-500, 500]$	−418.9829*5
$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]$	0
$f_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	30	$[-32, 32]$	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]$	0
$f_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\}$ $+\sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m; x_i > a \\ 0; -a < x_i < a \\ k(-x_i - a)^m; x_i < a \end{cases}$	30	$[-50, 50]$	0
$f_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + 10\sin^2(3\pi x_i + 1)]\right.$ $\left. + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	$[-50, 50]$	0

Table 3. Fixed-dimension multimodal benchmark function.

Function	Dim	Range	$f_{\min}$
$f_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})}\right)^{-1}$	2	$[-65, 65]$	1
$f_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	$[-5, 5]$	0.00030
$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]$	−1.0316
$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	$[-5, 5]$	0.398
$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30$ $+(2x_1 - 3x_2)^2 * \times(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$[-2, 2]$	3
$f_{19}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	3	$[1, 3]$	−3.86
$f_{20}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	6	$[0, 1]$	−3.32
$f_{21}(x) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	−10.1532
$f_{22}(x) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	−10.4028
$f_{23}(x) = -\sum_{i=1}^{10}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	−10.5363

The benchmark functions selected can be divided into three categories (i.e., unimodal benchmark functions, multimodal benchmark functions and fixed-dimension multimodal benchmark functions).

They are f_1–f_7 for unimodal benchmark functions, f_8–f_{13} for multimodal benchmark functions and f_{14}–f_{23} for fixed-dimension multimodal benchmark functions.

4.1. Experimental Setup

The proposed elite opposition-based Social Spider Optimization algorithm (EOSSO) compared with ABC, BA, DA, GGSA and SSO, respectively using the mean and standard deviation to compare their optimal performance. The parameters settings of algorithms are as follows: for all the optimization algorithms, population size N = 50, the iteration number is 1000 and execute 30 independent experiments. All of algorithms were programmed in MATLAB R2012a, simulated with an Inter (R) Core(TM) i5-4590 CPU and 4 GB memory.

4.2. Comparison of Each Algorithm Performance

The 30 independent runs were made for the six algorithms, the results obtained by six algorithms are presented in Tables 4–6. The evolution curves and the variance diagram of $f_1, f_4, f_7, f_9, f_{10}, f_{11}, f_{16}, f_{18}, f_{19}$ and f_{23} obtained by six algorithms are presented in Figures 1–10 and Figures 11–20, respectively.

In Tables 4–6, The "Function" represents test function, "Dim" represents dimension size, each number in the column "mean" is the average global optimal value of 30 time independent operation, the "best" is the best global optimal value of 30 time independent operation, the "worst" is the worst global optimal value of 30 time independent operation, each number in the column "std." represents standard deviation value of 30 time independent operation.

Table 4. Simulation results for test unimodal benchmark function.

Function	Dim	Algorithm	Best	Worst	Mean	std.
f_1	30	ABC	3.69×10^{-8}	2.32×10^{-6}	5.34×10^{-7}	5.40×10^{-7}
		BA	4.58×10^{2}	5.79×10^{3}	3.13×10^{3}	1.39×10^{3}
		GGSA	4.45×10^{-20}	4.17×10^{-19}	1.72×10^{-19}	9.49×10^{-20}
		DA	73.2	1.54×10^{3}	5.50×10^{2}	3.37×10^{2}
		SSO	8.63×10^{-2}	8.63×10^{-2}	8.63×10^{-2}	0
		EOSSO	3.53×10^{-67}	3.53×10^{-67}	3.53×10^{-67}	2.01×10^{-82}
f_2	30	ABC	9.47×10^{-6}	6.56×10^{-5}	2.75×10^{-5}	1.27×10^{-5}
		BA	1.14×10^{2}	2.93×10^{9}	1.03×10^{8}	5.35×10^{8}
		GGSA	8.97×10^{-10}	3.42×10^{-9}	1.88×10^{-9}	5.72×10^{-10}
		DA	5.81×10^{-3}	20.5	8.44	52.2
		SSO	1.20	1.20	1.20	2.26×10^{-16}
		EOSSO	1.92×10^{-38}	1.92×10^{-38}	1.92×10^{-38}	1.33×10^{-53}
f_3	30	ABC	1.14×10^{4}	2.45×10^{4}	1.97×10^{5}	3.08×10^{3}
		BA	4.33×10^{3}	1.71×10^{4}	9.12×10^{3}	3.17×10^{3}
		GGSA	1.32×10^{2}	5.37×10^{2}	2.68×10^{2}	90.5
		DA	3.60×10^{2}	1.49×10^{4}	5.39×10^{3}	3.87×10^{3}
		SSO	1.98	1.98	19.8	1.13×10^{-15}
		EOSSO	7.61×10^{-76}	7.61×10^{-76}	7.61×10^{-76}	6.24×10^{-91}
f_4	30	ABC	59.6	73.9	67.8	36.2
		BA	42.9	68.9	56.0	65.4
		GGSA	5.25×10^{-10}	1.96×10^{-9}	1.24×10^{-9}	4.04×10^{-10}
		DA	7.38	24.1	14.8	43.9
		SSO	1.47×10^{-1}	1.47×10^{-1}	1.47×10^{-1}	5.56×10^{-17}
		EOSSO	1.20×10^{-37}	1.20×10^{-37}	1.20×10^{-37}	2.12×10^{-53}

