



ARTICLE

Elite Population-Driven Backtracking Search Algorithm

Yiying Zhang*

College of Electrical Engineering and Automation, Shandong University of Science and Technology, Qingdao, China

*Corresponding Author: Yiying Zhang. Email: zhangyiying@sdust.edu.cn

Received: 14 June 2026; Accepted: 19 August 2026; Published: 22 October 2026

ABSTRACT: Backtracking search algorithm (BSA) is a very simple and efficient metaheuristic algorithm. However, BSA only relies on the random crossover vectors generated between the historical population and the current population to guide the search direction of the population, lacking purposefulness. In view of this, this paper proposes an improved backtracking search algorithm (IBSA) that designs three search strategies by creating two dynamic elite populations. Compared to BSA, IBSA has stronger purposefulness in the search process. To validate the effectiveness of the proposed algorithm, 30 challenging test functions and four complex constrained engineering design problems are solved by employing IBSA, BSA, and four other excellent metaheuristic algorithms. Experimental evidence attests to the superior effectiveness of IBSA in optimization tasks, which also proves the effectiveness of the improved strategy.

KEYWORDS: Elite population; backtracking search algorithm; global optimization; engineering optimization

1 Introduction

Metaheuristic algorithms are a class of optimization methods inspired by natural phenomena, characterized by simple structures, clear rules, and high computational efficiency. Currently, a large number of metaheuristic algorithms inspired by various natural phenomena have emerged. Kepler optimization algorithm is inspired by the Kepler's laws of planetary motion to predict the position and velocity of planets at any given time [1]. Cuckoo search is inspired by the obligate brood parasitic behaviour of some cuckoo species in combination with the Lévy flight behaviour of some birds and fruit flies [2]. Whale optimization algorithm is inspired by the social behavior of humpback whales and the bubble-net hunting strategy [3]. Teaching-learning-based optimization is inspired by the traditional classroom teaching model [4]. Group teaching optimization algorithm is inspired by the group teaching mode based on individualized instruction [5]. Bounty hunter optimizer is inspired by the behavior of bounty hunters [6]. Neural network algorithm is inspired by the biological nervous systems and artificial neural networks [7]. Biogeography-based optimization is inspired by the mathematics of biogeography [8]. Black hole algorithm is inspired by the black hole phenomenon [9]. Exchange market algorithm is inspired by the procedure of trading the shares on stock market [10]. Gravitational search algorithm is inspired by the law of gravity and mass interactions [11]. Honey badger algorithm is inspired by the intelligent foraging behavior of honey badger [12]. Gaining-sharing knowledge based algorithm is inspired by the process of gaining and sharing knowledge during the human life span [13]. Differential evolution is inspired by the evolutionary process [14]. In addition, metaheuristic algorithms have also been successfully applied to different types of engineering optimization problems. In particular, evolutionary algorithms have been very successful in applications to complex engineering problems [15–17].

Backtracking search algorithm (BSA) is a very classical metaheuristic algorithm, which simulates the return of a social group of living creatures at random intervals to hunting areas that are previously found fruitful for obtaining nourishment [18]. With its very simple structure, BSA guides the search direction based on the vector difference between the historical population and the current one, and enhances the algorithm's global search capability through two mechanisms: first, using random numbers drawn from a standard normal distribution to adjust the magnitude of the vector difference between historical and current populations; second, applying a binary matrix to perform crossover on the vector difference. BSA involves no complex calculations and requires only a few parameters to tune, making it highly convenient for application to various types of optimization problems. Guney et al. [19] tackled the synthesis problem of concentric circular antenna arrays With BSA. Jin and Yin [20] designed an improved BSA to identify the soil parameters. Zou et al. [21] employed the multi-objective discrete BSA with decomposition to solve the community detection problem in complex networks. Guney and Durmus [22] adopted BSA to address the nulling problem of antenna patterns for linear arrays. Shaheen et al. [23] presented an improved BSA to solve reactive power dispatch problem. Ahandani et al. [24] proposed a shuffled BSA to extract the unknown parameter of chaotic systems. Wang et al. [25] reported three variants of BSA to optimize the echo state networks. Wang et al. [26] put forward an advanced BSA to solve a new joint replenishment problem under trade credit with grouping constraint. Hu et al. [27] focused on solving The dynamic economic dispatch problem that accounts for valve-point effects by an adaptive BSA. Although BSA has been successfully applied in some optimization problems, it tends to get trapped in local optima when tackling complex optimization problems. This is mainly because BSA only has a search rule guided by the cross vector between the current population and the historical population, which has blindness in the search direction. Therefore, once the obtained optimal solution is a local optimal solution, the entire population may miss the global optimal solution. Motivated by this, this paper proposes an improved backtracking search algorithm (IBSA). The main contribution of this paper includes:

- This paper proposes an improved version of BSA, i.e., IBSA.
- IBSA is designed based on the defined two dynamic elite populations, i.e., local elite population and global elite population.
- IBSA is used to solve the challenging Congress on Evolutionary Computation (CEC) 2014 test suite with 30 functions.
- The performance of IBSA is investigated by the four complexly constrained engineering structural design problems.

The rest of this paper is organized as: [Section 2](#) introduces the basic BSA. [Section 3](#) presents the proposed IBSA. [Section 4](#) discusses the experimental results of solving the numerical problems. [Section 5](#) shows the experimental results of solving the engineering design problems. A summary of this paper and a look ahead to future research are given in [Section 6](#).

2 Backtracking Search Algorithm (BSA)

BSA has a very simple framework, whose framework primarily includes the following four parts, i.e., the initialization stage, followed by the operations in Section I and Section II, along with population generation [18]:

- Initialization. This stage is to initialize the population X ($X = \{x_1, x_2, \dots, x_N\}$), where N is the population size. Like other population-based metaheuristic algorithms, the population of BSA is also randomly initialized, and can be denoted by:

$$x_i = l_d + \lambda_1(u_d - l_d), i = 1, 2, \dots, N, \quad (1)$$

where $\mathbf{x}_i$ is the position of individual i , λ_1 is a random number between 0 and 1, $\mathbf{l}_d$ is the lower boundary of the decision variables, and $\mathbf{u}_d$ is the upper limit of the decision variables.

- Section-I. The purpose of this stage is to identify the historical population $\mathbf{X}_{old}$, which is the foundation of the search rule. The implementation of this stage relies on the following expression:

$$\mathbf{X}_{old} = \mathbf{X}, \text{ if } \lambda_2 \leq \lambda_3, \quad \mathbf{X}_{old} = K(\mathbf{X}_{old}), \text{ if } \lambda_2 > \lambda_3, \quad (2)$$

where λ_2 and λ_3 are two random numbers between 0 and 1, and $K(\cdot)$ is a function that reorders all individuals within $\mathbf{X}_{old}$.

- Section-II. The purpose of this stage is to create a trial population $\mathbf{X}_p$ ($\mathbf{X}_p = \{\mathbf{x}_{p,1}, \mathbf{x}_{p,2}, \dots, \mathbf{x}_{p,N}\}$). This step can be represented by:

$$\mathbf{x}_{p,i} = \mathbf{x}_i + 3\lambda_4 \mathbf{m}_i(\mathbf{x}_{old,i} - \mathbf{x}_i), \quad i = 1, 2, \dots, N, \quad (3)$$

where λ_4 is a random number obeying the standard normal distribution and $\mathbf{m}_i$ is a crossover vector controlled by the mix rate. The method of producing $\mathbf{m}_i$ can be found in [18].

