The Journal of Supercomputing (2020) 76:5609–5635
https://doi.org/10.1007/s11227-019-03098-9

Grasshopper optimization algorithm with principal component analysis for global optimization

Xiaofeng Yue[1] · Hongbo Zhang[1]

Published online: 3 December 2019

Abstract
As one of the latest meta-heuristic algorithms, the grasshopper optimization algorithm (GOA) has extensive applications because of its efficiency and simplicity. However, the basic GOA still has enough room for improvement. Therefore, a new variant GOA algorithm which combines two strategies, namely PCA–GOA, is proposed. Firstly, principal component analysis strategy is employed to obtain the grasshoppers with minimally correlated variables, which can improve the exploitation capability of the GOA. Then, a novel inertia weight is proposed to balance exploration and exploitation in an intelligent way, which makes the GOA to have better search capability. Furthermore, the performance of PCA–GOA is evaluated by solving a series of benchmark functions. The experimental results manifest that the PCA–GOA provides better outcomes than the basic GOA and other state-of-the-art algorithms on the majority of functions, which demonstrates the superiority of the PCA–GOA.

Keywords Grasshopper optimization algorithm · Principal component analysis · Novel inertia weight · Global optimization

1 Introduction

Optimization is used to select the best options among all the solutions, and it exists in many fields such as scheduling problem, parameter estimation, materials prediction and structure design [1–5]. However, it is difficult to solve the high dimension optimization problems with the traditional optimization algorithms, and this issue has drawn much attention of researchers [6]. It is reported that the heuristic and meta-heuristic algorithms show superiority on complex optimization tasks. The majority of these algorithms are proposed based on natural phenomena. Some of the

✉ Xiaofeng Yue
yuexf@ccut.edu.cn

1 School of Mechatronic Engineering, Changchun University of Technology, Changchun 130012, People's Republic of China

typical algorithms are genetic algorithm (GA) which is inspired from genetic mechanism [7], and particle swarm optimization (PSO) is proposed based on the behavior of birds [8]. Some recent optimization algorithms employ the ant colony optimization (ACO) [9], artificial bee colony algorithm (ABC) [10], firefly algorithm (FA) [11], cuckoo search (CS) [12], gravitational search algorithm (GSA) [13], bat algorithm (BA) [14], gray wolf optimization (GWO) [15] and moth-flame optimization algorithm (MFO) [16]. In addition, modifying the existing algorithms is an important work, and it helps using the hardware better. This point is also meaningful to supercomputing [17]. Graphics processor units (GPUs) have been widely used for large number computation due to its high performance [18]. If the optimization algorithm runs on GPU, the execution time will be shorter [19].

The grasshopper optimization algorithm (GOA) is newly proposed to solve optimization problems. The advantages of the GOA have been proven by comparing with some well-known and latest algorithms [20]. Therefore, practitioners and researchers are interested with employing the GOA to solve the various optimization problems. Aljarah et al. [21] used the GOA to solve the feature selection and support vector machine problem. Zhang et al. [22] applied GOA to analyze vibration signals. Wu and Wang [23] had applied adaptive GOA to trajectory optimization problem. Hekimoğlu and Ekinci [24] applied GOA to automatic voltage regulator system. Łukasik et al. [25] used GOA to solve data clustering problem. Lal et al. [26] applied GOA to optimize fuzzy PID controller. Fathy [27] used GOA to solve reconfiguration of PV array problem.

The basic GOA is a promising algorithm and has been successfully applied to many fields. However, the basic GOA has some shortcomings, such as limited exploitation capability and premature [28]. Therefore, many researchers are interested in improving the GOA. Ewees et al. [29] proposed a novel improved GOA which combined opposition-based learning (OBL) strategy. The outstanding performance of the proposed GOA was proved by benchmark functions and engineering problems. Saxena et al. [30] used chaotic strategy with 10 maps to enhance bridging mechanism of GOA. The results proved the efficiency of the improved GOA. In multilevel thresholding image segmentation, Liang et al. [31] presented a modified GOA, which combined the Levy flight algorithm. To overcome the drawbacks of the basic GOA algorithm, Luo et al. [32] proposed an improved GOA and applied to financial stress predicting problem. Three strategies named Gaussian mutation, Levy flight and opposition-based learning were integrated into GOA. The experiment results indicated that the modifications can significantly improve the basic GOA.

As we can see above, many versions of GOA have been proposed. However, the enhanced GOA algorithms are far from perfect. To further improve the GOA, an enhanced grasshopper optimization algorithm, namely PCA–GOA, is proposed. The main contributions of the paper are as follows:

(a) In the PCA–GOA, the principal component analysis is incorporated to generate the new population with high quality. Then, the generated population is used to replace the population with bad fitness values. This operation is able to improve the exploitation capability of the GOA.

(b) The novel inertia weight is introduced to control the search process in an intelligent way. Under this strategy, the grasshoppers with better fitness values move quickly to the target grasshopper and the grasshoppers with worse fitness values maintain strong exploration capabilities. Therefore, novel inertia weight plays a role in both enhancing the exploration and exploitation capability.

The structure of the rest of this paper is as follows. Section 2 presents the basic GOA algorithm. Section 3 shows the proposed PCA–GOA algorithm. Section 4 provides discussions and analyses of the experiment results. Section 5 concludes the paper.

2 Basic grasshopper optimization algorithm

The basic grasshopper optimization algorithm (GOA) was originated by Saremi and Mirjalili [20], which mimics the behavior of grasshopper. The mathematical model of the GOA can be expressed as follows:

$$X_i = S_i + G_i + A_i \tag{1}$$

where X_i is the position of the i-th grasshopper, S_i represents the social interaction, G_i denotes the gravity force of the i-th grasshopper, A_i refers the wind advection.

$$S_i = \sum_{\substack{j=1 \\ j\neq i}}^{N} s\left(d_{ij}\right)\frac{x_j - x_i}{d_{ij}} \tag{2}$$

$$d_{ij} = \left|x_j - x_i\right| \tag{3}$$

where x_i and x_j are the positions of i-th and j-th, respectively. d_{ij} denotes the distance of i-th and j-th grasshoppers. The s function represents social force is given as follows:

$$s(r) = fe^{-\frac{r}{l}} - e^{-r} \tag{4}$$

where f is the intensity of attraction, l denotes the attractive length scale.

$$G_i = -g\hat{e}_g \tag{5}$$

where g denotes the gravitational constant, $\hat{e}_g$ is unity vector.

$$A_i = u\hat{e}_w \tag{6}$$

where u denotes a constant drift, $\hat{e}_w$ represents a unity vector.

The mathematical model can be written as:

$$X_i = \sum_{\substack{j=1 \\ j\neq i}}^{N} s\left(\left|x_j - x_i\right|\right)\frac{x_j - x_i}{d_{ij}} - g\hat{e}_g + u\hat{e}_w \tag{7}$$

where N defines the number of grasshoppers.

In order to solve the optimization problem in a better way, a modified GOA is expressed as follows:

$$X_i^d = c\left\{\sum_{\substack{j=1 \\ j\neq i}}^{N} c\frac{ub_d - lb_d}{2}s\left(\left|x_j^d - x_i^d\right|\right)\frac{x_j - x_i}{d_{ij}}\right\} + \hat{T}_d \tag{8}$$

where ub_d and lb_d are the upper bound and lower bound, $\hat{T}_d$ is a target grasshopper. c is the parameter, which used to balance exploration and exploitation. c can be given by:

$$c = c_{\max} - l\frac{c_{\max} - c_{\min}}{L} \tag{9}$$

where $c_{\max}$ and $c_{\min}$ are the maximum value and the minimum value, l is the number of current iterations, L denotes the maximum number of iterations.

3 The proposed PCA–GOA algorithm

3.1 Principal component analysis

In basic GOA, the grasshoppers fly around the target grasshopper, which leads redundancy that reduces the diversity of population. More importantly, the grasshoppers with bad fitness have the low possibility to reach the optimal solution. In this work, principal component analysis is introduced to solve this problem. Principal component analysis is an important statistical analysis method, which has two main functions that are data reduction and interpretation [33, 34]. The minimally correlated variables can be transformed from correlated variables by principal component analysis [35]. In the PCA–GOA algorithm, the new generated solutions are minimally correlated and show the information of original solutions. Furthermore, the new solutions may have better quality than the original solutions, which enhance the exploitation capability of the GOA. Let that $U = \left(U_1, U_2 \ldots, U_p\right)'$ is an exemplar of the original data set, and it can be expressed as follows:

$$U = \begin{pmatrix} u_{11} & u_{12} & . & . & . & u_{1n} \\ u_{21} & u_{22} & . & . & . & u_{2n} \\ . & . & . & & & . \\ . & . & & . & & . \\ . & . & & & . & . \\ u_{p1} & u_{p2} & . & . & . & u_{pn} \end{pmatrix} = \begin{pmatrix} U_1 \\ U_2 \\ . \\ . \\ . \\ U_p \end{pmatrix} \tag{10}$$

where p represents the number of exemplars. The linear transformations can be calculated as follows:

$$\begin{cases} Z_1 = a_1^{'}U = a_{11}U_1 + a_{12}U_2 + \dots a_{1p}U_p, \\ Z_2 = a_2^{'}U = a_{21}U_1 + a_{22}U_2 + \dots a_{2p}U_p, \\ \qquad \dots \\ Z_p = a_p^{'}U = a_{p1}U_1 + a_{p2}U_2 + \dots a_{pp}U_p. \end{cases} \tag{11}$$

where a_i represents the coefficient vector. The linear transformation model can be represented in a simple way as follows:

$$Var\left(Z_i\right) = a_i^{'} \sum a_i, \quad i = 1, 2 \dots p \tag{12}$$

$$Cov\left(Z_i, Z_j\right) = a_i^{'} \sum a_j, \quad i,j = 1, 2 \dots p \tag{13}$$

It is important to choose the number m of principal components. The m is defined as follows:

$$\frac{\lambda_1 + \lambda_2 + \cdots \lambda_m}{\sum_{i=1}^{s} \lambda_i} \geq \delta \tag{14}$$

where δ is contribution rate. In this work, we use $\delta = 0.85.\lambda_1$, λ_2, λ_m denote eigenvalues of covariance matrix.