Table 4. *Cont.*

Function	Dim	Algorithm	Best	Worst	Mean	std.
f_5	30	ABC	767	89.3	37.9	20.7
		BA	243	5.12×10^{2}	59.4	92.4
		GGSA	255	1.18×10^{2}	33.0	22.0
		DA	30.9	2.36×10^{5}	3.41×10^{4}	5.38×10^{4}
		SSO	35.4	35.4	35.4	0
		EOSSO	26.8	26.8	26.8	1.08×10^{-14}
f_6	30	ABC	3.43×10^{-8}	6.92×10^{-6}	1.01×10^{-6}	1.67×10^{-6}
		BA	6.99×10^{2}	5.71×10^{3}	2.82×10^{3}	1.30×10^{3}
		GGSA	0	0	0	0
		DA	1.20×10^{2}	1.56×10^{3}	4.22×10^{2}	3.16×10^{2}
		SSO	1.08×10^{-1}	1.08×10^{-1}	1.08×10^{-1}	1.41×10^{-17}
		EOSSO	5.01×10^{-1}	5.01×10^{-1}	5.01×10^{-1}	2.26×10^{-16}
f_7	30	ABC	4.19×10^{-1}	10.2	7.09×10^{-1}	1.58×10^{-1}
		BA	6.93×10^{-3}	2.09×10^{-2}	1.23×10^{-2}	3.65×10^{-3}
		GGSA	3.90×10^{-3}	1.39×10^{-1}	1.73×10^{-2}	2.37×10^{-2}
		DA	1.75×10^{-2}	3.25×10^{-1}	1.27×10^{-1}	8.00×10^{-2}
		SSO	1.68×10^{-2}	1.68×10^{-2}	1.68×10^{-2}	0
		EOSSO	1.22×10^{-4}	1.22×10^{-4}	1.22×10^{-4}	5.51×10^{-20}

Table 5. Simulation results for test multimodal benchmark function.

Function	Dim	Algorithm	Best	Worst	Mean	std.
f_8	30	ABC	-1.21×10^{4}	-1.10×10^{4}	-1.16×10^{4}	2.81×10^{2}
		BA	-8.45×10^{3}	-5.97×10^{3}	-7.06×10^{3}	7.40×10^{2}
		GGSA	-3.93×10^{3}	-2.17×10^{3}	-3.08×10^{3}	3.64×10^{2}
		DA	7.38	24.1	14.8	4.39
		SSO	-7.61×10^{3}	-7.61×10^{3}	-7.61×10^{3}	2.78×10^{-12}
		EOSSO	-7.69×10^{3}	-7.69×10^{3}	-7.69×10^{3}	3.70×10^{-12}
f_9	30	ABC	1.02	11.6	5.47	2.9
		BA	1.48×10^{2}	2.51×10^{2}	1.99×10^{2}	26.3
		GGSA	7.95	23.8	16	4.57
		DA	34.3	1.99×10^{2}	1.14×10^{2}	49.9
		SSO	63.3	63.3	63.3	0
		EOSSO	0	0	0	0
f_{10}	30	ABC	9.16×10^{-5}	1.58×10^{-3}	4.79×10^{-4}	3.79×10^{-4}
		BA	18.5	19.9	19	2.50×10^{-1}
		GGSA	1.48×10^{-10}	5.26×10^{-10}	3.09×10^{-10}	7.42×10^{-11}
		DA	4.44×10^{-15}	10.8	6.46	2.01
		SSO	3.12×10^{-1}	3.12×10^{-1}	3.12×10^{-1}	5.65×10^{-17}
		EOSSO	4.44×10^{-15}	4.44×10^{-15}	4.44×10^{-15}	0
f_{11}	30	ABC	2.31×10^{-7}	1.34×10^{-2}	6.06×10^{-4}	2.47×10^{-3}
		BA	2.98×10^{2}	5.34×10^{2}	4.31×10^{2}	60
		GGSA	1.63	7.87	3.59	1.3
		DA	1.56	17.5	5.05	3.4
		SSO	1.26×10^{-2}	1.26×10^{-2}	1.26×10^{-2}	0
		EOSSO	0	0	0	0
f_{12}	30	ABC	1.02×10^{-10}	1.67×10^{-8}	3.32×10^{-9}	3.87×10^{-9}
		BA	22.1	49.7	38.3	7.94
		GGSA	4.78×10^{-22}	2.07×10^{-1}	3.83×10^{-2}	6.14×10^{-2}
		DA	1.71	52.6	12.2	12.9
		SSO	1.37×10^{-3}	1.37×10^{-3}	1.37×10^{-3}	2.21×10^{-19}
		EOSSO	9.29×10^{-3}	9.29×10^{-3}	9.29×10^{-3}	3.53×10^{-18}
f_{13}	30	ABC	1.27×10^{-8}	3.55×10^{-6}	2.93×10^{-7}	6.53×10^{-7}
		BA	80.8	1.19×10^{2}	1.04×10^{2}	10.1
		GGSA	5.41×10^{-21}	1.09×10^{-2}	3.66×10^{-3}	2.00×10^{-3}
		DA	4.95	3.09×10^{4}	1.52×10^{3}	5.69×10^{3}
		SSO	2.04×10^{-2}	2.04×10^{-2}	2.04×10^{-2}	3.53×10^{-18}
		EOSSO	6.17×10^{-1}	6.17×10^{-1}	6.17×10^{-1}	3.39×10^{-16}

Table 6. Simulation results for test fixed-dimension multimodal benchmark function.