- Population generation. This stage is to get the next generation population, which can be computed by:

$$\mathbf{x}_i = \mathbf{x}_{p,i}, \text{ if } f(\mathbf{x}_{p,i}) \leq f(\mathbf{x}_i). \quad (4)$$

Algorithm 1 presents the implementation of BSA. The source code of BSA can be accessed through the following link: <https://ww2.mathworks.cn/matlabcentral/profile/authors/2914830>.

Algorithm 1: The execution process of BSA

Require: The lower limit of variables $\mathbf{l}_d$, The upper limit of variables $\mathbf{u}_d$, the population size N , the dimension of the problem D , the maximum iteration count T_m

- 1: Define the initial population as well as the historical one by Eq. (1);
 - 2: Evaluate each individual and identify the optimal solution $\mathbf{x}_o$;
 - 3: **for** each $t \in [1, T_m]$ **do**
 - 4: Generate the crossover vectors $\mathbf{m}_i$;
 - 5: Get the historical population by Eq. (2);
 - 6: **for** each $i \in [1, N]$ **do**
 - 7: Compute the trail individual by Eq. (3);
 - 8: Update $\mathbf{x}_i$ by Eq. (4);
 - 9: **end for**
 - 10: Evaluate each individual and find the optimal solution $\mathbf{x}_o$;
 - 11: **end for**
 - 12: Output $\mathbf{x}_o$.
-

3 Improved Backtracking Search Algorithm (IBSA)

It has been widely documented in the literature that elite individuals or populations are employed to steer the search direction of a population to enhance the global optimization performance of population-based metaheuristic algorithms. Our previous work [28] proposed an elite archives-driven particle swarm optimization, where three elite archives are created to design multiple search strategies. Zhao and Wang [29] reported an elite-ordinary synergistic particle swarm optimization, where particles are divided into elite particles and ordinary particles, and two kinds of learning strategies are designed for elite and ordinary

particles. Yang et al. [30] presented a random neighbor elite guided differential evolution, where a random neighbor elite guided mutation strategy is defined to mutate individuals. Wang and Huang [31] designed an elite opposition-based Jaya algorithm, where the elite opposition learning strategy is used to design the search phase. Deng et al. [32] shown an elite neighborhood search and dynamic feature probability-guided crow search algorithm, where the elite neighborhood search strategy is employed to guide the local search of crows. Motivated by this, two dynamic elite populations are established in this study to lead the search direction and strengthen the global exploration ability of BSA.

3.1 Dynamic Elite Population

The idea of the dynamic elite population designed in this paper originates from our previous work [28]. In [28], we define two elite populations, which can be called local elite population and global elite population.

- Local elite population. The local elite population X_L with N individuals, which can be expressed by:

$$X_L = \{x_{L,1}, x_{L,2}, \dots, x_{L,N}\}. \quad (5)$$

The update mechanism of X_L can be described as follows: if individual i produces a trail individual $x_{p,i}$ that is superior to itself in one iteration, then that $x_{p,i}$ will be added to X_L . Furthermore, as shown in Algorithm 2, if the number of individuals in X_L is less than N , $x_{p,i}$ will be directly added to X_L , otherwise $x_{p,i}$ will randomly replace one individual in X_L .

Algorithm 2: The approach for refreshing the local elite population

Require: The population size N , the local elite population X_L , the trail individual $x_{p,i}$, the number of individuals in the local elite population v

- 1: **if** $v < N$ **then**
 - 2: Add $x_{p,i}$ into X_L ;
 - 3: **else**
 - 4: Randomly replace an individual in X_L with $x_{p,i}$;
 - 5: **end if**
 - 6: Output X_L .
-

- Global elite population. The global elite population X_G with N individuals, and can be expressed by:

$$X_G = \{x_{G,1}, x_{G,2}, \dots, x_{G,N}\}. \quad (6)$$

The update mechanism of X_G can be described as follows: if the best solution obtained by the entire population in one iteration is better than the best solution obtained in the previous iteration, the global elite population will receive the current best solution. Furthermore, as shown in Algorithm 3, if the number of individuals in X_G is less than N , the current best solution will be directly added to X_G , otherwise the current best solution will randomly replace one individual in X_G .

Algorithm 3: The approach for refreshing the global elite population

Require: The population size N , the global elite population X_G , the current best solution x_{cb} , the number of individuals in the global elite population z

- 1: **if** $z < N$ **then**
- 2: Add x_{cb} into X_G ;
- 3: **else**
- 4: Randomly replace an individual in X_G with x_{cb} ;
- 5: **end if**
- 6: Output X_G .

3.2 The Designed Search Strategies

As mentioned earlier, the defined elite population is designed to store the individuals that have the potential to achieve the global optimal solution. It is more likely to locate the global optimum when the search direction is steered by individuals drawn from the elite population. Therefore, the search strategy based on the elite population designed in this paper can be specified in the following way:

- The first search strategy revolves around the local elite population X_L , which can be expressed by:

$$x_{p,i} = x_i + \lambda_5 m_i (x_{L,a} - x_i), i = 1, 2, \dots, N, \quad (7)$$

where λ_5 is a matrix with 1 row and D columns, where each element is a random number between 0 and 1. a is a randomly selected integer between 1 and v . From Eq. (7), individual i is guided by the individual a in the local elite population. In addition, m_i is introduced into Eq. (7) can have a certain cross effect, which is beneficial for enriching population diversity.

- The second search strategy is built around the global elite population X_G , which can be expressed by:

$$x_{p,i} = x_i + \lambda_6 m_i (x_{G,b} - x_i), i = 1, 2, \dots, N, \quad (8)$$

where λ_6 is a matrix with 1 row and D columns, where each element is a random number between 0 and 1. b is a randomly selected integer between 1 and z . From Eq. (8), individual i is guided by the individual b in the global elite population. In addition, like Eq. (7), m_i is introduced into Eq. (8) can have a certain cross effect, which is beneficial for enriching population diversity.

3.3 The Implementation of IBSA

According to the designed two search strategies, the built search mechanism can be expressed as follows:

$$x_{p,i} = \begin{cases} x_i + 3\lambda_4 m_i (x_{old,i} - x_i), & \text{if } \beta < 1/3, \\ x_i + \lambda_5 m_i (x_{L,a} - x_i), & \text{if } \beta < 2/3, \\ x_i + \lambda_6 m_i (x_{G,b} - x_i), & \text{otherwise,} \end{cases} \quad (9)$$

where β is called the selection factor, which is a random number between 0 and 1. As shown in Eq. (9), the built search mechanism is based on Eqs. (3), (7) and (8). The three search strategies in Eq. (9) have the same contribution to the constructed search mechanism, and therefore are assigned the same probability of being executed. Furthermore, each of the three search strategies is devised from a unique standpoint, namely the original search strategy of BSA, the search strategy guided by the local elite population, and the search strategy guided by the global elite population. Therefore, in the absence of prior knowledge, allocating the same search probability to the three different search strategies is very natural. Based on the improvement strategies for the design, the implementation of IBSA has been demonstrated in Algorithm 4.

