J Inf Process Syst, Vol.16, No.1, pp.210~223, February 2020
https://doi.org/10.3745/JIPS.01.0049

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Symbiotic Organisms Search for Constrained Optimization Problems

Yanjiao Wang*, Huanhuan Tao*, and Zhuang Ma*

Abstract
Since constrained optimization algorithms are easy to fall into local optimum and their ability of searching are weak, an improved symbiotic organisms search algorithm with mixed strategy based on adaptive ε constrained (ε_SOSMS) is proposed in this paper. Firstly, an adaptive ε constrained method is presented to balance the relationship between the constrained violation degrees and fitness. Secondly, the evolutionary strategies of symbiotic organisms search algorithm are improved as follows. Selecting different best individuals according to the proportion of feasible individuals and infeasible individuals to make evolutionary strategy more suitable for solving constrained optimization problems, and the individual comparison criteria is replaced with population selection strategy, which can better enhance the diversity of population. Finally, numerical experiments on 13 benchmark functions show that not only is ε_SOSMS able to converge to the global optimal solution, but also it has better robustness.

Keywords
Constrained Optimization Problems, ε Constrained, Symbiotic Organisms Search

1. Introduction

The constrained optimization problem is a type of mathematics problems that exists widely in engineering applications, such as image processing, network communication, resource allocation optimization, and so on. Traditional deterministic optimization algorithms usually use gradient-based search methods to solve constrained optimization problems. The main disadvantage of these methods is that they need to set a good initial value. It is only applicable to the case where the objective function and the constrained conditions are different, and the solution is obtained mostly a local optimal solution [1,2]. In recent years, constrained optimization methods based on the swarm intelligence algorithms have become the main method to solve the constrained optimization problems because of the following two advantages. The first reason is the analytical form of the objective function should be independent of methods. The second is that it can converge to the global optimum with a large probability and has strong robustness [3,4].

However, almost all swarm intelligence algorithms are designed to solve unconstrained optimization problems. They lack a clear and effective mechanism to constrain the population search within the

Manuscript received November 29, 2017; first revision July 24, 2018; accepted September 14, 2018.
Corresponding Author: Yanjiao Wang (wangyanjiao1028@126.com)
* School of Electrical Engineering, Northeast Electric Power University, Jilin, China ({wangyanjiao1028, taociguan}@126.com, 867382014@qq.com)

feasible domain, so that the constrained optimization problems cannot be directly solved, and must be supplemented with certain constrained processing techniques. Therefore, many experts and scholars are working on the integration of constrained processing techniques, evolutionary strategies, and the combination of constrained processing techniques and evolutionary strategies to improve the ability of solving constrained optimization problems based on swarm intelligence algorithms [5-7].

Constrained processing technologies as an important part of the constrained optimization algorithms develop rapidly, such as the special operator method to the Deb's rule and *ε* constrained. Researches that combine the above constrained processing technologies and swarm intelligence algorithms have shown significant results. However, the accuracy and robustness of the constrained optimization algorithms still need to be further improved. A modified artificial bee colony algorithm (MABC) is proposed in [8]. In this paper, artificial bee colony (ABC) uses Deb's rules consisting of three simple heuristic rules and a probabilistic selection scheme for feasible solutions based on their fitness values and infeasible solutions based on their constrained violation degrees. The results show that it has poor results for some functions. The authors of [9] propose *ε*DE that combines the Differential Evolution algorithm with *ε* constrained, but *ε*DE has more function evaluation times. The authors of [10] propose an optimization algorithm based on adaptive *ε* constrained. They introduce a new individual comparison criterion and *ε* setting method to improve the search efficiency and robustness of the algorithm to a certain extent. However, it has a poor effect on solving complex high-dimensional constrained optimization problems with more equality constraints. Interior penalty rules based on evolutionary algorithm (IPES) is proposed in [11], in which interior penalty functions are used to evaluate feasible solutions and constrained violation degrees. Experiments show that the convergence accuracy of IPES is poor. An improved artificial bee colony (I-ABC) algorithm for constrained optimization problems is proposed in [12]. The better infeasible solutions and the periodic boundary processing method are used to enhance the algorithm's evolution ability. However, the experiments on the benchmark functions show that the optimization ability of I-ABC needs to be improved. In summary, the constrained conditions of the constrained optimization problems and the topological structure of the feasible domain and infeasible domain are very complex. Therefore, the constrained optimization algorithms are required to improve the ability of exploration and maintaining population diversity, and the processing ability of constrained processing technologies for infeasible solutions also needs to be improved.