Suppose the covariance matrix of grasshoppers $X = \left\{x_1^t, x_2^t, \dots x_n^t\right\}$ is V, principal population can be generated as follows:

$$\begin{cases} F_1^t = a_1^{'}X = a_{11}x_1^t + a_{12}x_2^t + \cdots a_{1n}x_{n,}^t, \\ F_2^t = a_2^{'}X = a_{21}x_1^t + a_{22}x_2^t + \cdots a_{2n}x_{n,}^t, \\ \qquad \dots \\ F_m^t = a_m^{'}X = a_{m1}x_1^t + a_{m2}x_2^t + \cdots a_{mn}x_{n,}^t. \end{cases} \tag{15}$$

where $(a_1^{'}, a_2^{'}, \dots, a_m^{'})$ denote feature vectors of V. According to the principal component analysis theory, the new generated population is uncorrelated. The individuals of bad fitness are substituted by principal population. Figure 1 shows the flow of principal component analysis operation.

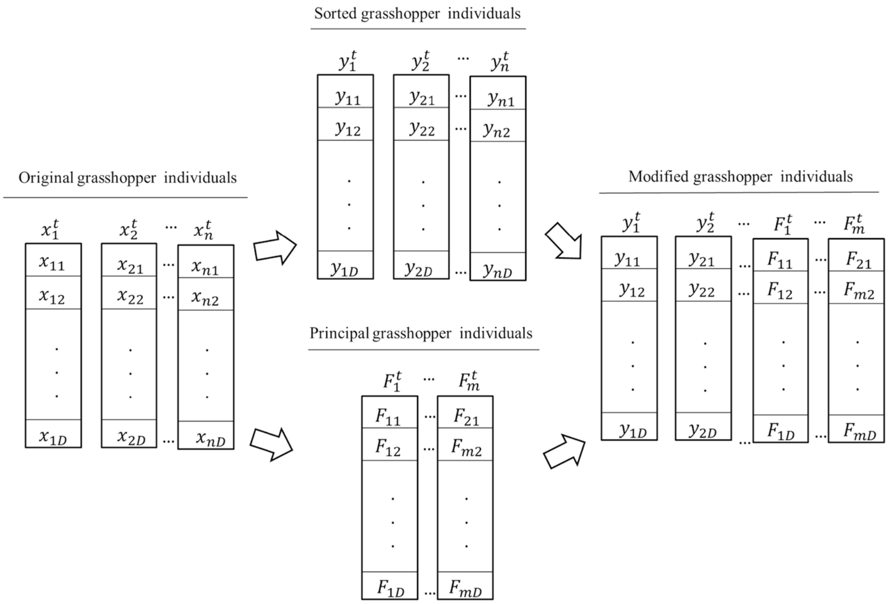

Fig. 1 Flowchart of principal component analysis operation

3.2 Novel inertia weight

In basic GOA algorithm, the parameter c plays an important role in balancing exploration and exploitation, which decreased with the number of iterations in a linear way. However, all the grasshoppers take the same parameter c without considering the fitness of each grasshopper. In this paper, we propose a novel inertia weight, which takes different strategies based on the fitness values of grasshoppers. To guarantee the steady of searching process, one of c is replaced. The novel inertia weight is calculated as follows:

$$\begin{cases} c_i^k = c_{\max} - \left(c_{\max} - c_{\min}\right) * \left(\frac{l}{L}\right) & if \quad fitness(i) \geq average \\ c_i^k = c_{\max} - \left(c_{\max} - c_{\min}\right) * \left[\frac{2l}{L} - \left(\frac{l}{L}\right)^2\right] & if \quad fitness(i) < average \end{cases} \tag{16}$$

The modified mathematical model can be expressed by:

$$X_i^d = c\left\{\sum_{\substack{j=1 \\ j\neq i}}^{N} c_i^k \frac{ub_d - lb_d}{2} s\left(\left|x_j^d - x_i^d\right|\right) \frac{x_j - x_i}{d_{ij}}\right\} + \hat{T}_d \tag{17}$$

The flowchart of the proposed PCA–GOA is shown in Fig. 2.

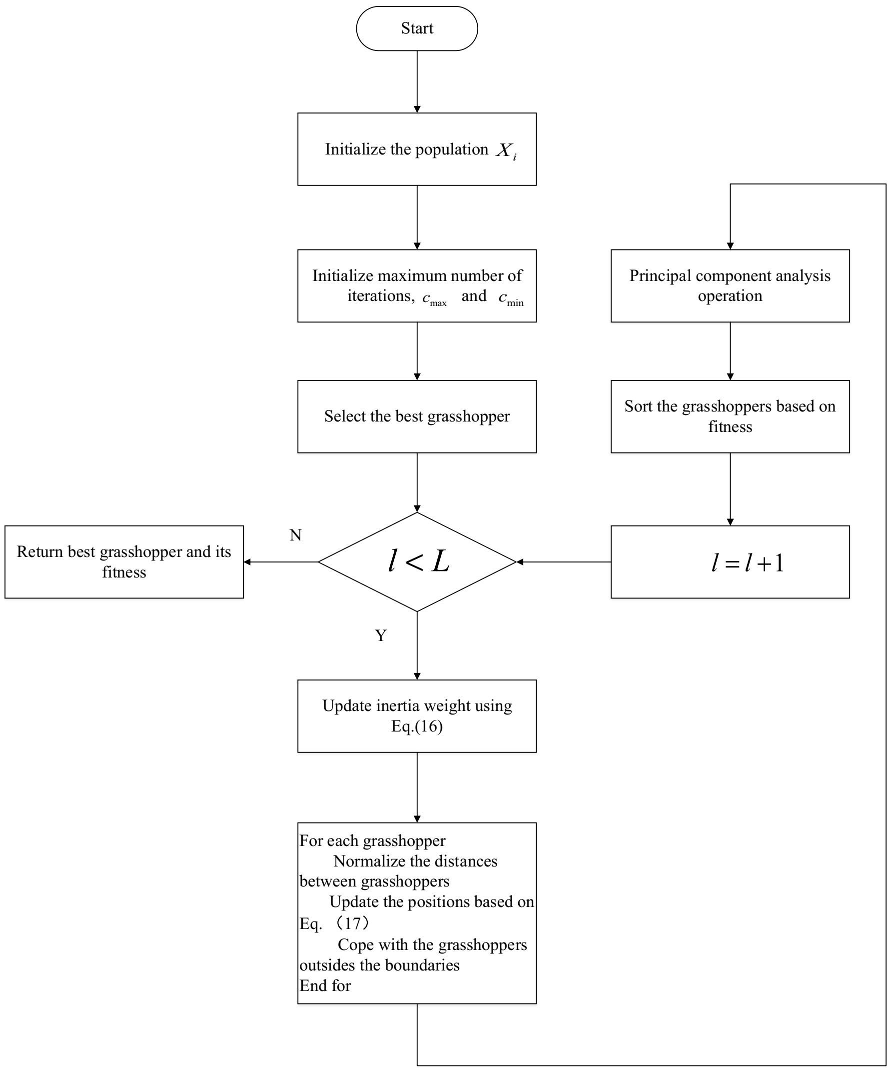

Fig. 2 Flowchart of the proposed PCA–GOA algorithm

4 Experimental results and discussion

In this section, unimodal benchmark functions, multimodal benchmark functions and fixed-dimension multimodal benchmark functions are used to test the performance of the PCA–GOA [36–39]. The benchmark functions are provided in Tables 1, 2 and 3. The exploitation of an algorithm can be tested by unimodal benchmark functions $\left(f_1 - f_7\right)$ as only one global is contained. Multimodal benchmark functions $\left(f_8 - f_{13}\right)$ and fixed-dimension multimodal benchmark functions $\left(f_{14} - f_{23}\right)$ are efficient tool for evaluating exploration capability and local optima

Table 1 Unimodal benchmark functions

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
$f_4(x) = max_i \{ \lvert x_i \rvert, 1 \leq i \leq n \}$	30	[− 100, 100]	0
$f_5(x) = \sum_{i=1}^{n-1} \left[100 (x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right]$	30	[− 30, 30]	0
$f_6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	30	[− 100, 100]	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	[− 1.28, 1.28]	0

avoidance. However, fixed-dimension multimodal benchmark functions are different from multimodal benchmark functions in dimensionality.