Function	Dim	Algorithm	Best	Worst	Mean	std.
f_{14}	2	ABC	9.98×10^{-1}	9.98×10^{-1}	9.98×10^{-1}	1.20×10^{-5}
		BA	9.98×10^{-1}	22.9	10	6.97
		GGSA	9.98×10^{-1}	10.7	3	2.36
		DA	9.98×10^{-1}	9.98×10^{-1}	9.98×10^{-1}	8.09×10^{-11}
		SSO	1.99	1.99	1.99	1.36×10^{-15}
		EOSSO	9.98×10^{-1}	9.98×10^{-1}	9.98×10^{-1}	4.52×10^{-16}
f_{15}	4	ABC	4.16×10^{-4}	1.48×10^{-3}	9.93×10^{-4}	2.52×10^{-4}
		BA	3.07×10^{-4}	5.19×10^{-3}	1.10×10^{-3}	1.17×10^{-3}
		GGSA	4.80×10^{-4}	3.43×10^{-3}	1.78×10^{-3}	6.36×10^{-4}
		DA	4.91×10^{-4}	3.44×10^{-3}	1.28×10^{-3}	6.43×10^{-3}
		SSO	4.18×10^{-4}	4.18×10^{-4}	4.18×10^{-4}	1.65×10^{-19}
		EOSSO	3.70×10^{-4}	3.70×10^{-4}	3.70×10^{-4}	2.21×10^{-19}
f_{16}	2	ABC	−1.03	−1.03	−1.03	1.25×10^{-6}
		BA	−1.03	2.1	-8.18×10^{-1}	6.19×10^{-1}
		GGSA	−1.03	−1.03	−1.03	6.18×10^{-16}
		DA	−1.03	−1.03	−1.03	6.27×10^{-8}
		SSO	−1.03	−1.03	−1.03	4.52×10^{-16}
		EOSSO	−1.03	−1.03	−1.03	6.78×10^{-16}
f_{17}	2	ABC	3.97×10^{-1}	3.98×10^{-1}	3.97×10^{-1}	3.64×10^{-5}
		BA	3.97×10^{-1}	3.97×10^{-1}	3.97×10^{-1}	8.40×10^{-9}
		GGSA	3.97×10^{-1}	3.97×10^{-1}	3.97×10^{-1}	0
		DA	3.97×10^{-1}	3.97×10^{-1}	3.97×10^{-1}	7.62×10^{-7}
		SSO	3.97×10^{-1}	3.97×10^{-1}	3.97×10^{-1}	0
		EOSSO	3.97×10^{-1}	3.97×10^{-1}	3.97×10^{-1}	0
f_{18}	2	ABC	3	3.03	3	8.64×10^{-3}
		BA	3	84.	12	21.6
		GGSA	3	3	3	1.98×10^{-15}
		DA	3	3	3	3.28×10^{-7}
		SSO	3	3	3	4.52×10^{-16}
		EOSSO	3	3	3	1.81×10^{-15}
f_{19}	3	ABC	−3.86	−3.86	−3.86	1.77×10^{-6}
		BA	−3.86	−3.08	−3.83	1.41×10^{-1}
		GGSA	−3.86	−3.86	−3.86	2.52×10^{-15}
		DA	−3.86	−3.85	−3.86	1.05×10^{-3}
		SSO	−3.86	−3.86	−3.86	1.36×10^{-15}
		EOSSO	−3.86	−3.86	−3.86	4.48×10^{-14}
f_{20}	6	ABC	−3.32	−3.31	−3.32	1.03×10^{-3}
		BA	−3.32	−3.2	−3.25	6.02×10^{-2}
		GGSA	−3.32	−2.81	−3.29	9.71×10^{-2}
		DA	−3.32	−3.07	−3.26	7.34×10^{-2}
		SSO	−3.2	−3.2	−3.2	1.36×10^{-15}
		EOSSO	−3.32	−3.2	−3.2	2.17×10^{-2}
f_{21}	4	ABC	−10.1	−9.64	−9.89	1.58×10^{-1}
		BA	−10.1	−2.63	−4.96	2.85
		GGSA	−10.1	−5.05	−5.22	9.30×10^{-1}
		DA	−10.1	−5.09	−9.64	1.54
		SSO	−10.1	−10.1	−10.1	1.81×10^{-15}
		EOSSO	−10.1	−10.1	−10.1	9.70×10^{-11}
f_{22}	4	ABC	−10.3	−9.78	−10.1	1.90×10^{-1}
		BA	−10.4	2.75	5.33	3.19
		GGSA	−10.4	−5.08	−7.56	2.69
		DA	−10.4	−5.08	−9.12	2.25
		SSO	−10.3	−10.3	−10.3	5.42×10^{-15}
		EOSSO	−10.4	−10.4	−10.4	2.02×10^{-11}
f_{23}	4	ABC	−10.4	−9.67	-1.01×10^{-1}	2.06×10^{-1}
		BA	−10.5	−2.42	−5.36	3.52
		GGSA	−10.5	−10.5	−10.5	1.32×10^{-15}
		DA	−10.5	−5.17	−9.98	1.63
		SSO	−10.5	−10.5	−10.5	3.61×10^{-15}
		EOSSO	−10.5	−10.5	−10.5	2.45×10^{-11}

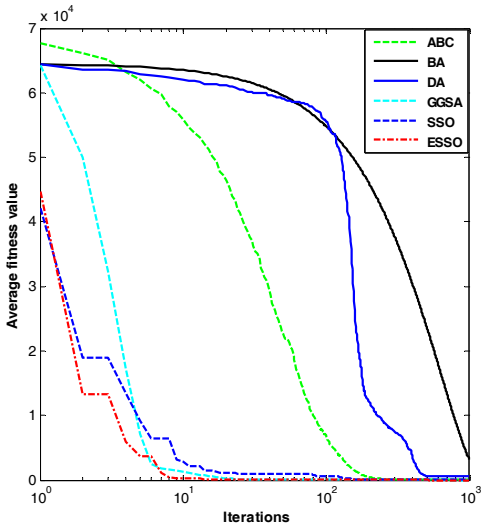

Figure 1. Dim = 30, evolution curves of fitness value for f_1.

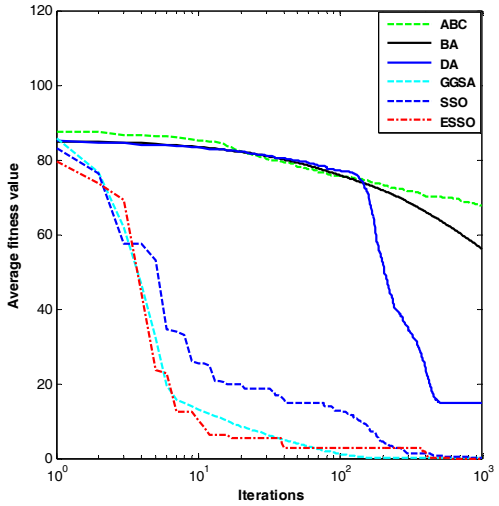

Figure 2. Dim = 30, evolution curves of fitness value for f_4.