Algorithm 4: The detailed implementation of IBSA

Require: The lower boundary of the decision variables $\mathbf{l}_d$, the upper boundary of the decision variables $\mathbf{u}_d$, the population size N , the dimension of the problem D , the maximum iteration count T_m

- 1: Define the initial population as well as the historical one by Eq. (1);
- 2: Evaluate each individual and find the optimal solution $\mathbf{x}_o$;
- 3: Initialize the local elite population $\mathbf{X}_L$ and global elite population $\mathbf{X}_G$ by $\mathbf{X}_L = \mathbf{x}_o$ and $\mathbf{X}_G = \mathbf{x}_o$;
- 4: **for** each $t \in [1, T_m]$ **do**
- 5: Carry out the generation of the crossover vectors $\mathbf{m}_i$;
- 6: Acquire the historical population by Eq. (2);
- 7: **for** each $i \in [1, N]$ **do**
- 8: Compute the trail individual by Eq. (9);
- 9: Update $\mathbf{x}_i$ by Eq. (4) and update $\mathbf{X}_L$ by Algorithm 2;
- 10: **end for**
- 11: Evaluate each individual and find the optimal solution $\mathbf{x}_o$;
- 12: Update $\mathbf{X}_G$ by Algorithm 3;
- 13: **end for**
- 14: Output $\mathbf{x}_o$.

From Algorithm 4, the advantages of IBSA over BSA lie in the following aspects:

- (1) Compared to BSA's single search strategy, IBSA employs three search strategies, providing the population with more directional choices during the search process. This helps enrich population diversity and enhances the algorithm's ability to escape local optima.
- (2) Compared with BSA, the two new search strategies designed by IBSA are both based on elite populations. That is to say, the direction of the population during the search process is guided by the elite solutions, which is beneficial for the convergence of the entire algorithm and the finding of higher-quality solutions.
- (3) Compared with BSA, IBSA introduces two new search strategies, but it does not significantly increase the computational cost. Compared with BSA, the main computational overhead added by IBSA is concentrated on the dynamic updates of the local elite population and the global elite population. However, as shown in Algorithms 2 and 3, the dynamic update methods for the local elite population and the global elite population in the design do not involve complex calculations but rather very simple computational structures. From Algorithm 4, the specific computational complexity analysis of IBSA can be summarized by: line 1 is to initialize the population and historical population, and the corresponding computational complexity is $O(N \times D)$; line 2 is to evaluate the each individual of the population and find the best individual, and the corresponding computational complexity is $O(N \times D)$; line 3 is to initialize two elite populations, and the corresponding computational complexity is $O(D)$; line 5 is to obtain the crossover matrix, and the worst computational complexity is $O(N \times D \times T_m)$ according to the method in [18]; line 6 is to get the historical population, and the corresponding computational complexity is $O(N \times D \times T_m)$; line 7 to line 10 are to get next generation individuals, and the corresponding computational complexity is $O(N \times D \times T_m)$; line 11 is to obtain the best individual, and the corresponding computational complexity is $O(N \times T_m)$; line 12 is to update the global elite population, and the corresponding computational complexity is $O(D \times T_m)$. Therefore, the overall computational complexity can be expressed as $O(N \times D \times T_m)$.

The notable feature of IBSA lies in simultaneously constructing dynamic local elite population and global elite population and applying them to the design of search strategies. This construction concept

can make full use of the promising individuals that the population has already discovered, guiding the population to search for better solutions more quickly. Furthermore, the purpose of constructing the two elite populations is to store promising individuals; updating the two elite populations involves continuously refreshing these promising individuals as the search progresses; the designed search strategy utilizes these promising individuals. The original search strategy of BSA lacks guidance from elite individuals, resulting in a population search that is directionless. The introduced learning strategy is built upon both local and global elite populations, affording the whole population with valuable search guidance and enabling it to search purposefully. In other words, the proposed elite population-based search strategy can effectively compensate for the lack of search direction in BSA's population. This also distinguishes IBSA from other BSA variants.

In addition, In [Eq. \(9\)](#), we designed three search strategies, each assigned the same execution probability. This can be explained as follows: as shown in [Eq. \(9\)](#), Each of the three search strategies originates from a different design perspective. The first is the conventional BSA, the second is founded on the local elite population, and the third is founded on the global elite population. Given the lack of prior knowledge in the optimization task, it is sensible to adopt the same execution probability for strategies generated from three varying angles. Likewise, relative to the conventional BSA, the strategies that utilize either the global or the local elite population emphasize global exploration to a greater extent. Therefore, when prior knowledge of the optimization problem has already been obtained, if there is a need to enhance the global exploration ability of IBSA, it is recommended that the execution probabilities of the search strategies based on the global elite population and the local elite population be appropriately elevated while reducing the execution probability of the original BSA search strategy; conversely, if there is a need to enhance the local exploitation capability of IBSA, it is sensible to properly reduce the likelihood of executing the search strategies that are founded on the global elite population as well as the local one while increasing the execution probability of the original BSA search strategy.

4 Experimental Results in Challenging Numerical Functions

4.1 Test Functions

To verify the performance of IBSA, this section aims to use IBSA to solve the CEC 2014 test suite [\[33\]](#). The CEC 2014 test set is a very classic and challenging set of numerical optimization functions, which consists of 30 test functions. The 30 test functions are divided into four parts: three unimodal functions, 13 simple multimodal functions, six hybrid functions, and seven composition functions. In other words, 90% of the test functions are very complex functions, which are highly suitable for testing the global optimization performance of IBSA.

4.2 Parameter Settings

The compared algorithms are as follows: BSA [\[19\]](#), Jaya algorithm (JAYA) [\[34\]](#), sine cosine algorithm (SCA) [\[35\]](#), blood-sucking leech optimizer (BSLO) [\[36\]](#), and generalized normal distribution optimization (GNDO) [\[37\]](#). The dimension of the problem considered in solving the CEC 2014 test suite is 30. Each algorithm performs 30 independent repeated experiments in each test function, with 10,000 iterations per independent repeated experiment. In addition, the population size is 40. The other parameter settings of the compared algorithm are obtained from the relevant original studies. All simulations were carried out using MATLAB running on Windows 11 operating system. The computing setup includes a Core i7 processor (2.3 GHz) and 16 GB of RAM. The analysis methods for experimental results include the following three aspects:

- The value-based analysis method. This method aims to conduct statistical analysis on the best solutions derived from 30 independent trials of each algorithm in each test function, and obtain the average solution (MEAN) and standard deviation (STD).
- The significant difference-based analysis method. This method aims to perform the Wilcoxon signed rank test on the optimal solutions obtained by conducting 30 independent repeated experiments in each test function for each algorithm, to determine the significant differences between IBSA and the compared algorithms.
- The ranking-based analysis method. This method aims to perform the Friedman test on the best solutions derived from 30 independent trials for each test function of each algorithm, in order to determine the ranking of the algorithm in each test function.
- The convergence-based analysis method. This method aims to average the convergence history obtained from 30 independent repeated experiments on each test function of each algorithm, in order to compare the convergence performance of algorithms in each test function.