4.1 Parameters analyze

Parameters setting can influence the performance of the PCA–GOA when solving the various optimization problems. In this study, the five parameters of PCA–GOA include maximum parameter $c_{\max}$, minimum parameter $c_{\min}$, number of search agents N, maximum number of iterations L, contribution rate δ are analyzed on benchmark function f_1. An orthogonal test is worked as a tool to show the relations between the performance and each parameter setting. Ranges of PCA–GOA parameters are described in Table 4. According to the number of factors and levels, we select the $L_{27}(3^{13})$ type to complete the task. To make the results more convincing, the average values after 30 runs are used to make comparisons.

As shown in Table 5, the maximum number of iterations L is the most important factor that affects the search capability of the PCA–GOA. Furthermore, the importance of the other factors is as follows: $c_{\max} > c_{\min} > N > \delta$. Figure 3 shows the relation between average fitness values and the five parameters. From Table 5, we can obtain the best combination is $c_{\max} = 1$, $c_{\min} = 0.00001$, $N = 50$, $L = 300$, $\delta = 0.85$.

4.2 Performance evaluation

Six typical and recent algorithms including particle swarm optimization (PSO) [8], bat algorithm (BA) [14], ant lion optimizer (ALO) [40], dragonfly algorithm (DA) [41], grasshopper optimization algorithm (GOA) [20] and chaotic grasshopper optimization algorithm (CGOA) [28] are employed to compare the performance of the PCA–GOA. The key parameters of the seven algorithms are shown

Table 2 Multimodal benchmark functions

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 × Dim
$f_9(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$	30	[− 5.12, 5.12]	0
$f_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right)+20+e$	30	[− 32, 32]	0
$f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	[− 600, 600]	0
$f_{12}(x)=\frac{\pi}{n}\left\{10\sin\left(\pi y_1\right)+\sum_{i=1}^{n-1}\left(y_i-1\right)^2\left[1+10\sin^2\left(\pi y_{i+1}\right)\right]+\left(y_n-1\right)^2\right\}+\sum_{i=1}^{n}u\left(x_i,10,100,4\right)$ $y_i=1+\frac{x_i+1}{4}$	30	[− 50, 50]	0
$f_{13}(x)=\sum_{i=1}^{n}0.1\left\{\sin^2\left(3\pi x_1\right)+\sum_{i=1}^{n}\left(x_i-1\right)^2\left[1+\sin^2\left(3\pi x_i+1\right)+\left(x_n-1\right)^2\left[1+\sin^2\left(2\pi x_n\right)\right]\right\}+\sum_{i=1}^{n}u\left(x_i,5,100,4\right)$	30	[− 50, 50]	0

Table 3 Fixed-dimension multimodal benchmark functions

Function	Dim	Range	$f_{\min}$
$f_{14}(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}\left(x_i-a_{ij}\right)^6}\right)^{-1}$	2	[−65, 65]	1
$f_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1\left(b_i^2+b_ix_2\right)}{b_i^2+b_ix_3+x_4}\right]^2$	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
$f_{18}(x)=\left[1+\left(x_1+x_2+1\right)^2\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right]$ $\times\left[30+\left(2x_1-3x_2\right)^2\times\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right]$	2	[−2, 2]	3
$f_{19}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{3}a_{ij}\left(x_j-p_{ij}\right)^2\right)$	3	[1, 3]	−3.86
$f_{20}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{ij}\left(x_j-p_{ij}\right)^2\right)$	6	[0, 1]	−3.32
$f_{21}(x)=-\sum_{i=1}^{5}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	4	[0, 10]	−10.1532
$f_{22}(x)=-\sum_{i=1}^{7}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	4	[0, 10]	−10.4028
$f_{23}(x)=-\sum_{i=1}^{10}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	4	[0, 10]	−10.5363

Table 4 Ranges of the PCA–GOA parameters

Level	parameter				
	$c_{\max}$	$c_{\min}$	N	L	δ
1	1	0.0000001	10	200	0.65
2	1.5	0.000001	30	250	0.75
3	2	0.00001	50	300	0.85

in Table 6. To test the generality of the PCA–GOA, different number of iterations and population sizes are used to make comparisons. The experiments are first tested on $N = 20$ and $L = 300$. Then, $N = 30$ and $L = 1000$ are used to further assess the performance of the proposed PCA–GOA. Each process independently runs 30 times. Best fitness values, worst fitness values, average fitness values and standard deviation values are recorded. It is obvious that the lower value means better search capability of the algorithm. For the drawbacks of the four indices cannot measure whether the results are significant, we use Wilcoxon statistical test to perfect comparisons [42, 43]. If $p < 0.05$, it demonstrates the improving

Table 5 $L_{27}(3^{13})$ orthogonal testing for f_1

No.	c_{max}	c_{min}	N	L	δ	Average fitness value
1	1	1	1	1	1	67.5811
2	1	1	1	1	2	58.7860
3	1	1	1	1	3	22.7077
4	1	2	2	2	1	1.79E−08
5	1	2	2	2	2	1.20E−08
6	1	2	2	2	3	1.06E−08
7	1	3	3	3	1	1.60E−08
8	1	3	3	3	2	1.16E−08
9	1	3	3	3	3	6.57E−09
10	2	1	2	3	1	3.12E−08
11	2	1	2	3	2	2.24E−08
12	2	1	2	3	3	1.74E−08
13	2	2	3	1	1	3.43E−06
14	2	2	3	1	2	3.05E−07
15	2	2	3	1	3	6.08E−07
16	2	3	1	2	1	1.34E−08
17	2	3	1	2	2	5.16E−08
18	2	3	1	2	3	8.30E−09
19	3	1	3	2	1	3.36E−05
20	3	1	3	2	2	1.63E−05
21	3	1	3	2	3	1.12E−05
22	3	2	1	3	1	1.36E−06
23	3	2	1	3	2	1.00E−08
24	3	2	1	3	3	9.12E−09
25	3	3	2	1	1	3.28E−05
26	3	3	2	1	2	5.87E−05
27	3	3	2	1	3	5.35E−05
K1	16.5639	16.5639	16.5639	16.5639	7.5090	
K2	4.99E−07	6.40E−07	1.61E−05	6.80E−06	6.5318	
K3	2.31E−05	1.61E−05	7.28E−05	1.65E−07	2.5231	
Rank	2	3	4	1	5	

affect is significant. Furthermore, CPU time and allocated memory are used to evaluated the computation amount of the PCA–GOA.

The details of experiments environment are as follows: Core (TM) i5-4590 CPU @ 3.30 GHz with 8 GB RAM and 64 bit under Windows 7 system using Matlab R2015a.

Tables 7, 8 and 9 illustrate the results of unimodal benchmark functions, multimodal benchmark functions and fixed-dimension multimodal benchmark functions, respectively. In addition, the best results are highlighted in bold.

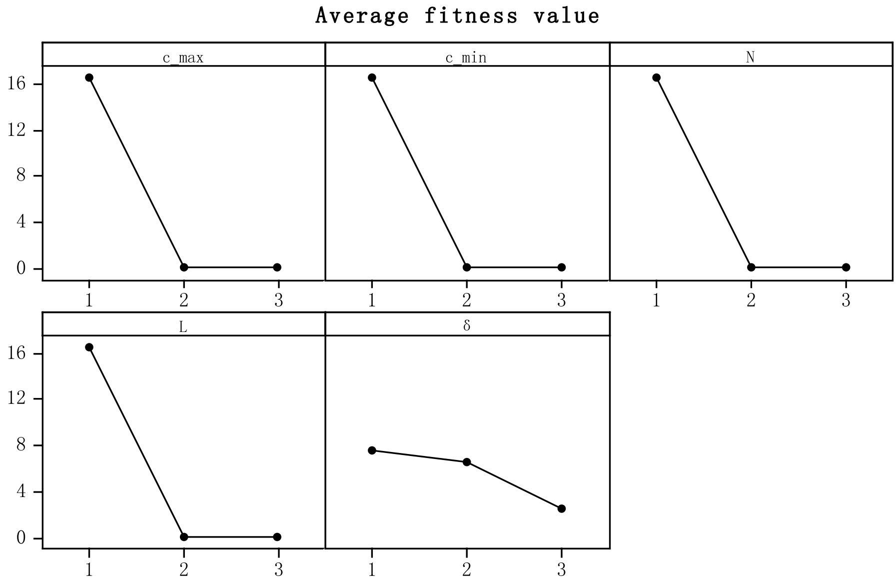

Fig. 3 Factor-index diagram

Table 6 Parameters setting for chosen optimization algorithms

Algorithm	Parameter	Value
PSO	Learning coefficient c_1	2
	Learning coefficient c_2	2
BA	Pulse rate R^0	0.5
	Loudness A^0	[1, 2]
	Minimum frequency value f_{min}	0
	Maximum frequency value $f_{\max}$	2
	Constant α	0.9
	Constant γ	0.9
ALO	Constant w_1	2, 3, 4, 5, 6
DA	Inertial weight w_2	09–0.4
	Separation weight s	2
	Alignment weight a	2
	Cohesion weight c	2
	Food factor f	2
	Enemy factor e	1
GOA	Minimum inertial weight $c_{\min}$	0.00001
	Maximum inertial weight $c_{\max}$	1
CGOA	Minimum inertial weight $c_{\min}$	0.00001
	Maximum inertial weight $c_{\max}$	1
	Adjustable parameter P	1
PCA–GOA	Minimum inertial weight $c_{\min}$	0.00001
	Maximum inertial weight $c_{\max}$	1
	Contribution rate δ	0.85