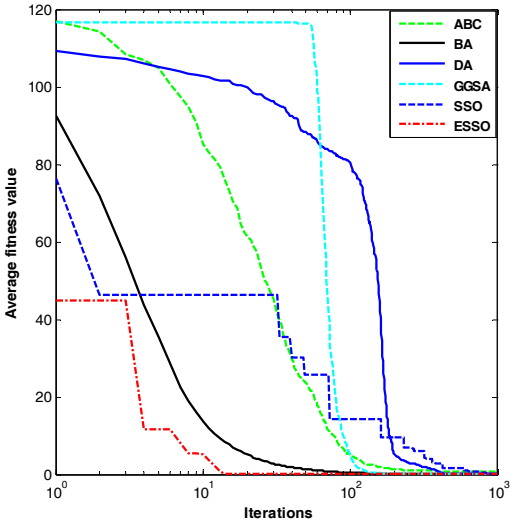

Figure 3. Dim = 30, evolution curves of fitness value for f_7.

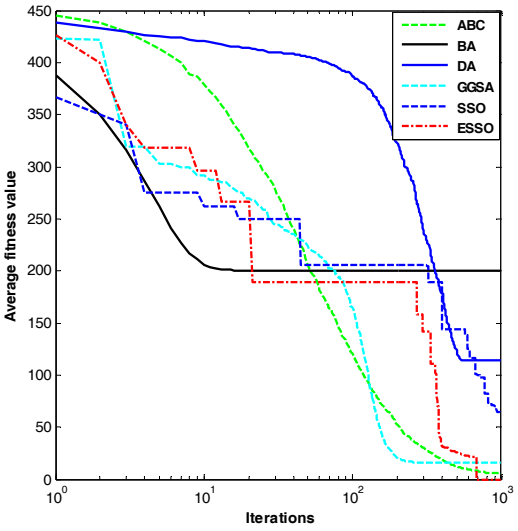

Figure 4. Dim = 30, evolution curves of fitness value for f_9.

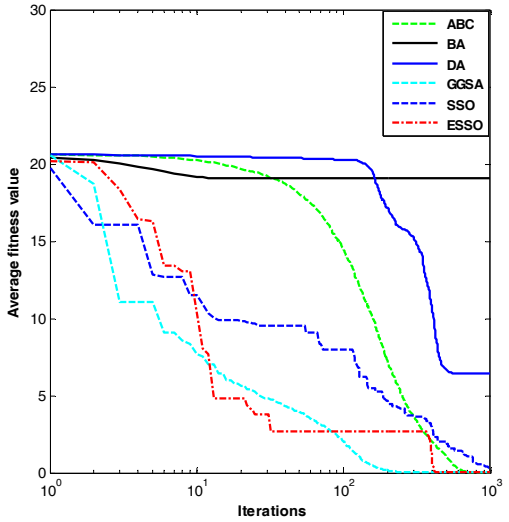

Figure 5. Dim = 30, evolution curves of fitness value for f_{10}.

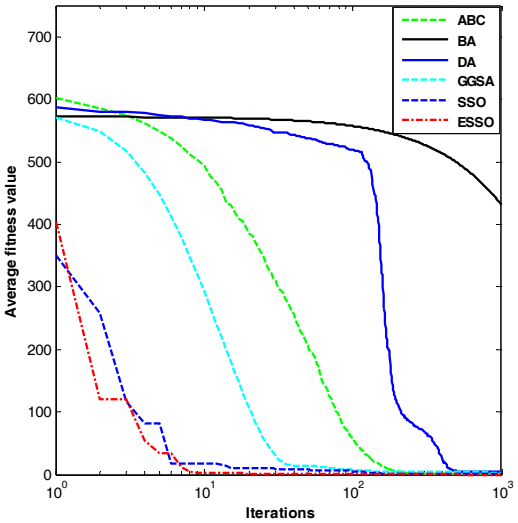

Figure 6. Dim = 30, evolution curves of fitness value for f_{11}.

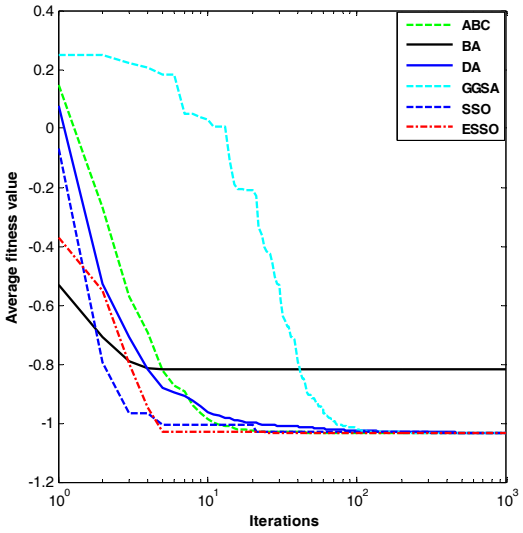

Figure 7. Dim = 2, evolution curves of fitness value for f_{16}.

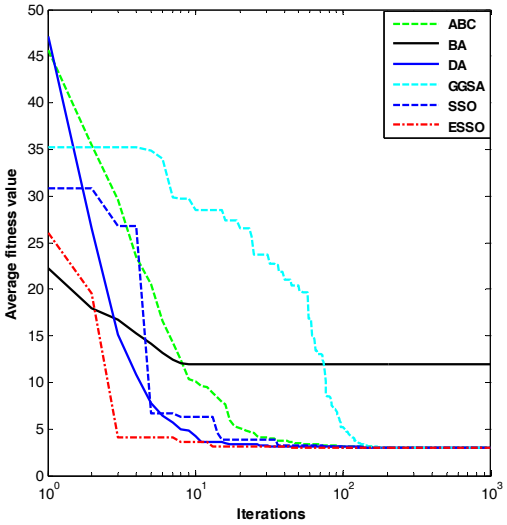

Figure 8. Dim = 2, evolution curves of fitness value for f_{18}.

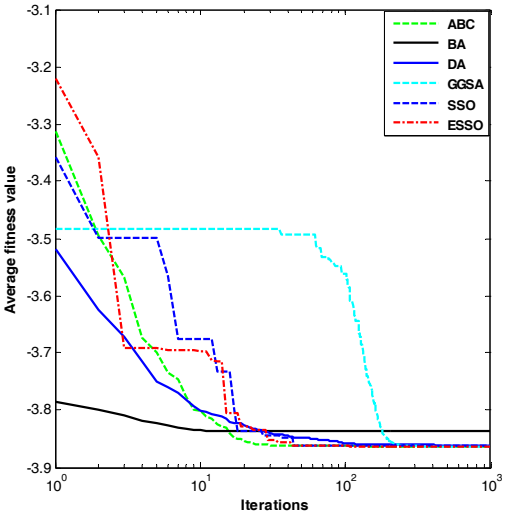

Figure 9. Dim = 3, evolution curves of fitness value for f_{19}.