In this paper, in order to solve constrained optimization problems better, an improved symbiotic organisms search algorithm with mixed strategy based on adaptive *ε* constrained (*ε*_SOSMS) is proposed. Adjusting the parameter ε setting method according to the real-time information of the evolutionary process to enhance the use of infeasible solutions and supply population diversity. The individual updating strategy and evolutionary strategies of symbiotic organisms search (SOS) are improved to improve the search ability, and the selection of optimal individuals is adjusted according to the proportion of feasible individuals and infeasible individuals. Finally, numerical experiments on 13 benchmark functions show that not only is *ε*_SOSMS able to converge to the global optimal solution, but also it has better robustness.

The remaining parts of this paper are arranged as follows: Section 2 describes the constrained optimization problems. Section 3 introduces SOS algorithm. Section 4 introduces an improved SOS algorithm for the constrained optimization problems. Section 5 is the experiments and discussion. Section 6 is the summary of the full text.

2. Constrained Optimization Problems

2.1 Problem Description

Without losing the generality, the minimized constrained optimization problem can be simply expressed as Eq. (1).

$$\min(f(x)),\ \ x=(x_1,x_2,...,x_n)$$
$$s.t.\begin{cases} g_i(x)\le 0,\ i=1,2,...m \\ h_j(x)=0,\ j=1,2,...,n \end{cases} \tag{1}$$

where x is an n-dimensional variable whose value is greater than u_i and less than l_i. $f(x)$ is the objective function. g_i is the inequality constraints, and m is the number of inequality constraints. h_j is the equality constraints, and n is the number of equality constraints. The optimal solution of this problem is the minimum value of the objective function under the premise of satisfying all constraints.

2.2 ε Constrained

ε constrained is one of the best constrained processing methods. It uses parameter ε to distinguish individuals. To extend the search of population by allowing more infeasible individuals with less constrained violation degrees to participate in the evolution, therefor the search ability of the algorithm is enhanced.

Constrained violation degree which represents the degree of an solution violates the constrained conditions is calculated as Eq. (2):

$$G(X)=\sum_{i=1}^{m}\max(0,g_i(X))+\sum_{\mathrm{j}=1}^{n}\max(0,|h_j(X)|-\delta) \tag{2}$$

In solving the constrained optimization problems, equality constraints are usually transformed into inequality constraints by Eq. (3):

$$|h_j|-\delta\le 0 \tag{3}$$

where δ generally takes a smaller number, and δ is constantly set to 0.0001 as literature [13].

ε constrained shows better results than the other constrained processing methods in dealing with relationship between feasible individuals and infeasible individuals. Its individual selection strategy is as follows. When the constrained violation degrees of two individuals are all greater than ε, the individual with smaller constrained violation degree is selected. When the constrained violation degrees of two individuals are all smaller than ε, the individual with smaller fitness value is selected. When the constrained violation degree of one individual is smaller than ε, and the other is greater than ε, the individual whose constrained violation degree is smaller than ε is selected. The method of setting parameter ε descripted in [14] is shown in Eqs. (4), (5), and (6).

$$\varepsilon(\mathrm{t})=\begin{cases}\varepsilon_0\times(1-\dfrac{t}{t_c})^{cp} & ,1\le t<t_a \\ 0,t_a\le t\le T\end{cases} \tag{4}$$

$$\varepsilon_0 = \phi(X_\theta) \tag{5}$$

$$\theta = 0.2 \times S \tag{6}$$

where X_θ is the θ-th individual in ascending order of constrained violation degrees of all individuals. S represents the population size. T is the maximum iteration number. t is a fixed number which is generally set by experiments. Φ is constrained violation degree.

Eq. (4) shows that parameter ε will decrease with the increase of the number of iterations until it reaches zero. In the pre-evolutionary phase, infeasible individuals with large violation degrees are allowed to participate in evolution. As the iteration progresses, the number of feasible individuals in the evolution population increases, and ε keeps getting smaller. Constrained violation degree that allows the infeasible solution to participate in evolution gradually decreases until all individuals in the evolution population are feasible solutions.

3. Symbiotic Organisms Search

As a new swarm intelligence optimization algorithm, SOS algorithm was proposed in 2014 [15]. Three kinds of update mechanisms are established in SOS by simulating the interaction between different organisms in nature, including mutualism phase, commensalism phase, and parasitism phase. To perform three update mechanisms in a loop until the termination condition is reached, then the optimal solution is output. Update mechanisms are shown as follows.