Table 7 Results of the unimodal benchmark functions

Functions	Result	Algorithm						
		PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_1	Best	44.7456	455.994	16.0323	2144.79	664.803	1023.80	**6.85E−11**
	Worst	10784.6	14858.1	1553.39	11,180.4	4806.42	3503.01	**6.55E−08**
	Mean	2095.24	5353.58	335.569	4480.67	1952.52	2049.01	**3.44E−09**
	Std	3000.54	3907.08	337.773	2134.99	927.865	613.662	**1.16E−08**
f_2	Best	10.5146	54.2206	6.53531	11.1674	11.8259	60.0089	**5.15E−06**
	Worst	106.546	94,409.5	994.336	54.7302	773,505	81,672.2	**143.459**
	Mean	54.0944	10,477.9	247.819	28.0580	27,308.8	5591.34	**22.2980**
	Std	20.8137	23,519.1	264.388	9.62,218	138,686	15,871.7	**45.3571**
f_3	Best	11,829.6	2511.01	4187.69	10,118.9	624.001	966.341	**7.70E−10**
	Worst	50,884.4	38,262.5	23,761.0	54,449.6	13,596.9	12,724.9	**7.17E−08**
	Mean	32,828.3	13,235.7	12,708.2	24,173.7	3281.71	4569.06	**1.73E−08**
	Std	9953.85	9023.05	3905.39	10,238.8	2965.31	3423.73	**1.65E−08**
f_4	Best	6.01133	18.8992	13.1275	21.8091	9.72136	9.98620	**4.50E−06**
	Worst	21.4396	55.4018	44.6687	54.7413	23.9253	23.3176	**5.24E−05**
	Mean	13.4216	37.1428	25.9201	37.6442	14.8801	14.5883	**1.75E−05**
	Std	3.84251	7.89345	6.27847	7.83138	3.28203	3.41408	**1.03E−05**
f_5	Best	6037.06	228.081	755.068	116025	17,855.2	20,930.9	**1.72E−08**
	Worst	2.00E+06	3320.95	327,120	8.65E+06	1.15E+06	2.32E+06	**28.7056**
	Mean	198,765	1058.57	30,933.4	2.24E+06	220,170	257,903	**26.7876**
	Std	364,163	846.992	63,529.6	2.25E+06	230,635	398,938	**7.15928**
f_6	Best	24.2599	372.793	4.41444	290.378	487.592	530.482	**1.64E−10**
	Worst	11,482.2	17,131.8	833.446	9192.13	3589.19	4309.04	**2.04E−08**
	Mean	2302.58	6024.18	286.438	4344.91	2056.71	1937.23	**2.93E−09**
	Std	3067.55	3413.35	258.718	2501.43	797.414	807.635	**4.82E−09**
f_7	Best	0.06860	88.6127	0.56282	0.40922	6.11739	3.80125	**0.05884**
	Worst	21.8953	170.189	1.61972	4.82324	65.6534	70.2914	**0.43351**
	Mean	2.63533	131.852	1.02894	2.10081	35.0712	32.6918	**0.18395**
	Std	4.68135	20.9115	0.31498	1.13123	14.4034	16.1802	**0.11214**

Table 7 summarizes the best fitness values, worst fitness values, average fitness values and standard deviation values achieved by PSO, BA, ALO, DA, GOA, CGOA and PCA–GOA. As the results reported in Table 7, the PCA–GOA provides the best results on the four indexes for f_1,f_2,f_3,f_4,f_5,f_6, and f_7, which shows the PCA–GOA has remarkable advantages in terms of unimodal test functions. The superiority of the PCA–GOA can be further proved by Table 10, because all p values are much less than 0.05. Therefore, we can conclude that the PCA–GOA has a high exploitation capability.

Observed from Table 8, it is apparent that PCA–GOA obtains better results for the majority functions except f_8 and f_9. For f_8, the PCA–GOA provides the best standard deviation values. It can be seen from Table 11, the PCA–GOA is markedly

Table 8 Results of the multimodal benchmark functions

Functions	Result	Algorithm						
		PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_8	Best	−7769.29	−7050.67	**−9830.28**	−6470.43	−7490.62	−7831.41	−7150.53
	Worst	−4146.60	−4716.06	**−5417.67**	−3586.03	−4428.24	−5195.25	−4695.32
	Mean	−5943.05	−5821.45	−5662.05	−5274.33	−6475.71	**−6595.77**	−5529.99
	Std	861.019	589.544	849.241	662.771	743.912	746.452	**506.517**
f_9	Best	125.992	256.280	**42.2945**	144.852	166.680	201.957	229.583
	Worst	318.843	373.845	**180.133**	287.057	313.132	326.242	353.799
	Mean	224.536	313.748	**93.3989**	196.020	243.580	251.374	298.371
	Std	47.3705	30.4101	**29.2119**	30.3164	33.1221	32.9490	32.0139
f_{10}	Best	6.32459	2.73882	8.68946	5.73971	7.19639	7.47218	**1.41E−06**
	Worst	19.9668	19.2956	**15.0760**	15.1013	20.1273	20.1652	19.9668
	Mean	11.2330	8.19860	12.9238	12.7004	17.3568	17.2874	**1.99241**
	Std	4.13117	6.55045	**1.60036**	1.79528	4.02749	4.26611	5.97723
f_{11}	Best	1.59920	112.612	1.01402	13.6865	6.12841	7.65095	**2.42E−10**
	Worst	97.5310	408.965	21.4825	78.4671	34.1774	59.9066	**7.39E−09**
	Mean	13.4052	269.125	4.07285	41.4504	17.5167	20.5439	**1.95E−09**
	Std	17.3527	77.2821	4.37847	16.6670	6.61021	10.7147	**1.53E−09**
f_{12}	Best	1.74612	12.4056	10.4007	14.0529	2.24718	5.34439	**5.25E−13**
	Worst	8210.38	53.5028	69.8349	3.85E+06	24.5174	31.5947	**14.2755**
	Mean	296.113	31.5681	29.2943	4.09E+05	11.2449	12.7267	**1.41507**
	Std	1470.35	10.3548	13.4668	880485	4.73019	6.28719	**3.94785**
f_{13}	Best	4.77693	1.28086	39.8757	644.449	18.1480	26.0738	**3.56E−12**
	Worst	1.07E+06	111.554	13569.7	1.37E+07	188,671	93,167.7	**2.30E−10**
	Mean	98,418.1	55.0237	1364.72	2.48E+06	13,504.7	9233.15	**7.24E−11**
	Std	238,404	25.1139	3342.62	3.38E+06	37,086.6	21,709.0	**6.48E−11**

better than the other algorithm for f_{10},f_{11},f_{12} and f_{13}. It suggests that the principal component analysis plays an important role in improving the search precision, and novel inertia weight is an efficient tool for keeping the exploration capability. In a word, the PCA–GOA has a perfect exploration ability and superior capability in jumping out of local optima. The PCA–GOA fails to achieve the best performance on some functions, and the reasons are as follows: the principal component analysis helps the algorithm finds the better solutions with a fast rate; however, this leads the grasshoppers gather around the best solution and the novel inertia weight mechanism is insufficient to jump out of the local best solution with a certain probability.

Table 9 shows the results from the seven algorithms for the fixed-dimension multimodal benchmark functions. As we can see the results from Table 9, the PCA–GOA obtains very competitive results by comparing with the other algorithms. The PCA–GOA achieves the best results on f_{16},f_{17},f_{20},f_{21},f_{22} and f_{23} in terms of average fitness values. The ALO provides the best average fitness values for f_{15}. However, the p values in Table 12 are bigger than 0.05, which demonstrates