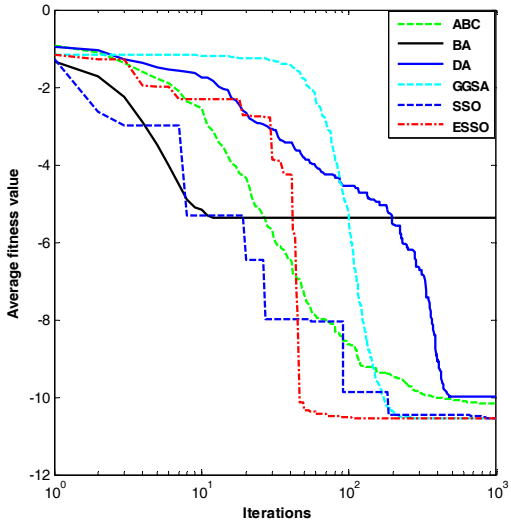

Figure 10. Dim = 4, evolution curves of fitness value for f_{23}.

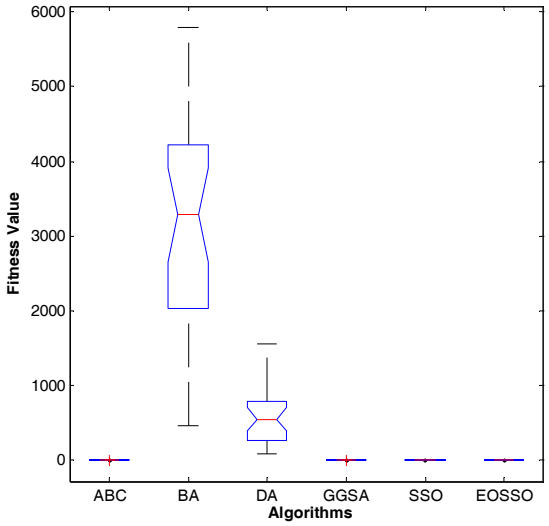

Figure 11. Dim = 30, variance diagram of fitness value for f_1.

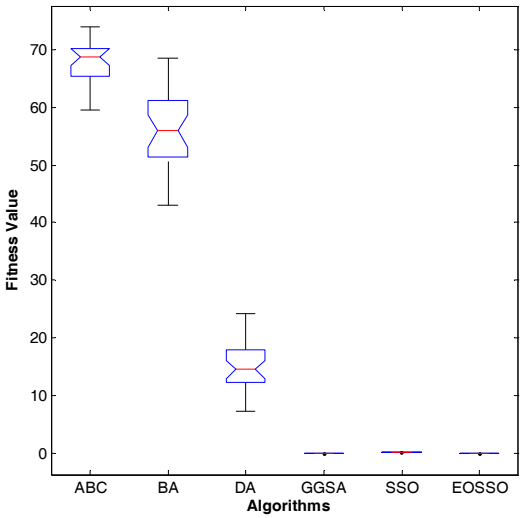

Figure 12. Dim = 30, variance diagram of fitness value for f_4.

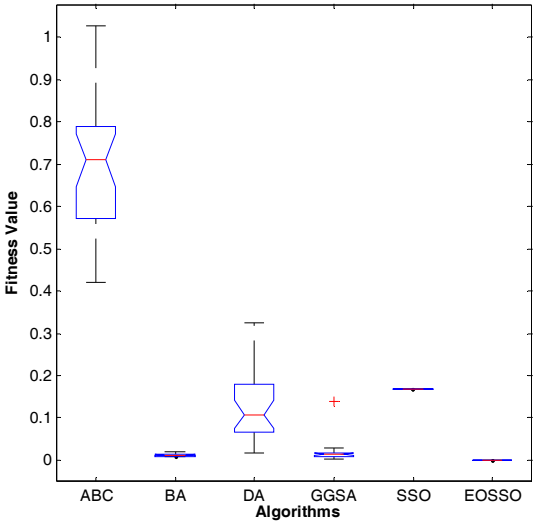

Figure 13. Dim = 30, variance diagram of fitness value for f_7.

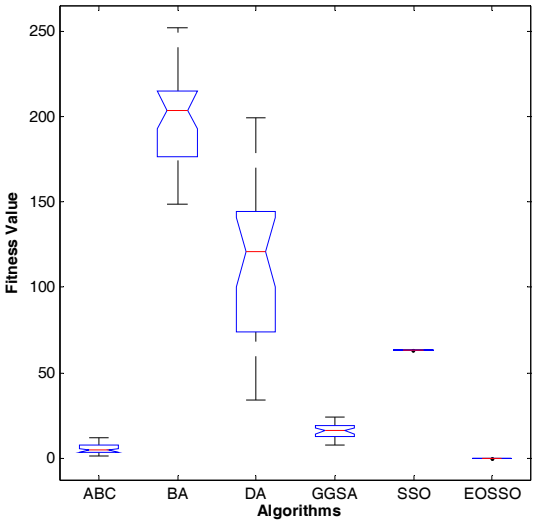

Figure 14. Dim = 30, variance diagram of fitness value for f_9.

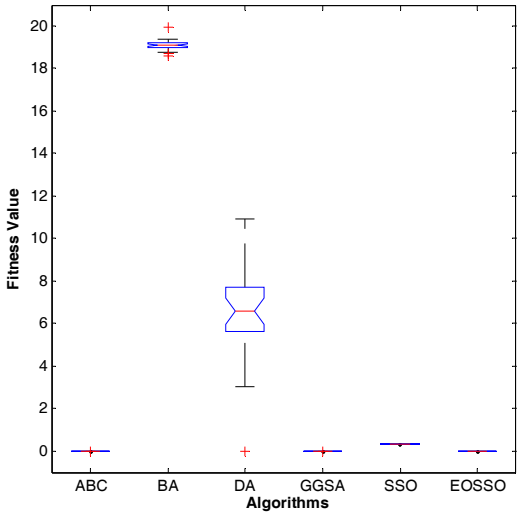

Figure 15. Dim = 30, variance diagram of fitness value for f_{10}.