3.1 Mutualism Phase

In this phase, a biological individual X_i interacts with another individual X_j, which is selected randomly from the population $(j \neq i)$, both of them promote their development under the guidance of the current optimal individual, then produce new individuals X_{inew} and X_{jnew}, the update methods are shown as Eqs. (7) and (8).

$$Mutual - Vector = \frac{X_i + X_j}{2} \tag{7}$$

$$\begin{cases} X_{inew} = X_i + rand(0,1) \times (X_{best} - Mutual\_Vector \times BF_1) \\ X_{jnew} = X_j + rand(0,1) \times (X_{best} - Mutual\_Vector \times BF_2) \end{cases} \tag{8}$$

where *rand* (0,1) is a random number between [0,1], X_{best} represents the individual with smallest fitness value. BF_1 and BF_2 are determined randomly as either 1 or 2. X_{inew} is the updated result of X_i according to formula (8). Preserve the one with better fitness value and so do X_{jnew} and X_j.

3.2 Commensalism Phase

The individual X_i in the population carries out the Commensalism phase according to the Eq. (9)

$$X_{inew} = X_i + rand(-1,1) \times (X_{best} - X_j) \tag{9}$$

where *rand* (-1,1) is a random number between [-1,1]. The remaining variables have the same meaning as Mutualism phase. Comparing X_{inew} with X_i and retaining the one with better fitness value.

3.3 Parasitic Phase

Individuals update mechanism in the parasitic phase is shown as follows. *Parasite_Vector* is created in the search space by duplicating individual X_i, then modify the randomly selected dimensions using a random number. $X_j(j \neq i)$ that is randomly selected compares with *Parasite_Vector*. If *Parasite_Vector* has a better fitness value, it can be saved. Otherwise, keeping X_j unchanged.

The flowchart of SOS is shown in Fig. 1.

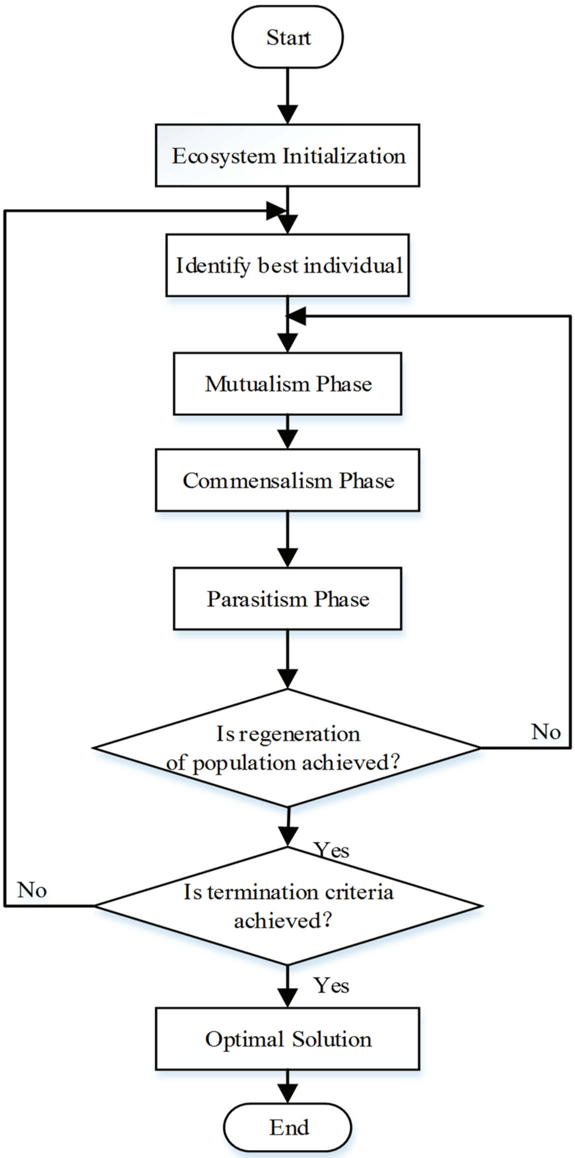


Fig. 1. Flowchart of SOS.

4. Symbiotic Organisms Search Algorithm for Constrained Optimization Problems

SOS is proposed for unconstrained optimization problem. In order to solve constrained optimization problems, an improved symbiotic organisms search algorithm with mixed strategy based on adaptive ε constrained (ε_SOSMS) is proposed in this paper.

4.1 ε Constrained

The framework of ε_SOSMS is shown as follows.

Step 1: Initialize parameters, including the size of population, searching range, and so on. And generate initial population randomly.

Step 2: Calculate the fitness and constrained violation degree of each individual.

Step 3: Select the optimal individual and update the individuals according to the mutualism strategy as Section 4.3

Step 4: Update the individuals according to the commensalism strategy as Section 4.3.

Step 5: Update the individuals according to the parasitic phase as Section 3.

Step 6: Select the individuals to form evolutionary population as Section 4.2.2.

Step 7: If the termination criterion is not satisfied, go to Step 3. Otherwise, stop and output the optimal solution.

The details of the method of selecting optimal individual of individuals update section of Step 3, Step 4 and Step 5 are shown in Section 4.3.2. The individual comparison method in the update strategy is carried out in the Section 4.2. The details of individuals' renewal in mutualism phase and commensalism phase are as follows in the Section 4.3.1.