4.3 Experimental Results in Numerical Optimization Problems

Table 1 displays the experimental results of IBSA and the compared algorithms in solving the CEC 2014 test suite according to the value-based analysis method. From Table 1, according to MEAN, IBSA can find the best solutions in 19 functions, which are $F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_{11}, F_{12}, F_{13}, F_{14}, F_{15}, F_{16}, F_{17}, F_{18}, F_{19}, F_{20}, F_{21},$ and F_{26} . In addition, BSLO also shows strong competitiveness, which can get the best solutions in 18 functions including $F_5, F_7, F_8, F_{10}, F_{12}, F_{13}, F_{14}, F_{15}, F_{16}, F_{19}, F_{23}, F_{24}, F_{25}, F_{26}, F_{27}, F_{28}, F_{29},$ and F_{30} . JAYA can achieve the best solutions in $F_{12}, F_{13}, F_{16},$ and F_{26} . BSA can search for the best solutions in 12 functions including $F_2, F_3, F_7, F_9, F_{12}, F_{13}, F_{14}, F_{15}, F_{16}, F_{19}, F_{22},$ and F_{26} . SCA and GNDO can show their performance advantages in five functions (i.e., $F_{12}, F_{13}, F_{16}, F_{24},$ and F_{26}) and four functions (i.e., $F_{12}, F_{13}, F_{14},$ and F_{16}), respectively. According to STD in Table 1, IBSA, BSLO, BSA, JAYA, and SCA can win other algorithms in 10 functions (i.e., $F_1, F_2, F_3, F_4, F_{17}, F_{18}, F_{19}, F_{20}, F_{21},$ and F_{26}), 11 functions (i.e., $F_6, F_8, F_{10}, F_{12}, F_{23}, F_{24}, F_{25}, F_{27}, F_{28}, F_{29},$ and F_{30}), four functions (i.e., $F_7, F_{13}, F_{14},$ and F_{15}), three functions (i.e., $F_9, F_{11},$ and F_{16}), and two functions (i.e., F_5 and F_{22}), respectively. GNDO cannot compete with other algorithms, which cannot get the best STD in any functions. Based on the above analysis, IBSA demonstrates better overall performance in terms of solution accuracy and stability compared to the algorithms being compared. Comparing the performance of BSA and IBSA can be summarized as follows:

- According to MEAN, IBSA outperforms BSA in 13 functions, which are $F_1, F_4, F_5, F_6, F_{10}, F_{11}, F_{17}, F_{18}, F_{20}, F_{21}, F_{28}, F_{29},$ and F_{30} ; IBSA has the same performance with BSA in 14 functions, which are $F_2, F_3, F_7, F_{12}, F_{13}, F_{14}, F_{15}, F_{16}, F_{19}, F_{23}, F_{24}, F_{25}, F_{26},$ and F_{27} ; IBSA is inferior to BSA in three functions, which are $F_8, F_9,$ and F_{22} .
- According to STD, IBSA can beat BSA in 21 functions, which are $F_1, F_2, F_3, F_4, F_5, F_6, F_{10}, F_{11}, F_{12}, F_{16}, F_{17}, F_{18}, F_{19}, F_{20}, F_{21}, F_{22}, F_{25}, F_{26}, F_{27}, F_{28},$ and F_{30} ; IBSA cannot compete with BSA in nine functions, which are $F_7, F_8, F_9, F_{13}, F_{14}, F_{15}, F_{23}, F_{34},$ and F_{29} .

Table 1: The experimental results of IBSA and the compared algorithms in solving CEC 2014 test suite.

No.	Algorithm	JAYA	BSA	SCA	BSLO	GNDO	IBSA
F_1	MEAN	6.22E+07	9.35E+06	2.17E+08	1.59E+06	1.71E+06	6.33E+04
	STD	2.35E+07	4.92E+07	6.04E+07	8.16E+05	1.92E+06	4.33E+04

(Continued)

Table 1 (continued)

No.	Algorithm	JAYA	BSA	SCA	BSLO	GNDO	IBSA
F ₂	MEAN	5.71E+09	2.00E+02	1.68E+10	1.52E+04	2.57E+07	2.00E+02
	STD	1.28E+09	1.61E−10	3.05E+09	1.38E+04	1.41E+08	1.04E−10
F ₃	MEAN	4.60E+04	3.00E+02	3.53E+04	2.75E+03	3.00E+02	3.00E+02
	STD	1.40E+04	2.28E−01	6.06E+03	2.27E+03	5.37E−04	1.13E−06
F ₄	MEAN	7.45E+02	4.59E+02	1.33E+03	4.85E+02	5.19E+02	4.04E+02
	STD	7.29E+01	3.88E+01	2.03E+02	3.55E+01	4.13E+01	1.03E+01
F ₅	MEAN	5.21E+02	5.21E+02	5.21E+02	5.20E+02	5.21E+02	5.20E+02
	STD	6.04E−02	1.54E−01	4.18E−02	4.21E−02	1.28E−01	9.05E−02
F ₆	MEAN	6.33E+02	6.22E+02	6.33E+02	6.17E+02	6.30E+02	6.13E+02
	STD	3.11E+00	7.50E+00	2.96E+00	2.84E+00	3.14E+00	3.04E+00
F ₇	MEAN	7.14E+02	7.00E+02	8.31E+02	7.00E+02	7.02E+02	7.00E+02
	STD	4.37E+00	1.23E−02	2.46E+01	3.23E−02	2.98E+00	1.82E−02
F ₈	MEAN	1.01E+03	8.22E+02	1.04E+03	8.00E+02	9.20E+02	8.44E+02
	STD	2.16E+01	1.10E+01	1.39E+01	1.15E−08	1.85E+01	1.54E+01
F ₉	MEAN	1.15E+03	9.76E+02	1.17E+03	1.02E+03	1.04E+03	9.97E+02
	STD	1.32E+01	1.83E+01	1.94E+01	2.19E+01	2.76E+01	2.50E+01
F ₁₀	MEAN	6.30E+03	2.47E+03	7.00E+03	1.02E+03	4.02E+03	1.23E+03
	STD	6.55E+02	7.86E+02	4.33E+02	7.04E+00	5.28E+02	2.20E+02
F ₁₁	MEAN	7.90E+03	5.97E+03	7.96E+03	3.84E+03	4.59E+03	3.71E+03
	STD	2.97E+02	1.05E+03	3.37E+02	4.84E+02	5.93E+02	6.02E+02
F ₁₂	MEAN	1.20E+03	1.20E+03	1.20E+03	1.20E+03	1.20E+03	1.20E+03
	STD	1.88E−01	3.56E−01	2.04E−01	4.25E−02	5.87E−01	1.46E−01
F ₁₃	MEAN	1.30E+03	1.30E+03	1.30E+03	1.30E+03	1.30E+03	1.30E+03
	STD	1.93E−01	6.36E−02	3.46E−01	1.51E−01	1.24E−01	7.59E−02
F ₁₄	MEAN	1.41E+03	1.40E+03	1.44E+03	1.40E+03	1.40E+03	1.40E+03
	STD	3.07E+00	4.29E−02	8.77E+00	2.97E−01	1.20E−01	7.36E−02
F ₁₅	MEAN	1.53E+03	1.51E+03	3.85E+03	1.51E+03	2.20E+03	1.51E+03
	STD	5.51E+00	2.59E+00	2.04E+03	3.01E+00	8.58E+02	2.67E+00
F ₁₆	MEAN	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03
	STD	1.76E−01	7.25E−01	2.36E−01	7.57E−01	5.66E−01	5.55E−01
F ₁₇	MEAN	3.49E+06	1.34E+04	6.04E+06	3.16E+05	2.28E+04	9.52E+03
	STD	1.47E+06	9.95E+03	3.18E+06	2.21E+05	3.75E+04	5.45E+03
F ₁₈	MEAN	1.59E+07	2.51E+07	1.27E+08	8.77E+03	3.60E+03	2.50E+03
	STD	2.32E+07	1.37E+08	7.96E+07	7.16E+03	2.89E+03	8.99E+02