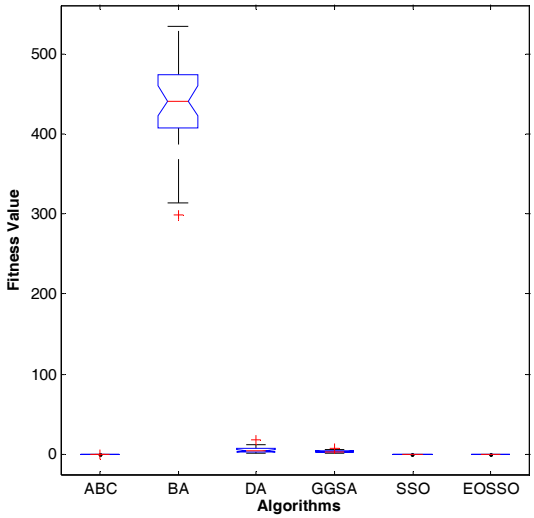

Figure 16. Dim = 30, variance diagram of fitness value for f_{11}.

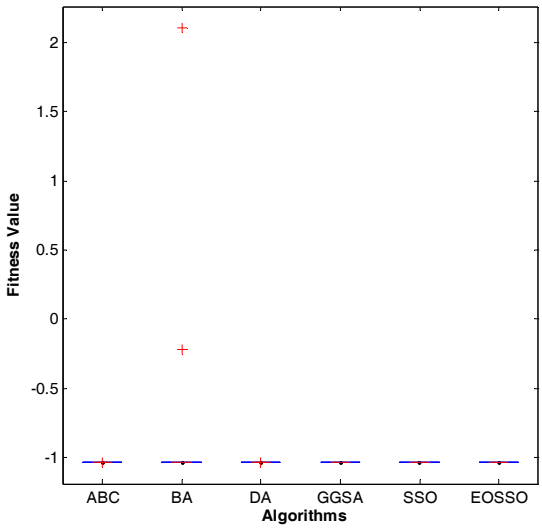

Figure 17. Dim = 2, variance diagram of fitness value for f_{16}.

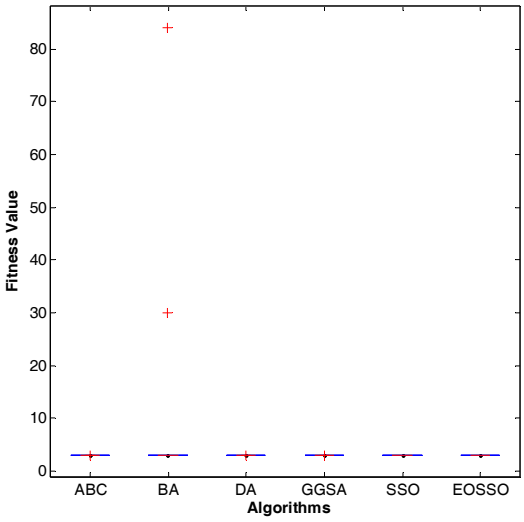

Figure 18. Dim = 2, variance diagram of fitness value for f_{18}.

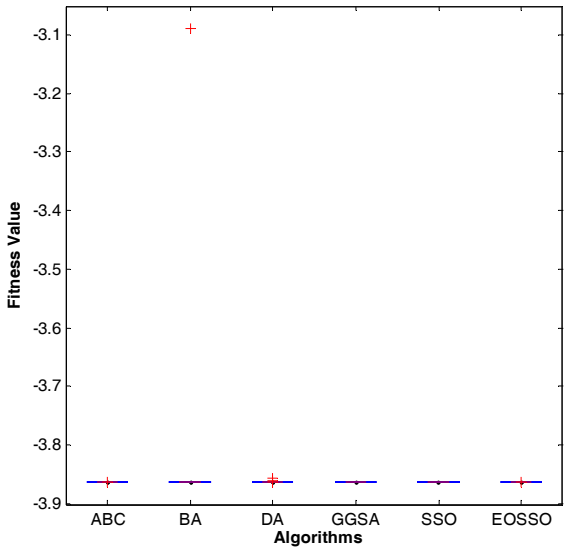

Figure 19. Dim = 3, variance diagram of fitness value for f_{19}.

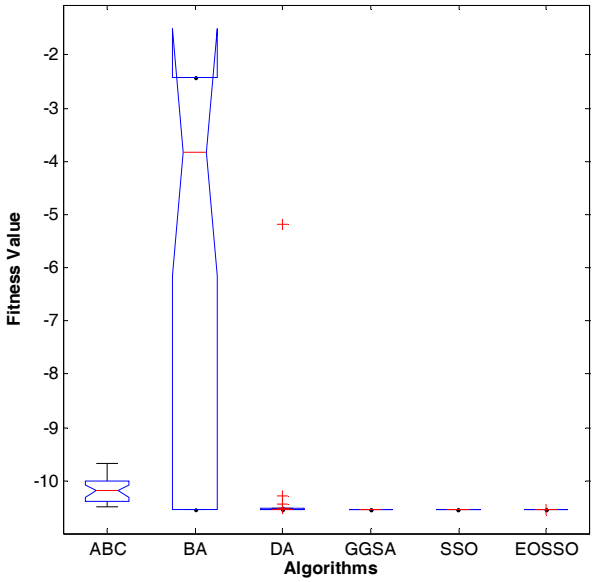

Figure 20. Dim = 4, variance diagram of fitness value for f_{23}.

5. Result Analysis

Seen from Table 4, in unimodal benchmark functions, EOSSO can get a better optimal solution for f_1, f_2, f_3, f_4, f_5 and f_7 and has a very strong robustness. For f_6, the precision of optimal fitness value and mean fitness value of GGSA are higher than other algorithms. For the seven functions in unimodal benchmark functions, standard deviation of EOSSO is less than that of other algorithm. In addition, this means that in the optimization of unimodal function, EOSSO has better stability.