4.2 Improved ε Constrained

ε constrained consists of setting parameter ε and individual selection strategy, which are improved as follows.

4.2.1 Adaptive ε constrained

Many studies have shown that the optimal solution of the constrained optimization problems often exist at the boundary of the feasible domain, and infeasible solutions which is regarded as poor individuals cannot be saved. Actually, the individuals with small constrained violation degrees are the link between feasible and infeasible regions. Allowing the individuals with small constrained violation degrees to participate in evolution can enhance boundary search capabilities and provide direction information for the evolution of the algorithm. However, too much infeasible individuals or the individuals with big constrained violation degrees can also affect evolution. Therefore, the reasonable disposal of feasible and infeasible solutions plays an important role in evolution. In previous studies, the method of setting parameter ε depends only on the number of evolution iterations. It does not make full use of the available information of feasible individuals and infeasible individuals with small constrained violation degrees in the iteration process, which affects the evolutionary efficiency of the algorithm to some extent. Based on this, this paper presents a new method of setting parameter ε as shown in Eq. (10).

$$\varepsilon(t)=\begin{cases}\dfrac{G_{max}-G^{'}}{G_{max}-G_{min}+\text{esp}}\times \mathrm{e}^{(1-\frac{t}{T})\times\beta},t<\dfrac{T}{n}\\ 0,\mathrm{t}\geq\dfrac{T}{n}\end{cases} \tag{10}$$

where G_{max} and G_{min} are the maximum and minimum value of constrained violation degrees of all individuals. $G^{'}$ is the average value of all individuals' constrained violations degrees. t is the number of current iteration, and T is the maximum number of iterations. β is the proportion of feasible individuals in the current iterations.

As shown in Eq. (10), the improved method of setting parameter ε is adaptively changed according to the constrained violation degrees and the number of iterations. Not only rely on the number of iterations, but also make full use of the effective information of infeasible solutions. Through adaptive adjustment, the balance between feasible individuals and infeasible individuals and the relationship between constrained violation degrees and fitness are enhanced, so the search ability of the algorithm is improved.

4.2.2 Individual selection strategy

In order to ensures that some better infeasible individuals are selected to make the algorithm move closer to the feasible domain faster, a new individual selection strategy of ε constrained is proposed in this paper according to three possible cases of the merged population combining new population and original population. The population includes all individuals are infeasible, feasible individuals and infeasible individuals exist simultaneously, and all individuals are feasible. The details are shown as follows.

(1) All individuals are infeasible. All individuals are ranked in ascending order according to the constrained violation degrees, and the top N individuals with the smaller constrained violation degrees constitute a new evolution population.

(2) Feasible individuals and infeasible individuals exist simultaneously. It is common knowledge that how to deal with the relationship between infeasible individuals and feasible individuals is the key to solve the constrained optimization problems. Based on the above viewpoint, both of the fitness value and constrained violation degree are taken into account to select the better individuals when the merged population include feasible individuals and infeasible individuals. The details are as follows.

Because the individual's fitness value and constraint violation degree lack of consistency measure, so they need to be normalized. The fitness value and constrained violation degree of individual X_i are normalized by Eq. (11) and Eq. (12) respectively, and sum up the normalized fitness value and the constrained violation degree as Eq. (13). Then sort the summed results in ascending order, and the top N individuals constitute a new evolution population.

$$f^{'}(X_i)=\frac{f(X_i)}{\sum f(X_i)} \tag{11}$$

$$G^{'}(X_i)=\frac{G(X_i)}{\sum G(X_i)} \tag{12}$$

$$F(X_i)=f^{'}(X_i)+G^{'}(X_i) \tag{13}$$

(3) All individuals are feasible. In this case, constrained optimization problem becomes unconstrained

optimization problem. Therefore, the N individuals with the smaller fitness value are selected to constitute a new evolution population.

4.3 Improved Mutualism Phase and Commensalism Phase

4.3.1 Improved the formulas of individual renewal

In SOS, the individuals need to learn from the optimal individuals in mutualism and commensalism phases as Eq. (8) and Eq. (9). However the convergence speed can be accelerated, it reduces the diversity of the population to a certain extent. Considering when there are too many infeasible individuals in population, it is necessary to make the individuals move to the feasible domain. In this paper, in order to enhance the population exploitation ability, the differential vector is added to Eq. (8) and Eq. (9) in mutualism phase and commensalism phase to form Eq. (14) and Eq. (15), respectively.

$$\begin{cases} new\_X_i = X_i + rand \times (X_{best} - BF_1 \times Mutual\_Vector) + rand \times (X\_c - X\_r) \\ new\_X_j = X_j + rand \times (X_{best} - BF_2 \times Mutual\_Vector) + rand \times (X\_c - X\_r) \end{cases} \tag{14}$$

$$new\_X_i = X_i + rand(-1,1) \times (X_{best} - X_j) + rand \times (X\_c - X\_r) \tag{15}$$

where X_{best} represents the optimal individual. X_c represents the individual with smallest constrained violation degree. X_j and X_r are random individuals $(r \neq i \neq j)$.