(Continued)

Table 1 (continued)

No.	Algorithm	JAYA	BSA	SCA	BSLO	GNDO	IBSA
F ₁₉	MEAN	1.93E+03	1.91E+03	1.99E+03	1.91E+03	1.93E+03	1.91E+03
	STD	1.53E+01	1.07E+01	2.45E+01	1.49E+01	2.03E+01	1.76E+00
F ₂₀	MEAN	5.57E+03	2.29E+03	1.56E+04	4.26E+03	2.22E+03	2.08E+03
	STD	1.65E+03	2.76E+02	4.25E+03	2.33E+03	9.69E+01	6.11E+01
F ₂₁	MEAN	5.71E+05	1.01E+04	1.09E+06	1.38E+05	4.78E+03	4.73E+03
	STD	3.53E+05	1.33E+04	4.12E+05	1.31E+05	4.00E+03	1.78E+03
F ₂₂	MEAN	2.76E+03	2.43E+03	2.96E+03	2.75E+03	2.75E+03	2.45E+03
	STD	1.53E+02	1.23E+02	9.93E+01	2.25E+02	2.16E+02	1.09E+02
F ₂₃	MEAN	2.64E+03	2.62E+03	2.66E+03	2.50E+03	2.62E+03	2.62E+03
	STD	3.96E+00	1.31E−12	9.63E+00	0.00E+00	3.39E−04	1.47E−12
F ₂₄	MEAN	2.62E+03	2.63E+03	2.60E+03	2.60E+03	2.64E+03	2.63E+03
	STD	2.24E+01	4.15E+00	2.64E−02	0.00E+00	8.28E+00	5.54E+00
F ₂₅	MEAN	2.72E+03	2.71E+03	2.73E+03	2.70E+03	2.71E+03	2.71E+03
	STD	4.86E+00	4.86E+00	5.69E+00	0.00E+00	2.60E+00	4.63E+00
F ₂₆	MEAN	2.70E+03	2.70E+03	2.70E+03	2.70E+03	2.76E+03	2.70E+03
	STD	2.96E−01	4.76E−01	5.16E−01	1.19E−01	5.02E+01	6.12E−02
F ₂₇	MEAN	3.73E+03	3.62E+03	3.42E+03	2.90E+03	3.68E+03	3.25E+03
	STD	2.17E+02	2.41E+02	3.04E+02	0.00E+00	2.82E+02	1.42E+02
F ₂₈	MEAN	4.03E+03	4.04E+03	4.77E+03	3.00E+03	4.29E+03	3.76E+03
	STD	2.02E+02	3.22E+02	3.57E+02	0.00E+00	3.04E+02	8.18E+01
F ₂₉	MEAN	4.75E+06	7.82E+06	1.15E+07	3.10E+03	4.15E+06	4.26E+06
	STD	4.21E+06	1.99E+06	8.10E+06	0.00E+00	5.63E+06	4.48E+06
F ₃₀	MEAN	1.32E+04	7.58E+04	2.36E+05	3.20E+03	7.40E+03	5.57E+03
	STD	3.76E+03	7.27E+04	9.56E+04	0.00E+00	1.90E+03	1.27E+03

Note: JAYA, Jaya algorithm; BSA, backtracking search algorithm; SCA, sine cosine algorithm; BSLO, blood-sucking leech optimizer; GNDO, generalized normal distribution optimization; IBSA, improved backtracking search algorithm; CEC, Congress on Evolutionary Computation; MEAN, mean solution; STD, standard deviation.

Obviously, IBSA is significantly better than BSA in terms of solving accuracy and stability. This also indicates that the designed improvement strategy is very effective and can significantly enhance the global search capability of BSA. In addition, Fig. 1 shows some typical convergence curves of IBSA and the compared algorithms in solving CEC 2014 test suite, which indicates the superior global exploration capacity and the ability to jump out of local optima of IBSA.

As listed in Table 2, the average ranking performance of IBSA and the baseline algorithms for the CEC 2014 suite is presented. From Table 2, IBSA and BSLO are significantly better than the other six algorithms. IBSA and BSLO can get the best ranking in 11 functions (i.e., F₁, F₂, F₄, F₆, F₇, F₁₁, F₁₅, F₁₆, F₁₈, F₂₀, and F₂₆) and 11 functions (i.e., F₅, F₈, F₁₀, F₁₂, F₂₃, F₂₄, F₂₅, F₂₇, F₂₈, F₂₉, and F₃₀), respectively. In addition, BSA and GNDO can get the best ranking in five functions (i.e., F₉, F₁₃, F₁₄, F₁₉, and F₂₂) and three functions

(i.e., F_3 , F_{17} , and F_{21}), respectively. JAYA and SCA cannot win other algorithms in any functions. Thus, based on the average ranking in the last row, the algorithms from best to worst are as follows: IBSA, BSLO, BSA, GNDO, JAYA, and SCA. According to the ranking results of IBSA and BSA, BSA outperforms IBSA in 8 functions including F_8 , F_9 , F_{13} , F_{14} , F_{19} , F_{22} , F_{24} , and F_{25} . However, BSA is inferior to IBSA in the remaining 22 functions. That is to say, IBSA has demonstrated the better global search ability than BSA, which likewise shows the validity of the improved approach.