Table 9 Results of the fixed-dimension multimodal benchmark functions

Functions	Result	Algorithm						
		PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_{14}	Best	**0.99800**	**0.99800**	**0.99800**	**0.99800**	**0.99800**	**0.99800**	**0.99800**
	Worst	**3.019081**	21.9884	16.4409	4.95049	20.1535	22.9006	23.8094
	Mean	**1.242847**	10.7078	5.20271	1.85687	10.6552	11.8060	9.79672
	Std	**0.529945**	6.53078	4.51716	1.19142	6.10022	5.92049	5.90027
f_{15}	Best	0.00150	0.00082	**0.00031**	0.00036	0.00058	0.00067	0.00042
	Worst	0.03100	**0.02072**	0.02541	0.02255	0.07687	0.11853	0.06252
	Mean	0.01126	0.00699	**0.00607**	0.00940	0.01379	0.02298	0.00896
	Std	0.00971	**0.00829**	0.00867	0.00878	0.01741	0.02935	0.01327
f_{16}	Best	−1.03117	−1.03159	**−1.03163**	**−1.03163**	**−1.03163**	**−1.03163**	**−1.03163**
	Worst	−0.99039	−1.01257	**−1.03163**	−1.03162	**−1.03163**	−0.21546	**−1.03163**
	Mean	−1.02121	−1.02743	**−1.03163**	**−1.03163**	**−1.03163**	−0.86840	**−1.03163**
	Std	0.01071	0.00408	4.31E−13	1.58E−06	1.13E−12	0.32647	**2.50E−15**
f_{17}	Best	0.39790	0.39791	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**
	Worst	0.40443	0.40666	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**
	Mean	0.39935	0.40010	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**
	Std	0.00144	0.00258	1.26E−13	5.58E−07	2.73E−12	3.49E−08	**9.52E−16**
f_{18}	Best	3.00813	3.01324	**3.00000**	**3.00000**	**3.00000**	**3.00000**	**3.00000**
	Worst	3.82120	3.68405	**3.00000**	3.00004	84.0000	84.0000	84.0000
	Mean	3.13235	3.21874	**3.00000**	**3.00000**	14.7000	14.7000	7.50000
	Std	0.16144	0.16923	**2.65E−12**	6.68E−06	27.6082	24.8276	15.7178
f_{19}	Best	−3.86278	−3.85705	**−3.86278**	**−3.86278**	−3.83819	−3.85825	−3.85209
	Worst	−3.51466	−3.57156	**−3.86278**	−3.84810	−1.11806	−1.81892	−2.51195
	Mean	−3.82797	−3.77399	**−3.86278**	−3.86169	−3.22176	−3.20552	−3.40506
	Std	0.10444	0.06778	**6.14E−10**	0.00293	0.60521	0.55498	0.36121
f_{20}	Best	−3.32200	−2.91423	**−3.32200**	**−3.32199**	**−3.32200**	−3.32200	**−3.32200**
	Worst	−1.22316	−2.26797	−3.17060	−3.02542	**−3.17392**	−3.20059	−3.13770
	Mean	−3.03652	−2.64994	−3.26850	−3.22538	−3.26209	−3.25419	**−3.27061**
	Std	0.53898	0.15028	0.06586	0.10031	0.06430	**0.05930**	0.06776
f_{21}	Best	**−10.1532**	−8.53340	**−10.1532**	**−10.1532**	**−10.1532**	**−10.1532**	**−10.1532**
	Worst	**−2.63047**	−2.12528	**−2.63047**	−2.62871	−2.63047	−2.63047	−2.63047
	Mean	−5.11310	−4.07360	−5.28661	−5.08699	−4.89798	−4.73412	**−5.73063**
	Std	2.49861	**1.83602**	2.88456	2.49568	3.04342	2.88410	3.45632
f_{22}	Best	**−10.4029**	−9.16168	**−10.4029**	**−10.4029**	**−10.4029**	**−10.4029**	**−10.4029**
	Worst	**−1.83759**	−1.59016	**−1.83759**	**−1.83759**	**−1.83759**	**−1.83759**	**−1.83759**
	Mean	-5.53057	−4.20034	−4.72202	−5.50857	−4.74938	−4.96962	**−5.61956**
	Std	3.29602	**2.33436**	2.44516	2.59755	3.16905	3.13466	3.25517
f_{23}	Best	**−10.5364**	−9.00198	**−10.5364**	**−10.5364**	**−10.5364**	**−10.5364**	**−10.5364**
	Worst	**−2.42173**	−1.54064	−2.42173	−1.67649	−1.67655	−1.85948	−1.85948
	Mean	−4.82206	−4.03275	−5.64316	−5.71188	−4.31280	−4.41213	**−5.90321**
	Std	3.00525	**2.21962**	3.51030	3.51850	3.20004	3.40047	3.60952

Table 10 *P* values obtained from the Wilcoxon rank-sum test on unimodal benchmark functions

Function	PSO versus PCA–GOA	BA versus PCA–GOA	ALO versus PCA–GOA	DA versus PCA–GOA	GOA versus PCA–GOA	CGOA versus PCA–GOA
f_1	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
f_2	0.007	3.52E−06	1.97E−05	0.159	1.24E−05	2.35E−06
f_3	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
f_4	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
f_5	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
f_6	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
f_7	1.13E−05	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06

Table 11 P values obtained from the Wilcoxon rank-sum test on multimodal benchmark functions

Function	PSO versus PCA–GOA	BA versus PCA–GOA	ALO versus PCA–GOA	DA versus PCA–GOA	GOA versus PCA–GOA	CGOA versus PCA–GOA
f_8	0.039	0.069	0.531	0.092	1.30E−04	1.13E−05
f_9	1.02E−05	0.106	1.73E−06	1.73E−06	1.73E−06	2.06E−05
f_{10}	2.05E−04	1.25E−04	3.18E−06	3.88E−06	3.18E−06	6.53E−06
f_{11}	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
f_{12}	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.36E−05	2.35E−06
f_{13}	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06

Table 12 P values obtained from the Wilcoxon rank-sum test on the fixed-dimension multimodal benchmark functions

Function	PSO versus PCA–GOA	BA versus PCA–GOA	ALO versus PCA–GOA	DA versus PCA–GOA	GOA versus PCA–GOA	CGOA versus PCA–GOA
f_{14}	3.18E−06	0.393	0.004	8.76E−06	0.516	0.315
f_{15}	0.043	0.926	0.813	0.530	0.453	0.027
f_{16}	1.73E−06	1.73E−06	1.72E−06	0.063	1.73E−06	1.73E−06
f_{17}	1.73E−06	1.73E−06	1.73E−06	0.370	1.73E−06	1.73E−06
f_{18}	0.003	0.003	0.003	0.078	1.89E−04	0.001
f_{19}	1.73E−06	4.45E−05	1.73E−06	1.73E−06	0.185	0.441
f_{20}	0.141	1.73E−06	0.600	0.033	0.644	0.704
f_{21}	0.926	0.106	0.417	0.153	0.405	0.185
f_{22}	0.894	0.066	0.399	0.991	0.156	0.258
f_{23}	0.213	0.001	0.558	0.565	0.086	0.015

Table 13 Comparisons of the runtime on the unimodal benchmark functions

Function	Runtime (s)						
	PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_1	0.11	0.25	7.42	8.98	3.97	3.94	4.86
f_2	0.10	0.25	7.29	8.42	4.28	4.10	5.24
f_3	0.66	0.73	7.75	8.73	4.53	4.56	5.45
f_4	0.11	0.26	7.67	9.16	4.20	4.33	4.97
f_5	0.13	0.28	7.58	8.84	4.33	4.32	5.11
f_6	0.09	0.25	7.47	8.78	4.30	4.24	5.14
f_7	0.13	0.28	7.44	8.74	4.27	4.30	5.21

Table 14 Comparisons of the runtime on the multimodal benchmark functions

Function	Runtime (s)						
	PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_8	0.10	0.27	7.41	8.29	4.30	4.31	5.21
f_9	0.11	0.30	7.49	8.54	4.26	4.27	5.24
f_{10}	0.14	0.32	7.54	8.90	4.36	4.32	5.14
f_{11}	0.12	0.30	7.61	8.72	4.47	4.70	5.62
f_{12}	0.24	0.41	8.03	9.27	4.63	4.66	5.63
f_{13}	0.24	0.42	8.08	9.17	4.44	4.42	5.32

Table 15 Comparisons of the runtime on the fixed-dimension multimodal benchmark functions

Function	Runtime (s)						
	PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_{14}	0.73	0.89	1.69	3.96	4.64	4.67	5.29
f_{15}	0.11	0.30	1.57	3.42	4.17	4.18	4.84
f_{16}	0.09	0.28	1.08	3.42	4.09	4.02	4.68
f_{17}	0.09	0.27	1.07	3.29	3.99	4.00	4.62
f_{18}	0.09	0.28	1.07	3.39	4.07	4.14	4.81
f_{19}	0.16	0.33	1.29	3.29	4.04	4.06	4.70
f_{20}	0.17	0.34	1.96	3.89	4.05	4.06	4.64
f_{21}	0.28	0.39	1.59	3.62	4.06	4.09	4.68
f_{22}	0.32	0.42	1.62	3.66	4.13	4.14	4.76
f_{23}	0.39	0.47	1.68	3.74	4.13	4.22	4.82

the advantage is not obvious. Therefore, it can be claimed that the PCA–GOA shows better performance than other algorithms on fixed-dimension multimodal benchmark functions.