According to the above search strategy, the selected individuals which have the smaller constrained violation degree or fitness represent the excellent evolutionary information of the current population. Learning from them ensure that the population can always move closer to the optimal location, and make the infeasible individuals transform into feasible individuals gradually.

4.3.2 The selection of optimal individuals

As described in Section 4.3.1, Eq. (14) and Eq. (15) need to select the optimal individual X_{best}. Considering the individuals in current population may exist three cases, which all individuals are infeasible, feasible individuals and infeasible individuals exist simultaneously, and all individuals are feasible, X_{best} can be selected according to the following three methods.

(1) All individuals are infeasible. In this case, the individuals should be guided to close to the feasible domain as soon as possible. The constrained violation degree represents the relationship between the individual and the feasible domain. The smaller the constrained violation degree is, the closer the individual is to the feasible region. Therefore the individual with the smallest constrained violation degree will be selected as X_{best}.

(2) Feasible individuals and infeasible individuals exist simultaneously. The individual with the smallest fitness can provide direction information of population evolution. While the individual with small constrained violation degree usually carry better location information, it can enhance the boundary search capability of the algorithm. Based on the above considerations, X_{best} is selected by Eq. (16) in this case.

$$\begin{cases} X_{best} = X(min(f)), rand \le p1 \\ X_{best} = X(min(G)), otherwise \end{cases} \tag{16}$$

where *X*(*min(G*)) represents the individual with the smallest constrained violation degree, and *X*(*min*(*f*)) represents the individual with the smallest fitness value in the current iteration.

(3)All individuals are feasible individuals. When all individuals are feasible, constrained optimization problem is transformed into unconstrained optimization problem. The individual with smallest fitness value is selected as X_{best} .

The pseudo code of the method of selecting the optimal individual X_{best} is shown below.

```
1.  if percent==0
2.        select the individual with smallest constrained violation degree;
3.      end if
4.      if percent>0& percent<1
5.        if rand<p1
6.           select the individual with smallest fitness;
7.        else
8.           select the individual with smallest constrained degree;
9.         end if
10.     end if
11.     if percent==1
12.           select the individual with smallest fitness;
13.     end if
```

In the above pseudo code, percent is equal to the number of feasible individuals over the number of all individuals, and p1 is a random number. The first 3 lines correspond to the situation that all individuals are infeasible, Lines 4 to 10 correspond to that feasible and infeasible individuals exist simultaneously, and lines 11 to 13 correspond to that all individuals are feasible.

Table 1. Details of 13 benchmark functions

Function	D	ρ/%	LI	NI	NE	α
g01	13	0.0111	9	0	0	6
g02	20	99.9971	0	2	0	1
g03	10	0.0000	0	0	1	1
g04	5	52.1230	0	6	0	2
g05	4	0.0000	2	0	3	3
g06	2	0.0066	0	2	0	2
g07	10	0.0003	3	5	0	6
g08	2	0.8560	0	2	0	0
g09	7	0.5121	0	4	0	2
g10	8	0.0010	3	3	0	6
g11	2	0.0000	0	0	1	1
g12	3	4.7713	0	9^3	0	0
g13	5	0.0000	0	0	3	3

5. Experiments and Discussion

5.1 Test Functions

The performance of proposed ε_SOSMS is tested on 13 well-known benchmark functions [16]. Table 1 exhibits the details of the 13 benchmark functions. In Table 1, D is the dimension of variables, ρ is the estimated ratio between the feasible region and the search space, LI is the number of linear inequality constrains, NI is the number of nonlinear inequality constrains, NE is the number of nonlinear equality constrains, and a is the number of constrained active at the optimal solution.

5.2 Experimental Setup

In order to verify the performance of ε_SOSMS, ε_SOSMS is compared with ε_SOS, MABC [8], εDE [9], IPES [11], and I-ABC [12]. ε_SOS is an improved SOS algorithm based on improved ε constrained of Section 4.2. The parameters of ε_SOSMS are set as follows: δ=0.0001, n=1.1, p1=0.8, p2=0.9. For each test function, each algorithm runs 30 times independently and the size of population of each algorithm is 50. εDE is stopped when function evaluation times reach 2.0×10^5, and the other algorithms are stopped when function evaluation times reach 2.4×10^4. All the experiments are executed on Windows 8.1 with Interl Core i5-3337u CPU 1.8 GHz and 4.0 GB RAM.