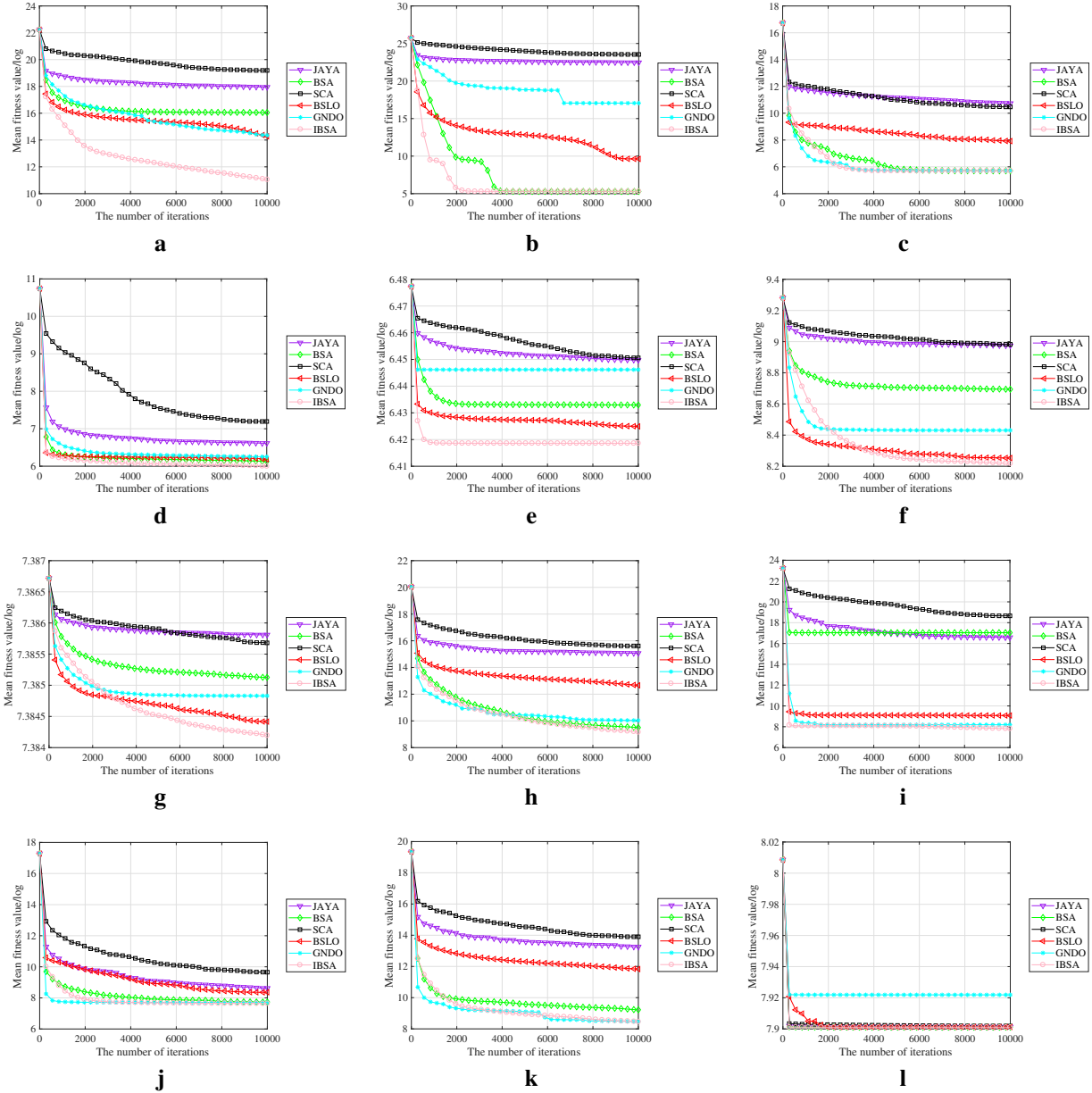


Figure 1: 12 representative convergence curves for eight algorithms in the 30-dimensional CEC 2014 benchmark suite: (a) F_1 , (b) F_2 , (c) F_3 , (d) F_4 , (e) F_6 , (f) F_{11} , (g) F_{16} , (h) F_{17} , (i) F_{18} , (j) F_{20} , (k) F_{21} , and (l) F_{26} ; Note: JAYA, Jaya algorithm; BSA, backtracking search algorithm; SCA, sine cosine algorithm; BSLO, blood-sucking leech optimizer; GNDO, generalized normal distribution optimization; IBSA, improved backtracking search algorithm.

Table 2: Average ranks of IBSA and its competing algorithms in the CEC 2014 benchmark suite.

No.	JAYA	BSA	SCA	BSLO	GNDO	IBSA
F ₁	4.967	2.267	5.967	3.533	3.200	1.067
F ₂	5.000	1.783	6.000	3.900	3.100	1.217
F ₃	5.733	2.967	5.267	4.000	1.300	1.733
F ₄	5.000	2.400	6.000	2.767	3.700	1.133
F ₅	4.800	3.167	5.067	1.000	4.967	2.000
F ₆	4.967	3.000	5.200	2.033	4.400	1.400
F ₇	5.000	1.950	6.000	2.433	3.833	1.783
F ₈	5.100	2.100	5.900	1.000	4.000	2.900
F ₉	5.167	1.300	5.833	3.000	3.633	2.067
F ₁₀	5.167	3.067	5.800	1.100	3.967	1.900
F ₁₁	5.333	3.767	5.667	1.733	2.933	1.567
F ₁₂	5.533	3.233	5.467	1.100	3.300	2.367
F ₁₃	5.000	1.433	6.000	3.667	3.167	1.733
F ₁₄	5.000	1.400	6.000	3.800	2.467	2.333
F ₁₅	4.000	2.133	5.867	2.400	5.133	1.467
F ₁₆	5.700	3.533	5.200	2.167	2.967	1.433
F ₁₇	5.300	2.233	5.700	3.933	1.867	1.967
F ₁₈	5.067	2.833	5.867	3.200	2.267	1.767
F ₁₉	4.633	1.500	5.933	2.967	4.233	1.733
F ₂₀	4.700	2.333	6.000	4.167	2.600	1.200
F ₂₁	5.100	2.467	5.833	4.067	1.700	1.833
F ₂₂	3.967	1.633	5.467	4.200	4.033	1.700
F ₂₃	5.000	2.900	6.000	1.000	4.000	2.100
F ₂₄	4.367	3.933	2.000	1.000	5.367	4.333
F ₂₅	4.733	2.400	5.767	1.000	3.967	3.133
F ₂₆	4.400	1.800	5.367	3.267	4.800	1.367
F ₂₇	4.967	4.467	3.600	1.000	4.533	2.433
F ₂₈	3.733	3.567	5.833	1.000	4.733	2.133
F ₂₉	4.000	4.333	4.867	1.000	3.567	3.233
F ₃₀	4.233	4.633	5.867	1.000	3.000	2.267
Avg.	4.856	2.684	5.511	2.414	3.558	1.977

Note: JAYA, Jaya algorithm; BSA, backtracking search algorithm; SCA, sine cosine algorithm; BSLO, blood-sucking leech optimizer; GNDO, generalized normal distribution optimization; IBSA, improved backtracking search algorithm; CEC, Congress on Evolutionary Computation.