Similarly, seen from Table 5, EOSSO can find the optimal solution for f_9 and f_{11}. In addition, the standard deviations are zeros. For f_9, f_{10} and f_{11}, the precision of mean fitness value, best fitness value, worst fitness value and standard deviation of EOSSO are better than other algorithms. These results mean that EOSSO has a strong searching ability and a great stability for solving multimodal function optimization.

For f_{14}–f_{23}, we can see from Table 6 above that the best fitness value, the worst fitness value, mean fitness value and standard deviation produced by EOSSO are better than other algorithms.

In addition, for, EOSSO can get better best fitness value and mean fitness value and worst fitness value, but standard deviation are worse than that of SSO. After analyzing Tables 4–6, a conclusion can be easily drawn that EOSSO has a great ability for solving function optimization problems according to the experimental results.

Figures 1–10 show the evolution curves of fitness value for $f_1, f_4, f_7, f_9, f_{10}, f_{11}, f_{16}, f_{18}, f_{19}$ and f_{23}. From these Figures, we can easily find that EOSSO converge faster than other population based algorithms mentioned above, and the values obtained by EOSSO are closer to the optimal value of benchmark functions. These show that EOSSO has a faster convergence speed and a better precision than SSO and other population based algorithms. Figures 11–20 show the anova test of global minimum for $f_1, f_4, f_7, f_9, f_{10}, f_{11}, f_{16}, f_{18}, f_{19}$ and f_{23}. From these figures, we can discover that the standard deviation of EOSSO is much smaller, and the number of outlier is less than other algorithms. These imply that EOSSO has a great performance with a high degree of stability. In sum, proposed EOSSO is an algorithm with fast convergence speed, high level of precision and a great performance of stability.

In Section 4.2, 23 standard benchmark functions are selected to evaluate the performance of elite opposition-based Social Spider Optimization algorithm (EOSSO). f_1–f_7 are unimodal function, f_8–f_{13} are multimodal function, f_{14}–f_{23} are fixed-dimension multimodal benchmark function. Experiment results are listed in Tables 4–6. Figures 1–20 are evaluation curves of fitness values and anova test of global minimum for $f_1, f_4, f_7, f_9, f_{10}, f_{11}, f_{16}, f_{18}, f_{19}$ and f_{23}. The results in tables show that a more precise solution can be found by EOSSO. Figures listed in paper reflect a fact that EOSSO has a faster convergence speed and a higher stability.

6. Conclusions and Future Works

In order to overcome the disadvantage of Social Spider Optimization algorithm that it is still easy to fall into a local optimal solution, this paper presents a novel SSO algorithm called EOSSO by using OBL and the elite selection mechanism. Opposition-based Learning (OBL) is a new concept in computational intelligence and it has been proven to be an effective strategy to improve performance of various optimization algorithms. The main idea behind OBL is to transform solutions in the current search space to a new search space. By simultaneously considering the solutions in the current search space and the transformed search space, OBL can provide a higher chance of finding solutions that are closer to the global optimum. This kind of mechanism enhances the diversity of the population, which helps to improve its exploration ability. From the results of the 23 benchmark functions, the performance of EOSSO is better than, or at least comparable with other population-based algorithm mentioned in this paper. EOSSO has a fast convergence speed, a relatively high degree of stability and it is also much more accurate in precision.

For EOSSO, there are various issues that still deserve further study. Firstly, multi-objective optimization problems can be seen here and there in the real world. Compared with single objective optimization problems, it is often very challenging to obtain high-equality solution. The proposed elite opposition-based Social Spider Optimization algorithm should be used to solve these multi-objective optimization problems in the future to validate its performance. Secondly, there exist many NP-hard problems in literature, such as the traveling salesman problem, graph coloring problem, radial basis probabilistic neural networks, and finder of polynomials based on root moments. In order to test the performance of EOSSO, it should be used to solve these NP-hard problems in the future. Thirdly, the proposed EOSSO should solve some practical engineering problems in the future, for example, welding beam and spring pressure design problem, vehicle scheduling optimization problem and scheduling optimization of hydropower station, etc. Finally, although the proposed algorithm is tested with 23 benchmark functions, a more comprehensive computational study should be made to test the efficiency of the proposed solution technique in the future.

Acknowledgments: This work is supported by National Science Foundation of China under Grants No. 61463007; 61563008, and Project of Guangxi Natural Science Foundation under Grant No. 2016GXNSFAA380264.