5.3 Experimental Results

The experimental results of ε_SOSMS and other algorithms are summarized in Table 2. The result of best, worst, mean and variance represent the best value, the worst value, the average value and standard deviation of 30 independent running results respectively. Where the first three indicators are used to compare the convergence accuracy, and the variance is used to compare the stability.

Table 2. Performance comparison of algorithms

Function/Theoretical optimum	Algorithm	Best	Worst	Mean	Variance
g01/-15.000	ε_SOSMS	-15.000	-15.000	-15.000	7.1E-18
	ε_SOS	-14.995	-14.849	-14.957	5.0E-02
	εDE	-15.000	-15.000	-15.000	5.8E-14
	I-ABC	-15.000	-15.000	-15.000	4.6E-16
	MABC	-15.000	-15.000	-15.000	00E+00
	IPES	-14.999	-14.999	-14.999	3.7E-15
g02/0.803619	ε_SOSMS	-0.799317	-0.785304	-0.799015	5.3E-04
	ε_SOS	-0.690979	-0.564224	-0.637003	3.9E-02
	εDE	-0.803619	-0.792608	-0.803004	2.5E-02
	I-ABC	-0.803619	-0.778278	-0.800094	7.1E-03
	MABC	-0.801982	-0.749797	-0.792412	1.2E-02
	IPES	-0.803607	-0.792771	-0.769193	7.9E-03
g03/-1.000	ε_SOSMS	-1.000	-1.000	-1.000	00E+00
	ε_SOS	-1.000	-0.937	-0.989	1.9E-02
	εDE	-1.000	-1.000	-1.000	3.9E-06
	I-ABC	-1.000	-1.000	-0.999	3.4E-06
	MABC	-1.000	-1.000	-1.000	00E+00
	IPES	-1.000	-1.000	-1.000	9.3E-06

Table 2. Continued

Function/Theoretical optimum	Algorithm	Best	Worst	Mean	Variance
g04/-30665.539	ε_SOSMS	-30665.539	-30665.539	-30665.539	00E+00
	ε_SOS	-30665.539	-30665.539	-30665.539	4.2E-12
	εDE	-30665.539	-30665.539	-30665.539	2.1E-05
	I-ABC	-30665.539	-30665.539	-30665.539	1.0E-05
	MABC	-30665.539	-30665.539	-30665.539	00E+00
	IPES	-30665.539	-30665.539	-30665.539	5.7E-12
g05/5126.498	ε_SOSMS	5126.498	5126.498	5126.498	2.3E-07
	ε_SOS	5126.551	5468.185	5245.576	1.1E+02
	εDE	5126.498	5126.498	5126.498	1.7E-05
	I-ABC	5126.498	5148.944	5131.861	5.5E+00
	MABC	5126.484	5438.387	5185.714	7.5E+01
	IPES	5126.498	5139.003	5197.991	2.4E+00
g06/-6961.814	ε_SOSMS	-6961.814	-6961.814	-6961.814	4.1E-13
	ε_SOS	-6961.747	-6953.748	-6959.364	2.5E+00
	εDE	-6961.814	-6961.814	-6961.814	2.3E-08
	I-ABC	-6961.814	-6961.814	-6961.814	1.8E-12
	MABC	-6961.814	-6961.805	-6961.813	2.0E-03
	IPES	-6961.814	-6961.814	-6961.814	3.8E-12
g07/24.306	ε_SOSMS	24.306	24.306	24.306	5.2e-03
	ε_SOS	24.869	34.704	28.442	3.0E+00
	εDE	24.306	24.306	24.306	6.3E-06
	I-ABC	24.311	24.677	24.366	6.9E-02
	MABC	24.330	25.190	24.473	1.9E-01
	IPES	24.307	24.316	24.333	6.6E-03
g08/-0.095825	ε_SOSMS	-0.095825	-0.095825	-0.095825	00E+00
	ε_SOS	-0.095825	-0.095825	-0.095825	5.9E-10
	εDE	-0.095825	-0.095825	-0.095825	8.4E-17
	I-ABC	-0.095825	-0.095825	-0.095825	2.8E-17
	MABC	-0.095825	-0.095825	-0.095825	00E+00
	IPES	24.307	24.316	24.333	6.6E-03
g09/680.630	ε_SOSMS	680.630	680.630	680.630	7.9E-05
	ε_SOS	681.374	687.418	683.451	2.3E+00
	εDE	680.630	680.630	680.630	2.2E-07
	I-ABC	680.631	680.637	680.633	1.5E-03
	MABC	680.634	680.653	680.640	4.0E-03
	IPES	680.630	680.673	680.639	4.3E-02
g10/7049.248	ε_SOSMS	7049.248	7158.173	7077.542	1.3E+01
	ε_SOS	7506.216	9816.488	8583.910	1.7E+02
	εDE	7049.248	7049.248	7049.248	9.0E-06
	I-ABC	7049.321	7285.383	7124.042	5.9E+01
	MABC	7053.904	7604.132	7224.407	1.3E+02
	IPES	7051.341	7210.360	7376.721	4.6E+02
g11/0.750	ε_SOSMS	0.750	0.750	0.750	00E+00
	ε_SOS	0.750	0.766	0.752	5.1E-03
	εDE	0.750	0.750	0.750	6.9E-14
	I-ABC	0.750	0.750	0.750	2.4E-06
	MABC	0.750	0.750	0.750	00E+00
	IPES	0.750	0.750	0.750	7.9E-04