Table 3 describes the Wilcoxon signed test results of IBSA and the compared algorithms in solving CEC 2014 test suite. Table 3 lists several statistical indicators with a significance level of 0.05: the positive rank sum “R(+)”, the negative rank sum “R(−)”, the resulting sign “S”, and the test p -value. The sign “S” takes three values: “+” (IBSA performs better than the compared algorithm in the given function), “−” (IBSA is worse), and “=” (both algorithms achieve equivalent performance). From Table 3, SCA only outperforms IBSA in F₂₄, while SCA loses to IBSA in the remaining 29 functions. JAYA has the same performance with IBSA in F₂₄ and F₂₉, which cannot compete with IBSA in the remaining 28 functions. Compared to IBSA, BSA has advantages in five functions (i.e., F₈, F₉, F₁₄, F₂₄, and F₂₅), has the same performance in five functions (i.e., F₇, F₁₃, F₁₉, F₂₂, and F₂₃), and has disadvantages in the remaining 20 functions. GNDO is only superior to IBSA in F₃. It performs identically to IBSA on five functions (i.e., F₁₄, F₁₇, F₁₈, F₂₁, and F₂₉), but is inferior to IBSA in the remaining 24 functions. BSLO outperforms IBSA in 11 functions (i.e., F₅, F₈, F₁₀, F₁₂, F₂₃, F₂₄, F₂₅, F₂₇, F₂₈, F₂₉, and F₃₀), but still lags behind IBSA in more than half of the functions including F₁, F₂, F₃, F₄, F₆, F₉, F₁₃, F₁₄, F₁₅, F₁₇, F₁₈, F₁₉, F₂₀, F₂₁, F₂₂, and F₂₆. Based on the above discussion, the performance of IBSA is significantly superior to that of JAYA, BSA, SCA, and GNDO. BSLO demonstrates excellent overall search capabilities, but its performance is still inferior to that of IBSA. Overall, based on the Wilcoxon signed test results of the significance test, IBSA is the top performer among all competing algorithms, which also attests to the high efficacy of the design enhancements.

Table 3: Results of the Wilcoxon signed-rank test comparing IBSA with other algorithms over the CEC 2014 problems, with a significance threshold of 0.05.

No.	IBSA vs.	JAYA			BSA			SCA			BSLO			GNDO							
		R(+)	R(-)	p-Value	S	R(+)	R(-)	p-Value	S	R(+)	R(-)	p-Value	S	R(+)	R(-)	p-Value	S				
F ₁		465	0	1.73E-06	+	462	3	2.35E-06	+	465	0	1.73E-06	+	465	0	1.73E-06	+	465	0	1.73E-06	+
F ₂		465	0	1.73E-06	+	331.5	133.5	7.49E-03	+	465	0	1.73E-06	+	465	0	1.73E-06	+	465	0	1.73E-06	+
F ₃		465	0	1.73E-06	+	465	0	1.73E-06	+	465	0	1.73E-06	+	465	0	1.73E-06	+	134	331	4.28E-02	-
F ₄		465	0	1.73E-06	+	446	19	1.13E-05	+	465	0	1.73E-06	+	465	0	1.73E-06	+	465	0	1.73E-06	+
F ₅		465	0	1.73E-06	+	465	0	1.73E-06	+	465	0	1.73E-06	+	465	0	1.73E-06	+	465	0	1.73E-06	+
F ₆		465	0	1.73E-06	+	440	25	1.97E-05	+	465	0	1.73E-06	+	424	41	8.19E-05	+	465	0	1.73E-06	+
F ₇		465	0	1.73E-06	+	220	245	5.86E-01	=	465	0	1.73E-06	+	265	200	5.04E-01	=	454	11	5.22E-06	+
F ₈		465	0	1.73E-06	+	23	442	1.64E-05	-	465	0	1.73E-06	+	0	465	1.73E-06	-	465	0	1.73E-06	+
F ₉		465	0	1.73E-06	+	41	424	8.19E-05	-	465	0	1.73E-06	+	368	97	5.32E-03	+	464	1	1.92E-06	+
F ₁₀		465	0	1.73E-06	+	465	0	1.73E-06	+	465	0	1.73E-06	+	17	448	9.32E-06	-	465	0	1.73E-06	+
F ₁₁		465	0	1.73E-06	+	463	2	2.13E-06	+	465	0	1.73E-06	+	272	193	4.17E-01	=	438	27	2.37E-05	+
F ₁₂		465	0	1.73E-06	+	408	57	3.06E-04	+	465	0	1.73E-06	+	9	456	4.29E-06	-	410	55	2.61E-04	+
F ₁₃		465	0	1.73E-06	+	160	305	1.36E-01	=	465	0	1.73E-06	+	463	2	2.13E-06	+	461	4	2.60E-06	+
F ₁₄		465	0	1.73E-06	+	60	405	3.88E-04	-	465	0	1.73E-06	+	455	10	4.73E-06	+	277	188	3.60E-01	=
F ₁₅		465	0	1.73E-06	+	358	107	9.84E-03	+	465	0	1.73E-06	+	412	53	2.22E-04	+	465	0	1.73E-06	+
F ₁₆		465	0	1.73E-06	+	458	7	3.52E-06	+	465	0	1.73E-06	+	318	147	7.86E-02	=	453	12	5.75E-06	+
F ₁₇		465	0	1.73E-06	+	344	121	2.18E-02	+	465	0	1.73E-06	+	465	0	1.73E-06	+	265	200	5.04E-01	=
F ₁₈		465	0	1.73E-06	+	384	81	1.83E-03	+	465	0	1.73E-06	+	449	16	8.47E-06	+	316	149	8.59E-02	=
F ₁₉		465	0	1.73E-06	+	174	291	2.29E-01	=	465	0	1.73E-06	+	447	18	1.02E-05	+	465	0	1.73E-06	+
F ₂₀		465	0	1.73E-06	+	412	53	2.22E-04	+	465	0	1.73E-06	+	465	0	1.73E-06	+	463	2	2.13E-06	+
F ₂₁		465	0	1.73E-06	+	368	97	5.32E-03	+	465	0	1.73E-06	+	465	0	1.73E-06	+	178	287	2.62E-01	=
F ₂₂		460	5	2.88E-06	+	182	283	2.99E-01	=	465	0	1.73E-06	+	446	19	1.13E-05	+	465	0	1.73E-06	+
F ₂₃		465	0	1.73E-06	+	235.5	229.5	2.50E-01	=	465	0	1.73E-06	+	0	465	4.32E-08	-	465	0	1.73E-06	+
F ₂₄		154	311	1.06E-01	=	134	331	4.28E-02	-	0	465	1.73E-06	-	0	465	1.73E-06	-	432	33	4.07E-05	+
F ₂₅		438	27	2.37E-05	+	120	345	2.07E-02	-	465	0	1.73E-06	+	0	465	1.73E-06	-	365	100	6.42E-03	+
F ₂₆		465	0	1.73E-06	+	340	125	2.70E-02	+	465	0	1.73E-06	+	465	0	1.73E-06	+	465	0	1.73E-06	+
F ₂₇		465	0	1.73E-06	+	454	11	5.22E-06	+	354	111	1.25E-02	+	0	465	1.73E-06	-	452	13	6.34E-06	+
F ₂₈		465	0	1.73E-06	+	431	34	4.45E-05	+	465	0	1.73E-06	+	0	465	1.73E-06	-	465	0	1.73E-06	+
F ₂₉		286	179	2.71E-01	=	378.5	86.5	8.85E-04	+	442	23	1.64E-05	+	0	465	1.73E-06	-	254	211	6.58E-01	=
F ₃₀		465	0	1.73E-06	+	462	3	2.35E-06	+	465	0	1.73E-06	+	0	465	1.73E-06	-	411	54	2.41E-04	+
+/-/-		28/2/0			20/5/5			29/0/1			16/3/11			24/5/1							

Note: JAYA, Jaya algorithm; BSA, backtracking search algorithm; SCA, sine cosine algorithm; BSLO, blood-sucking leech optimizer; GNDO, generalized normal distribution optimization; IBSA, improved backtracking search algorithm; CEC, Congress on Evolutionary Computation.