Author Contributions: Ruixin Zhao designed algorithm; Qifang Luo performed the experiments and results analysis; Yongquan Zhou wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Socha, K.; Dorigo, M. Ant colony optimization for continuous domains. *Eur. J. Oper. Res.* **2008**, *185*, 1155–1173. [CrossRef]
2. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
3. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 1995; Volume IV, pp. 1942–1948.
4. Yang, X.S. Multi-objective firefly algorithm for continuous optimization. *Eng. Comput.* **2013**, *29*, 175–184. [CrossRef]
5. Liu, J.; Zhou, Y.; Zhao, G. Leader glowworm swarm optimization algorithm for solving nonlinear equations systems. *Electr. Rev.* **2012**, *88*, 101–106.
6. Mucherino, A.; Seref, O. Monkey search: A novel metaheuristic search for global optimization. In Proceedings of the American Institute of Physics Conference, Gainesville, FL, USA, 28–30 March 2007; pp. 162–173.
7. Alatas, B. Chaotic harmony search algorithms. *Appl. Math. Comput.* **2010**, *216*, 2687–2699. [CrossRef]
8. Yang, X.S.; Deb, S. Cuckoo search via Levy flights. In Proceedings of the World Congress on Nature and Biologically Inspired Computing (NaBIC2009), Coimbatote, India, 9–11 December 2009; pp. 210–214.
9. Wang, G.-G.; Gandomi, A.H.; Zhao, X.; Chu, H.E. Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. *Soft Comput.* **2016**, *20*, 273–285. [CrossRef]
10. Yang, X.S. A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization*; Gonzalez, J.R., Pelta, D.A., Cruz, C., Eds.; Springer: Berlin, Germany, 2010; pp. 65–74.
11. Wang, G.-G.; Guo, L.; Gandomi, A.H.; Hao, G.-S.; Wang, H. Chaotic krill herd algorithm. *Inf. Sci.* **2014**, *274*, 17–34. [CrossRef]
12. Wang, G.-G.; Gandomi, A.H.; Alavi, A.H. Stud krill herd algorithm. *Neurocomputing* **2014**, *128*, 363–370. [CrossRef]
13. Wang, G.; Guo, L.; Wang, H.; Duan, H.; Liu, L.; Li, J. Incorporating mutation scheme into krill herd algorithm for global numerical optimization. *Neural Comput. Appl.* **2014**, *24*, 853–871. [CrossRef]
14. Cuevas, E.; Cienfuegos, M.; Zaldivar, D.; Perez-Cisneros, M. A swarm optimization algorithm inspired in the behavior of the social-spider. *Expert Syst. Appl.* **2013**, *40*, 6374–6384. [CrossRef]
15. Wang, H.; Rahnamay, S.; Wu, Z. Parallel differential evolution with self-adapting control parameters and generalized opposition-based learning for solving high-dimensional optimization problems. *J. Parallel Distrib. Comput.* **2013**, *73*, 62–73. [CrossRef]
16. Zhou, Y.; Wang, R.; Luo, Q. Elite opposition-based flower pollination algorithm. *Neurocomputing* **2016**, *188*, 294–310. [CrossRef]
17. Tizhoosh, H.R. Opposition-based learning: A new scheme for machine intelligence. In Proceedings of the International Conference on Computation Intelligence on Modeling Control Automation and International Conference on Intelligent Agents, Web Technologies Internet Commerce, Vienna, Austria, 28–30 November 2005; pp. 695–701.
18. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M.A. Opposition versus randomness in soft computing techniques. *Appl. Soft Comput.* **2008**, *8*, 906–918. [CrossRef]
19. Wang, H.; Li, H.; Liu, Y.; Li, C.; Zeng, S. Opposition based particle swarm algorithm with Cauchy mutation. In Proceedings of the IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007; pp. 4750–4756.
20. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M.A. Opposition-based differential evolution for optimization of noisy problems. In Proceedings of the IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, 16–21 July 2006; pp. 1865–1872.
21. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M.A. Opposition-based differential evolution algorithms. In Proceedings of the IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, 16–21 July 2006; pp. 2010–2017.

22. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M.A. Opposition-based differential evolution. *IEEE Trans. Evol. Comput.* **2008**, *2*, 64–79. [CrossRef]
23. Tizhoosh, H.R. Opposition-based reinforcement learning. *J. Adv. Comput. Intell. Inf.* **2006**, *10*, 578–585.
24. Khalvati, F.; Tizhoosh, H.R.; Aagaard, M.D. Opposition-based window memorization for morphological algorithms. In Proceedings of the IEEE Symposium on Computational Intelligence in Image and Signal Processing, Honolulu, HI, USA, 1–5 April 2007; pp. 425–430.
25. Al-Qunaieer, F.S.; Tizhoosh, H.R.; Rahnamayan, S. Oppositional fuzzy image thresholding. In Proceedings of the IEEE International Conference on Fuzzy Systems, Barcelona, Spain, 18–23 July 2010; pp. 1–7.
26. Tizhoosh, H.R. Opposite fuzzy sets with applications in image processing. In Proceedings of the Joint 2009 International Fuzzy Systems Association World Congress and 2009 European Society of Fuzzy Logic and Technology Conference, Lisbon, Portugal, 20–24 July 2009; pp. 36–41.
27. Malisia, A.R.; Tizhoosh, H.R. Applying opposition-based ideas to the ant colony system. In Proceedings of the IEEE Swarm Intelligence Symposium, Honolulu, HI, USA, 1–5 April 2007; pp. 182–191.
28. Haiping, M.; Xieyong, R.; Baogen, J. Oppositional ant colony optimization algorithm and its application to fault monitoring. In Proceedings of the 29th Chinese Control Conference (CCC), Beijing, China, 29–31 July 2010; pp. 3895–3903.
29. Lin, Z.Y.; Wang, L.L. A new opposition-based compact genetic algorithm with fluctuation. *J. Comput. Inf. Syst.* **2010**, *6*, 897–904.
30. Ventresca, M.; Tizhoosh, H.R. Improving the convergence of back propagation by opposite transfer functions. In Proceedings of the International Joint Conference on Neural Networks (IJCNN'06), Vancouver, BC, Canada, 16–21 July 2006; pp. 4777–4784.
31. Ventresca, M.; Tizhoosh, H.R. Opposite transfer functions and back propagation through time. In Proceedings of the IEEE Symposium on Foundations of Computational Intelligence, Honolulu, HI, USA, 1–5 April 2007; pp. 570–577.
32. Han, L.; He, X.S. A novel opposition-based particle swarm optimization for noisy problems. In Proceedings of the Third International Conference on Natural Computation (ICNC 2007), Haikou, China, 24–27 August 2007; pp. 624–629.
33. Shaw, B.; Mukherjee, V.; Ghoshal, S.P. A novel opposition-based gravitational search algorithm for combined economic and emission dispatch problems of power systems. *Int. J. Electr. Power Energy Syst.* **2012**, *35*, 21–33. [CrossRef]
34. Chatterjee, A.; Ghoshal, S.P.; Mukherjee, V. Solution of combined economic and emission dispatch problems of power systems by an opposition based harmony search algorithm. *Int. J. Electr. Power Energy Syst.* **2012**, *39*, 9–20. [CrossRef]
35. Bhattacharya, A.; Chattopadhyay, P.K. Solution of economic power dispatch problems using oppositional biogeography-based optimization. *Electr. Power Compon. Syst.* **2010**, *38*, 1139–1160. [CrossRef]
36. Ergezer, M.; Simon, D.; Du, D.W. Oppositional biogeography-based optimization. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC 2009), San Antonio, TX, USA, 11–14 October 2009; pp. 1009–1014.
37. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]