Table 2. Continued

Function/Theoretical optimum	Algorithm	Best	Worst	Mean	Variance
g12/-1.000	ε_SOSMS	-1.000	-1.000	-1.000	00E+00
	ε_SOS	-0.293	-0.293	-0.293	6.8E-08
	εDE	-1.000	-1.000	-1.000	00E+00
	I-ABC	-1.000	-1.000	-1.000	00E+00
	MABC	-1.000	-1.000	-1.000	00E+00
	IPES	-1.000	-1.000	-1.000	00E+00
g13/0.053950	ε_SOSMS	0.053950	0.053953	0.053951	4.5E-06
	ε_SOS	0.054174	0.999820	0.314003	2.4E-01
	εDE	0.053954	0.054884	0.069631	7.6E-02
	I-ABC	0.053958	0.054144	0.055130	2.7E-04
	MABC	0.760	1.000	0.968	5.5E-02
	IPES	0.053950	0.146260	0.453029	1.1E-01

From Table 2, some conclusion can be obtained. Firstly, for all benchmark functions, ε_SOSMS achieves better results than ε_SOS. It shows that the improvement of evolutionary strategy of SOS in Section 4.2 is effective. Secondly, ε_SOSMS can find the optimal values of 12 test functions except g02. εDE did not find the theoretical optimal value only on g13. However fitness evaluation times of εDE is significantly more than ε_SOSMS. I-ABC finds theoretical optimal values only on 9 test functions. Compared with I-ABC, ε_SOSMS only obtains poor results on g02 function. MABC can find the theoretical optimal value of 6 benchmark functions. In summary, ε_SOSMS proposed in this paper is significantly superior to the other four algorithms in the ability of finding theoretical optimal values. Thirdly, except for the function g10, the variance of ε_SOSMS is almost equal to 0. In addition, for most benchmark functions, the variance obtained by ε_SOSMS is less than other algorithms. It shows that the proposed algorithm has better stability than other algorithms.

Table 3. The number of fitness evaluations

Function	ε_SOSMS		MABC	
	FES	SR/%	FES	SR/%
g01	70990	100	105370	100
g02	NA	0	367360	16.67
g03	72957	100	98560	100
g04	24530	100	74380	100
g05	97430	100	308320	63.33
g06	12190	100	140160	96.67
g07	73051	100	NA	0
g08	1890	100	2750	100
g09	71763	100	NA	0
g10	176820	66.67	NA	0
g11	32418	100	4730	100
g12	4760	100	5280	100
g13	46803	100	NA	0

In order to verify the convergence speed of ε_SOSMS, ε_SOSMS is compared with MABC which has better optimization performance among the above contrast algorithms. These two algorithms run 30 times independently. The results are shown in Table 3. FES represents the average value of minimum function evaluation times that are needed to find the theoretical optimum value for each algorithm, and SR represents the success rate of reaching the theoretical optimal value in 30 independent experiments. NA shows that the algorithm does not find the theoretical optimal value when function evaluation times reach 2.4×10^4.

From Table 3, ε_SOSMS can find the optimal value on 12 benchmark functions, the success rate of ε_SOSMS on the 10 test functions is 100%. But MABC can find the optimal value on 9 benchmark functions, and the success rate of MABC on only 5 test functions is 100%. It shows that ε_SOSMS is superior to MABC in robustness. In addition, except $g02$ and $g11$, function evaluation times that is needed to find the theoretical optimum value by ε_SOSMS is significantly less than MABC, which shows that ε_SOSMS is superior to MABC in convergence speed.

Through the above experiments, we can find that ε_SOSMS has obvious advantages in the optimization ability and convergence speed.