Tables 13, 14 and 15 display the CPU time of the PSO, BA, ALO, DA, GOA, CGOA and PCA–GOA. From Tables 13 and 14, the DA requires the more computation time than the other algorithms on unimodal benchmark functions and multimodal benchmark functions. Table 15 shows that the PCA–GOA demands

Table 16 Comparisons of the allocated memory on the unimodal benchmark functions

Function	Memory usage (Kb)						
	PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_1	5880	7496	22,308	32,976	87,720	72,568	110,752
f_2	4588	7356	44,140	43,928	70,112	68,776	98,768
f_3	15,056	18,044	47,836	51,424	67,632	69,660	88,816
f_4	6280	6840	51,296	53,632	93,284	83,968	119,292
f_5	3512	5084	37,940	35,032	70,068	66,408	96,400
f_6	3788	5680	24,012	44,340	66,616	67,272	82,712
f_7	5300	6408	23,668	34,964	66,336	68,752	102,104

Table 17 Comparisons of the allocated memory on the multimodal benchmark functions

Function	Memory usage (Kb)						
	PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_8	6812	6892	30,804	29,441	94,362	102,864	128,320
f_9	5600	7053	35,488	38,780	87,500	123,126	144,208
f_{10}	6405	8908	42,952	54,964	98,804	116,568	152,620
f_{11}	4876	7549	33,748	40,268	97,936	107,316	132,065
f_{12}	5698	8562	30,436	31,548	114,535	113,083	119,862
f_{13}	5835	8632	28,956	20,480	76,588	77,859	91,629

the highest computation time among the seven algorithms on the fixed-dimension multimodal benchmark functions. However, the CPU time of PCA–GOA is slightly longer than the GOA and CGOA. It indicates that time complexity of the principal component analysis mechanism is acceptable.

Tables 16, 17 and 18 present the allocated memory of the seven algorithms. These tables show that PCA–GOA demands more allocated memory than the other algorithms. However, the allocated memory of PCA–GOA is almost in the same in the level with the GOA and CGOA.

To continue evaluating the search capability and robustness of the proposed PCA–GOA, different number of iterations and population size are used to make further comparisons. The results are shown in Tables 19, 20 and 21. From the tables, with the increasing of number of iterations and population size, all the algorithms provide better results.

Table 19 lists the results of the seven algorithms for unimodal test functions. Compared with other algorithms, the PCA–GOA always achieve better results on all unimodal test functions; meanwhile the superiority of the PCA–GOA is

Table 18 Comparisons of the allocated memory on the fixed-dimension multimodal benchmark functions

Function	Memory usage (Kb)						
	PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_{14}	19,148	22,884	41,432	41,076	51,121	56,566	59,195
f_{15}	6408	8968	43,672	30,536	29,352	34,890	47,789
f_{16}	3916	8152	30,008	28,164	28,392	24,515	44,243
f_{17}	5400	8232	29,768	32,000	28,260	27,288	50,784
f_{18}	3336	8660	24,920	24,896	30,388	54,248	68,811
f_{19}	5984	7632	21,388	23,928	32,516	48,434	52,879
f_{20}	4300	9480	23,304	24,644	30,872	44,670	45,288
f_{21}	8396	11,788	24,396	26,884	43,360	47,404	65,909
f_{22}	10,872	11,912	27,560	25,892	25,952	30,900	43,257
f_{23}	10,196	13,192	29,720	30,996	35,536	36,516	38,920

demonstrated by Table 22. The results show that the PCA–GOA is efficient enough to deal with the unimodal test functions

It can be seen from Tables 20 and 23 the PCA–GOA performs significantly better than the other algorithms on f_{10},f_{11},f_{12} and f_{13}. Table 21 exhibits statistical results for the fixed-dimension multimodal benchmark functions. From the perspective of average fitness values, the PCA–GOA is the first rank on f_{14},f_{16},f_{17},f_{18} and f_{20}. In addition, Table 24 shows that p values of f_{14},f_{16} f_{17} and f_{18} are much below 0.05, which verify the performance of the PCA–GOA is much better than the other algorithms. Mainly, the PCA–GOA performs better than the PSO, BA, ALO, DA, GOA and CGOA on two types multimodal benchmark functions.

Tables 25, 26, 27, 28, 29 and 30, the CPU time and the allocated memory are increased due to the changing of iterate numbers and population size. The conclusions are consistent with the above.

5 Conclusion

In this paper, the GOA is modified by incorporating it with principal component analysis and novel inertia weight. The proposed PCA–GOA is assessed on 23 benchmark functions with different number of iterations and population sizes. The experiments results show that the proposed PCA–GOA is much better than PSO, BA, ALO, DA, basic GOA and CGOA in most test cases. In the PCA–GOA, combing with the two strategies leads the CPU time and allocated memory rise. However, the increasing computation amount is limited.

For some cases, the PCA–GOA falls into local best solutions, and this is the biggest issue requires to be resolved in the future. Furthermore, the PCA–GOA is planned to solve some engineering problems.

Table 19 Results of the unimodal benchmark functions

Functions	Result	Algorithm						
		PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_1	Best	0.21309	0.02583	5.76E−07	405.388	0.06608	1.22671	**3.99E−13**
	Worst	203.657	0.53878	3.02E−05	2415.59	483.001	527.755	**8.68E−11**
	Mean	54.2991	0.14988	8.94E−06	1092.68	106.618	93.5359	**1.11E−11**
	Std	58.0715	0.11796	7.13E−06	444.735	115.877	112.435	**1.73E−11**
f_2	Best	1.97737	13.6744	0.05527	1.45367	0.50270	1.10884	**7.84E−07**
	Worst	62.0617	535.638	125.933	24.1610	712.760	137.472	**1.24E−05**
	Mean	27.4249	117.827	36.2333	13.6662	85.5351	58.1106	**2.29E−06**
	Std	15.7761	83.6296	47.2792	5.95520	127.017	47.4482	**2.27E−06**
f_3	Best	384.546	1.72656	400.425	2560.12	197.353	315.905	**3.77E−11**
	Worst	45548.0	8522.64	2080.09	31,917.2	1810.51	10,641.6	**3.19E−09**
	Mean	16,828.9	1998.53	1161.83	11,004.1	533.559	1455.05	**4.52E−10**
	Std	11,391.3	2295.62	458.115	6929.55	378.420	2027.90	**7.08E−10**
f_4	Best	0.11238	5.40982	2.79544	10.7867	1.80080	1.31416	**5.80E−07**
	Worst	4.89848	26.5353	17.7754	40.0000	10.0082	10.8589	**3.51E−06**
	Mean	1.75054	13.8021	11.0507	24.3342	4.33432	5.00720	**1.66E−06**
	Std	1.17736	4.87076	3.64998	7.22017	1.69257	2.17990	**6.63E−07**
f_5	Best	40.5924	415.523	21.5934	17425.4	30.2311	26.1317	**7.09E−11**
	Worst	94279.2	2461.73	1676.74	738,990	46,348.6	7620.50	**28.7031**
	Mean	12,066.5	888.751	259.503	134,082	3784.22	1588.88	**20.0873**
	Std	26,850.4	490.323	454.051	141,900	8850.00	2213.22	**13.1502**
f_6	Best	6.70496	0.03009	1.28E−06	289.813	0.21175	0.63739	**1.51E−12**
	Worst	166.673	271.561	3.07E−05	3127.03	836.020	1026.11	**2.74E−10**
	Mean	53.1534	9.23151	8.70E−06	1245.81	123.395	152.789	**3.15E−11**
	Std	46.2904	48.7138	5.88E−06	608.794	182.278	235.586	**6.19E−11**
f_7	Best	0.03447	61.2860	0.04027	0.12114	0.06016	0.12844	**0.00007**
	Worst	8.15545	130.805	0.14088	0.96778	0.27978	2.62027	**0.03364**
	Mean	0.73988	99.4434	0.09834	0.34497	0.16635	0.49900	**0.00655**
	Std	1.92179	17.0332	0.02405	0.17594	0.05426	0.48635	**0.00868**

Table 20 Results of the multimodal benchmark functions

Functions	Result	PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_8	Best	−8610.29	−8283.98	**−9805.83**	−7063.12	−8362.62	−8496.78	−8113.62
	Worst	−5485.07	−4875.35	−5417.67	−4092.59	−5731.11	**−5828.87**	−4868.62
	Mean	−6948.11	−7028.53	−5799.46	−5499.38	−7112.65	**−7263.26**	−6215.06
	Std	815.674	661.431	1058.97	635.520	**629.307**	706.873	789.169
f_9	Best	57.3497	208.553	**49.7479**	68.1314	110.545	107.471	114.269
	Worst	248.540	355.320	**143.273**	249.593	287.755	268.726	333.720
	Mean	150.536	271.298	**80.5915**	156.251	178.974	192.217	220.072
	Std	46.3310	34.4496	**20.3012**	37.9877	40.7262	41.3614	49.5172
f_{10}	Best	0.13149	2.77894	1.34042	3.00933	0.93868	2.62095	**2.15E−07**
	Worst	14.7262	19.8364	3.78562	11.8128	19.6302	19.8435	**3.19E−06**
	Mean	3.49438	8.18858	2.27194	8.63044	8.28998	9.17465	**9.49E−07**
	Std	3.21745	6.87101	0.62751	1.83046	6.84657	6.91831	**6.60E−07**
f_{11}	Best	0.44963	100.001	0.00010	2.82206	0.05124	0.46660	**7.11E−13**
	Worst	3.00872	320.790	0.05754	27.0634	4.60759	8.97092	**3.02E−11**
	Mean	1.45743	201.190	0.01328	11.4754	1.95515	1.98355	**8.45E−12**
	Std	0.63252	48.1193	0.01252	6.58211	1.06081	2.02854	**7.53E−12**
f_{12}	Best	0.68150	13.9170	3.62897	5.71001	1.22570	0.29290	**2.55E−15**
	Worst	3.76674	40.2508	17.8881	29872.2	5.58527	9.15811	**2.49E−13**
	Mean	1.72744	24.7966	9.34067	1026.15	2.98264	3.45463	**5.06E−14**
	Std	0.77104	6.01853	3.15490	5356.60	1.16218	1.76212	**5.06E−14**
f_{13}	Best	4.13490	0.25575	0.00001	52.1945	0.07992	0.12519	**1.05E−13**
	Worst	16.8400	0.61280	13.0282	553861	32.9731	31.9114	**2.40E−12**
	Mean	6.14172	0.39239	0.57375	74209.1	4.52317	5.43694	**8.78E−13**
	Std	2.73566	0.09009	2.35404	120,116	6.20642	7.47445	**6.63E−13**