6. Conclusions

In this paper, an improved Symbiotic Organisms Search algorithm with mixed strategy based on adaptive ε constrained method (ε_SOSMS) is proposed. Firstly, an adaptive ε constrained is proposed to take full advantage of infeasible individuals, which can improve searching capabilities in the search space and avoid algorithms falling into local optimum. Secondly, the evolutionary strategy of SOS is improved to use the optimal individual for accelerating the convergence speed and search capabilities. Experiments on 13 benchmark functions show that ε_SOSMS has better performance than the other algorithms.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (No. 61501107), the Education Department of Jilin province science and technology research project of "13th Five-Year" in2016 (No. 95), and the Project of Scientific and Technological Innovation Development of Jilin (No. 201750219, 201750227).

References

[1] Z. Y. Li, T. Huang, S. M. Chen, and R. F. Li, "Overview of constrained optimization evolutionary algorithms," *Journal of Software*, vol. 28, no. 6, pp. 1529-1546, 2017.

[2] D. H. Xia, Y. X. Li, W. Y. Gong, and G. L. He, "An adaptive differential evolution algorithm for constrained optimization problems," *Acta Electronica Sinica*, vol. 44, no. 10, pp. 2535-2542, 2016.

[3] H. C. Liu and Z. J. Wu, "Differential evolution algorithm using rotation-based learning," *Acta Electronica Sinica*, vol. 31, no. 10, pp. 2040-2046, 2015.

[4] H. Zhou. H. Zhao, M. Li, and Y Cai, "Multi-strategy adaptive symbiotic organisms search algorithm," *Journal of Air Force Engineering University (Natural Science Edition)*, vol. 17, no. 4, pp. 101-106, 2016.

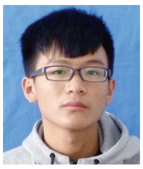
[5] A. K. Ojha and Y. R. Naidu, "Hybridizing particle swarm optimization with invasive weed optimization for solving nonlinear constrained optimization problems," in *Proceedings of Fourth International Conference on Soft Computing for Problem Solving*. New Delhi, India: Springer, 2015, pp. 599-610.
[6] W. Gong, Z. Cai, and D. Liang, "Adaptive ranking mutation operator based differential evolution for constrained optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 4, pp. 716-727, 2014.
[7] W. Long, W. Z. Zhang, Y. F. Huang, and Y. X. Chen, "A hybrid cuckoo search algorithm with feasibility-based rule for constrained structural optimization," *Journal of Central South University*, vol. 21, no. 8, pp. 3197-3204, 2014.
[8] D. Karaboga and B. Akay, "A modified artificial bee colony (ABC) algorithm for constrained optimization problems," *Applied Soft Computing*, vol. 11, no. 3, pp. 3021-3031, 2011.
[9] J. G. Zheng, X. Wang, and R. H. Liu, "Epsilon-differential evolution algorithm for constrained optimization problems," *Journal of Software*, vol. 23, no. 9, pp. 2374-2387, 2012.
[10] X. J. Bi and L. Zhang, "Self-adaptiveεconstrained optimization algorithm," *Systems Engineering and Electronics*, vol. 37, no. 8, pp. 1909-1915, 2015.
[11] C. G. Cui and X. F. Yang, "Interior penalty rule based evolutionary algorithm for constrained optimization," *Journal of Software*, vol. 26, no.7, pp. 1688-1699, 2015.
[12] Y. Liang, Z., Wan, and D. Fang, "An improved artificial bee colony algorithm for solving constrained optimization problems," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 3, pp. 739-754, 2017.
[13] Y. Wang and Z. Cai, "Combining multiobjective optimization with differential evolution to solve constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 117-134, 2012.
[14] T. Takahama and S. Sakai, "Efficient constrained optimization by the ε constrained adaptive differential evolution," in *Proceedings of IEEE Congress on Evolutionary Computation*, Barcelona, Spain, 2010, pp. 1-8.
[15] M. Y. Cheng and D. Prayogo, "Symbiotic organisms search: a new metaheuristic optimization algorithm," *Computers & Structures*, vol. 139, pp. 98-112, 2014.
[16] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284-294, 2000.

Yanjiao Wang **https://orcid.org/0000-0001-7383-0521**

She received her M.S. degree in Signal and Information Processing from Harbin Engineering University in 2010. She received the Ph.D. degree in Signal and Information Processing from Harbin Engineering University in 2013. After then, she has become a teacher of Northeast Electric Power University, and her main research interests include evolutionary computing, and intelligent information processing.

Huanhuan Tao **https://orcid.org/0000-0002-4841-0645**

She is currently the master's degree in School of Electrical Engineering with Northeast Electric Power University. Her research interests are evolutionary computing and intelligent information processing.



Zhuang Ma **https://orcid.org/0000-0002-4209-5699**

He is currently the master's degree in School of Electrical Engineering with Northeast Electric Power University. His research interests are evolutionary computing and intelligent information processing.