Table 21 Results of the fixed-dimension multimodal benchmark functions

Functions	Result	Algorithm						
		PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_{14}	Best	**0.99800**	**0.99800**	**0.99800**	**0.99800**	**0.99800**	**0.99800**	**0.99800**
	Worst	0.99815	22.9006	2.98211	**0.99800**	**0.99800**	**0.99800**	**0.99800**
	Mean	0.99801	9.10887	1.36222	**0.99800**	**0.99800**	**0.99800**	**0.99800**
	Std	0.00003	6.45325	0.60029	5.91E−10	4.64E−16	1.24E−12	**3.33E−16**
f_{15}	Best	0.00133	0.00060	0.00050	0.00039	0.00052	0.00066	**0.00031**
	Worst	0.02255	0.02037	**0.02036**	**0.02036**	0.07747	0.10778	0.02166
	Mean	0.00739	0.00561	**0.00212**	0.00480	0.01238	0.00710	0.00627
	Std	0.00795	0.00816	**0.00488**	0.00716	0.01556	0.01978	0.00878
f_{16}	Best	− 1.03158	− 1.03161	**− 1.03163**	**− 1.03163**	**− 1.03163**	**− 1.03163**	**− 1.03163**
	Worst	− 1.02640	− 1.03020	**− 1.03163**	− 1.03162	**− 1.03163**	**− 1.03163**	**− 1.03163**
	Mean	− 1.03027	− 1.03118	**− 1.03163**	**− 1.03163**	**− 1.03163**	**− 1.03163**	**− 1.03163**
	Std	0.00114	0.00040	4.25E−14	1.89E−06	1.70E−14	1.66E−08	**0**
f_{17}	Best	**0.39789**	0.39791	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**
	Worst	0.40016	0.39930	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**
	Mean	0.39833	0.39827	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**
	Std	0.00048	0.00038	7.65E−15	4.67E−07	2.47E−14	4.89E−09	**1.11E−16**
f_{18}	Best	3.00033	3.00132	**3.00000**	**3.00000**	**3.00000**	**3.00000**	**3.00000**
	Worst	3.11706	3.20848	**3.00000**	3.00018	84.0000	84.0000	**3.00000**
	Mean	3.02447	3.04631	**3.00000**	3.00001	8.40000	5.70000	**3.00000**
	Std	0.02650	0.04725	2.65E−12	0.00003	20.2050	14.5400	**1.14E−14**
f_{19}	Best	**− 3.86278**	− 3.85824	**− 3.86278**	**− 3.86278**	− 3.86278	− 3.86263	**− 3.86278**
	Worst	**− 3.86278**	− 3.76405	**− 3.86278**	− 3.86186	− 2.43776	− 2.74575	− 3.28329
	Mean	**− 3.86278**	− 3.83391	**− 3.86278**	− 3.86270	− 3.41151	− 3.61797	− 3.75482
	Std	**2.66E−15**	0.02232	2.45E−14	0.00020	0.40574	0.30596	0.11268
f_{20}	Best	**− 3.32200**	− 3.05784	**− 3.32200**	**− 3.32200**	**− 3.32200**	**− 3.32200**	**− 3.32200**
	Worst	− 1.70606	− 2.63925	**− 3.20307**	− 3.07987	− 3.19243	− 3.19122	− 3.20203
	Mean	− 3.13002	− 2.81651	− 3.26651	− 3.27100	− 3.26880	− 3.28115	**− 3.28224**
	Std	0.40197	0.10239	0.05932	0.07351	0.06087	0.05780	**0.05622**
f_{21}	Best	**− 10.1532**	− 9.65726	**− 10.1532**	**− 10.1532**	**− 10.1532**	**− 10.1532**	**− 10.1532**
	Worst	**− 2.63047**	− 2.38874	**− 2.63047**	**− 2.63047**	**− 2.63047**	**− 2.63047**	**− 2.63047**
	Mean	− 5.71467	− 4.83449	− 7.12753	**− 7.86558**	− 5.63605	− 5.65504	− 6.05127
	Std	3.07974	**2.11232**	3.13865	2.63198	3.12298	3.49476	3.26626
f_{22}	Best	**− 10.4029**	− 9.85836	**− 10.4029**	**− 10.4029**	**− 10.4029**	**− 10.4029**	**− 10.4029**
	Worst	− 2.75193	− 1.72241	**− 2.76590**	− 1.83660	− 1.83759	− 1.83759	− 1.83759
	Mean	− 6.14797	− 5.35734	− 7.72039	**− 8.62315**	− 6.03851	− 6.72014	− 6.15327
	Std	3.32395	**2.81672**	3.12212	2.98015	3.62025	3.55225	3.55723
f_{23}	Best	**− 10.5364**	− 9.64671	**− 10.5364**	**− 10.5364**	**− 10.5364**	**− 10.5364**	**− 10.5364**
	Worst	− 1.85948	− 1.63682	− 1.85948	**− 2.42160**	− 1.67655	− 1.67655	− 1.67655
	Mean	− 5.46098	− 4.64417	− 5.67463	**− 8.28955**	− 5.63423	− 5.11482	− 6.15239
	Std	3.45541	**2.75573**	3.32986	3.00765	3.80109	3.41296	3.89297

Table 22 P values obtained from the Wilcoxon rank-sum test on unimodal benchmark functions

Function	PSO versus PCA–GOA	BA versus PCA–GOA	ALO versus PCA–GOA	DA versus PCA–GOA	GOA versus PCA–GOA	CGOA versus PCA–GOA
f_1	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
f_2	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
f_3	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
f_4	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
f_5	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
f_6	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
f_7	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06

Table 23 P values obtained from the Wilcoxon rank-sum test on multimodal benchmark functions

Function	PSO versus PCA–GOA	BA versus PCA–GOA	ALO versus PCA–GOA	DA versus PCA–GOA	GOA versus PCA–GOA	CGOA versus PCA–GOA
f_8	0.006	4.90E−04	0.005	0.003	1.15E−04	1.25E−04
f_9	6.32E−05	3.88E−04	1.73E−06	1.64E−05	0.007	0.045
f_{10}	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
f_{11}	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
f_{12}	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06
f_{13}	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06	1.73E−06

Table 24 P values obtained from the Wilcoxon rank-sum test on the fixed-dimension multimodal benchmark functions

Function	PSO versus PCA–GOA	BA versus PCA–GOA	ALO versus PCA–GOA	DA versus PCA–GOA	GOA versus PCA–GOA	CGOA versus PCA–GOA
f_{14}	1.73E−06	1.73E−06	0.005	1.23E−05	0.025	1.73E−06
f_{15}	0.098	0.614	0.026	0.750	0.178	0.428
f_{16}	1.73E−06	1.73E−06	4.70E−06	4.01E−05	4.74E−06	1.73E−06
f_{17}	1.73E−06	1.73E−06	7.92E−06	4.01E−05	3.66E−06	1.73E−06
f_{18}	1.73E−06	1.73E−06	1.73E−06	8.67E−05	6.15E−06	1.73E−06
f_{19}	1.73E−06	0.001	1.73E−06	1.92E−06	3.32E−04	0.027
f_{20}	0.036	1.73E−06	0.600	0.041	0.003	0.136
f_{21}	0.808	0.141	0.614	0.147	0.465	0.478
f_{22}	0.497	0.299	0.086	0.079	0.829	0.894
f_{23}	0.637	0.360	0.478	0.020	0.910	0.797

Table 25 Comparisons of the runtime on the unimodal benchmark functions

Function	Runtime (s)						
	PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_1	0.57	1.38	57.77	55.27	31.64	32.06	35.65
f_2	0.61	1.40	60.49	55.56	31.35	32.03	36.08
f_3	3.09	3.83	62.44	57.65	33.24	35.04	37.90
f_4	0.67	1.43	60.60	55.21	31.50	32.53	35.89
f_5	0.74	1.50	60.69	57.11	31.51	32.40	35.41
f_6	0.57	1.35	60.58	56.78	31.34	32.02	36.02
f_7	0.75	1.52	60.01	57.18	31.38	31.00	36.83

Table 26 Comparisons of the runtime on the multimodal benchmark functions

Function	Runtime (s)						
	PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_8	0.63	1.39	61.62	55.41	31.20	32.72	35.57
f_9	0.70	1.51	58.73	56.17	31.26	32.96	34.91
f_{10}	0.80	1.59	59.64	57.44	31.56	32.49	34.95
f_{11}	0.74	1.49	58.17	56.77	31.56	32.78	35.10
f_{12}	1.27	2.03	59.78	58.46	32.20	33.20	36.30
f_{13}	1.28	2.02	59.56	57.73	32.21	33.26	36.38

Table 27 Comparisons of the runtime on the fixed-dimension multimodal benchmark functions

Function	Runtime (s)						
	PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_{14}	3.39	4.08	10.08	23.17	30.46	33.98	36.20
f_{15}	0.66	1.43	11.18	20.98	30.90	30.69	31.17
f_{16}	0.56	1.33	9.60	21.29	27.98	29.37	29.93
f_{17}	0.54	1.31	9.47	20.95	28.08	29.38	30.03
f_{18}	0.56	1.32	9.48	21.09	28.15	29.41	31.19
f_{19}	0.93	1.67	10.78	21.99	31.38	30.97	33.51
f_{20}	1.01	1.72	15.15	24.65	31.48	30.79	35.03
f_{21}	1.22	1.97	11.46	22.93	31.28	31.11	33.40
f_{22}	1.39	2.12	11.83	23.41	31.77	31.46	33.90
f_{23}	1.66	2.39	12.65	23.77	31.73	31.41	34.52

Table 28 Comparisons of the allocated memory on the unimodal benchmark functions

Function	Memory usage (Kb)						
	PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_1	13,380	14,748	31,080	43,044	242,692	241,512	254,692
f_2	17,104	28,396	57,036	64,900	240,724	243,280	245,168
f_3	26,140	33,932	54,152	79,524	240,612	233,640	245,076
f_4	9480	12,804	66,784	70,048	242,988	235,980	264,092
f_5	6012	12,532	55,804	55,416	236,688	230,896	239,996
f_6	6560	10,844	27,908	57,136	248,404	239,700	264,172
f_7	6716	12,036	49,664	44,200	232,292	219,176	240,216

Table 29 Comparisons of the allocated memory on the multimodal benchmark functions

Function	Memory usage (Kb)						
	PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_8	11,780	14,764	52,416	35,148	272,692	243,556	279,596
f_9	18,768	14,220	40,952	43,344	234,656	232,984	255,340
f_{10}	6680	73,052	45,288	68,737	232,964	211,020	256,888
f_{11}	4908	62,080	56,864	44,512	235,824	221,036	245,436
f_{12}	30,740	51,756	32,816	34,912	236,956	230,976	242,620
f_{13}	10,632	35,876	36,592	24,520	231,308	212,776	251,096

Table 30 Comparisons of the allocated memory on the fixed-dimension multimodal benchmark functions

Function	Memory usage (Kb)						
	PSO	BA	ALO	DA	GOA	CGOA	PCA–GOA
f_{14}	48,628	49,540	49,172	56,404	70,014	71,071	76,228
f_{15}	17,780	50,564	48,560	31,128	44,668	45,468	59,324
f_{16}	15,264	37,504	42,848	33,076	55,240	52,144	64,076
f_{17}	19,348	27,700	34,776	39,228	75,480	74,036	80,360
f_{18}	13,124	35,976	23,080	30,904	69,084	66,208	79,404
f_{19}	20,460	34,452	23,904	30,148	92,824	90,548	101,320
f_{20}	12,064	13,952	27,640	28,740	60,416	63,236	65,624
f_{21}	23,212	18,392	31,280	32,300	68,964	62,396	89,816
f_{22}	37,436	53,500	31,780	28,112	55,840	57,772	62,360
f_{23}	43,036	19,820	34,904	36,844	42,560	41,012	47,360

Acknowledgements This paper is supported by Jilin Province Science and Technology Department Foundation of China under Grant No. 2017-00005000605, Research on Visual Inspection System of Industrial Robot for Car Stamping Parts.

References

1. Shehab M, Khader AT, Laouchedi M et al (2018) Hybridizing cuckoo search algorithm with bat algorithm for global numerical optimization. J Supercomput 1–28
2. Nouiri M, Bekrar A, Jemai A et al (2018) An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. J Intell Manuf 29(3):603–615
3. Chen X, Xu B, Mei C et al (2018) Teaching–learning-based artificial bee colony for solar photovoltaic parameter estimation. Appl Energy 212:1578–1588
4. Payne A, Avendaño-Franco G, Bousquet E et al (2018) Firefly algorithm applied to noncollinear magnetic phase materials prediction. J Chem Theory Comput 14(8):4455–4466
5. Kaveh A, Zakian P (2018) Improved GWO algorithm for optimal design of truss structures. Eng Comput 34(4):685–707
6. Sun Y et al (2018) A modified whale optimization algorithm for large-scale global optimization problems. Expert Syst Appl 114:563–577
7. Holland John H (1992) Genetic algorithms. Sci Am 267(1):66–73
8. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995. MHS'95. IEEE, pp 39–43
9. Dorigo M, Di Caro G (1999) Ant colony optimization: a new meta-heuristic. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). IEEE, vol 2, pp 1470–1477
10. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department
11. Yang XS (2009) Firefly algorithms for multimodal optimization. In: International Symposium on Stochastic Algorithms. Springer, Berlin, pp 169–178
12. Yang XS, Deb S (2009) Cuckoo search via Lévy flights. In: World Congress on Nature and Biologically Inspired Computing, 2009. NaBIC 2009. IEEE, pp 210–214
13. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248
14. Yang XS (2010) A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Springer, Berlin, pp 65–74
15. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
16. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl Based Syst 89:228–249
17. Riganati John P, Schneck Paul B (1984) Supercomputing. Computer 10:97–113
18. Garcia C et al (2013) Multi-GPU based on multicriteria optimization for motion estimation system. EURASIP J Adv Signal Process 2013.1, p 23
19. Wei K-C, Wu C, Wu C-J (2013) Using CUDA GPU to accelerate the ant colony optimization algorithm. In: 2013 International Conference on Parallel and Distributed Computing, Applications and Technologies. IEEE
20. Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimisation algorithm: theory and application. Adv Eng Softw 105:30–47
21. Aljarah I, Ala'M AZ, Faris H et al (2018) Simultaneous feature selection and support vector machine optimization using the grasshopper optimization algorithm. Cognitive Comput, pp 1–18
22. Zhang X, Miao Q, Zhang H et al (2018) A parameter-adaptive VMD method based on grasshopper optimization algorithm to analyze vibration signals from rotating machinery. Mech Syst Signal Process 108:58–72
23. Wu J, Wang H, Li N et al (2017) Distributed trajectory optimization for multiple solar-powered UAVs target tracking in urban environment by Adaptive Grasshopper Optimization Algorithm. Aerosp Sci Technol 70:497–510

24. Hekimoğlu B, Ekinci S (2018) Grasshopper optimization algorithm for automatic voltage regulator system. In: 2018 5th International Conference on Electrical and Electronic Engineering (ICEEE). IEEE, pp 152–156
25. Łukasik S, Kowalski PA, Charytanowicz M et al (2017) Data clustering with grasshopper optimization algorithm. In: 2017 Federated Conference on Computer Science and Information Systems (FedCSIS). IEEE, pp 71–74
26. Lal DK, Barisal AK, Tripathy M (2018) Load frequency control of multi area interconnected microgrid power system using grasshopper optimization algorithm optimized fuzzy PID controller. In: 2018 Recent Advances on Engineering, Technology and Computational Sciences (RAETCS). IEEE, pp 1–6
27. Fathy A (2018) Recent meta-heuristic grasshopper optimization algorithm for optimal reconfiguration of partially shaded PV array. Sol Energy 171:638–651
28. Arora S, Anand P (2018) Chaotic grasshopper optimization algorithm for global optimization. Neural Comput Appl, pp 1–21
29. Ewees AA, Elaziz MA, Houssein EH (2018) Improved grasshopper optimization algorithm using opposition-based learning. Expert Syst Appl 112:156–172
30. Saxena A, Shekhawat S, Kumar R (2018) Application and development of enhanced chaotic grasshopper optimization algorithms. Model Exp Eng 2018:4945157
31. Liang H, Jia H, Xing Z et al (2019) Modified Grasshopper algorithm based multilevel thresholding for color image segmentation. IEEE Access 7:11258–11295
32. Luo J, Chen H, Xu Y et al (2018) An improved grasshopper optimization algorithm with application to financial stress prediction. Appl Math Model 64:654–668
33. Johnson RA, Wichern DW (2005) Applied multivariate statistical analysis, 6/E. Technometrics 47(4):517–517
34. Zhao X, Lin W, Zhang Q (2014) Enhanced particle swarm optimization based on principal component analysis and line search. Appl Math Comput 229:440–456
35. Cui Z, Li F, Zhang W (2018) Bat algorithm with principal component analysis. Int J Mach Learn Cybern 10(3):603–622
36. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102
37. Digalakis JG, Margaritis KG (2001) On benchmarking functions for genetic algorithms. Int J Comput Math 77(4):481–506
38. Molga M, Smutnicki C (2005) Test functions for optimization needs. http://www.robermarks.org/Classes/ENGR5358/Papers/functions.pdf
39. Yang XS (2010) Firefly algorithm, stochastic test functions and design optimisation. arXiv preprint arXiv:1003.1409
40. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98
41. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 27(4):1053–1073
42. Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull 1(6):80–83
43. García S, Molina D, Lozano M et al (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. J Heuristics 15(6):617