The CEC 2014 test suite involves four different types of functions: unimodal functions, multimodal functions, composition functions, and hybrid functions. As discussed above, compared with the algorithms used for comparison, IBSA demonstrates superior performance across various evaluation metrics when solving the CEC 2014 test suite. This is primarily attributed to the search mechanism designed in IBSA. On the one hand, the constructed local elite population and global elite population preserve highly promising individuals; by incorporating these promising individuals into the search strategy, IBSA's convergence speed and global search capability can be enhanced. On the other hand, with three different search rules incorporated in its search mechanism, IBSA can enrich population diversity and somewhat strengthen its ability to escape local optima. Hence, IBSA achieves excellent optimization results.

4.4 A Study on the Contribution of the Elite Population

This section studies the contributions of the locally elite population and the globally elite population, as defined in this research, to the IBSA. In addition, the version that retains only the search strategy designed for the local elite population along with the original BSA's search strategy is referred to as IBSA-1, while the version that retains only the search strategy designed for the entire elite population along with the original BSA's search strategy is referred to as IBSA-2. In this section, the focus is on comparing the performance differences among BSA, IBSA, IBSA-1, and IBSA-2. The relevant experimental parameters are exactly the same as those in [Section 4.2](#). The experimental results are presented in [Table 4](#), from which we can draw the following conclusions:

- Comparison results between IBSA and BSA (this comparison is significant because it examines how the improvement strategies that rely on local and global elite populations affect IBSA). IBSA can achieve the better MEAN than BSA in 14 functions, which are $F_1, F_4, F_5, F_6, F_{10}, F_{11}, F_{17}, F_{18}, F_{20}, F_{21}, F_{27}, F_{28}, F_{29}$, and F_{30} . In addition, IBSA and BSA can get the same MEAN in 13 functions, which are $F_2, F_3, F_7, F_{12}, F_{13}, F_{14}, F_{15}, F_{16}, F_{19}, F_{23}, F_{24}, F_{25}$, and F_{26} . IBSA can achieve better STD than BSA in seventy percent of the functions, which are $F_1, F_2, F_3, F_4, F_5, F_6, F_{10}, F_{11}, F_{12}, F_{16}, F_{17}, F_{18}, F_{19}, F_{20}, F_{21}, F_{22}, F_{25}, F_{26}, F_{27}, F_{28}$, and F_{30} .
- Comparison results between IBSA and IBSA-1 (the significance of this comparison lies in studying the impact of improvement strategy based on global elite population on IBSA). IBSA outperforms IBSA-1 in 12 functions in terms of MEAN, which are $F_1, F_4, F_{10}, F_{11}, F_{17}, F_{18}, F_{20}, F_{21}, F_{27}, F_{28}, F_{29}$, and F_{30} . IBSA-1 can obtain better MEAN than IBSA in three functions, i.e., F_8, F_9 , and F_{22} . For the remaining 15 functions including $F_2, F_3, F_5, F_6, F_7, F_{12}, F_{13}, F_{14}, F_{15}, F_{16}, F_{19}, F_{23}, F_{24}, F_{25}$, and F_{26} , IBSA and IBSA-1 have the same MEAN.
- Comparison results between IBSA and IBSA-2 (the significance of this comparison lies in studying the impact of improvement strategy based on local elite population on IBSA). According to MEAN, IBSA is superior to IBSA-2 in more than half of the functions including $F_1, F_4, F_6, F_8, F_9, F_{10}, F_{11}, F_{18}, F_{20}, F_{21}, F_{22}, F_{26}, F_{27}, F_{28}, F_{29}$, and F_{30} . IBSA-2 only has a better MEAN than IBSA in F_7 . IBSA and IBSA-2 have the same performance in the remaining 13 functions, namely $F_2, F_3, F_5, F_7, F_{12}, F_{13}, F_{14}, F_{15}, F_{16}, F_{19}, F_{23}, F_{24}$, and F_{25} .
- Comparison results between IBSA-1 and BSA (the significance of this comparison lies in studying the impact of improvement strategy based on local elite population on BSA). In terms of MEAN, BSA performs worse than IBSA-1 in 14 function, which are $F_1, F_5, F_6, F_{10}, F_{11}, F_{17}, F_{18}, F_{20}, F_{21}, F_{22}, F_{27}, F_{28}, F_{29}$, and F_{30} . In addition, IBSA-1 and BSA can obtain the same MEAN in 13 functions, which are $F_2, F_3, F_7, F_{12}, F_{13}, F_{14}, F_{15}, F_{16}, F_{19}, F_{23}, F_{24}, F_{25}$, and F_{26} . In terms of STD, BSA performs worse than IBSA-1 in 21 functions, which are $F_1, F_3, F_4, F_5, F_6, F_{10}, F_{11}, F_{12}, F_{13}, F_{16}, F_{18}, F_{19}, F_{20}, F_{21}, F_{22}, F_{24}, F_{25}, F_{26}, F_{27}, F_{28}$, and F_{30} .

- Comparison results between IBSA-2 and BSA (the significance of this comparison lies in studying the impact of improvement strategy based on global elite population on BSA). IBSA-2 can achieve a better MEAN than BSA in 14 functions including $F_1, F_4, F_5, F_6, F_{10}, F_{11}, F_{17}, F_{18}, F_{20}, F_{21}, F_{27}, F_{28}, F_{29}$, and F_{30} . In addition, IBSA-2 and BSA can get the same MEAN in 12 functions, which are $F_2, F_3, F_7, F_{12}, F_{13}, F_{14}, F_{15}, F_{16}, F_{19}, F_{23}, F_{24}$, and F_{25} . In terms of STD, IBSA-2 can beat BSA in 19 functions including $F_1, F_2, F_3, F_4, F_5, F_6, F_{10}, F_{11}, F_{12}, F_{15}, F_{16}, F_{17}, F_{18}, F_{20}, F_{21}, F_{22}, F_{27}, F_{28}$, and F_{30} . Moreover, IBSA-2 and BSA can get the same STD in F_{25} .

Table 4: The experimental results of BSA, IBSA-1, IBSA-2, and IBSA in solving CEC 2014 test suite.

No.	BSA		IBSA-1		IBSA-2		IBSA	
	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
F_1	9.35E+06	4.92E+07	1.05E+05	7.69E+04	8.93E+04	4.72E+04	6.33E+04	4.33E+04
F_2	2.00E+02	1.61E−10	2.00E+02	1.29E−09	2.00E+02	3.45E−12	2.00E+02	1.04E−10
F_3	3.00E+02	2.28E−01	3.00E+02	7.57E−06	3.00E+02	7.39E−05	3.00E+02	1.13E−06
F_4	4.59E+02	3.88E+01	4.66E+02	3.63E+01	4.18E+02	2.63E+01	4.04E+02	1.03E+01
F_5	5.21E+02	1.54E−01	5.20E+02	1.10E−01	5.20E+02	1.01E−01	5.20E+02	9.05E−02
F_6	6.22E+02	7.50E+00	6.13E+02	3.09E+00	6.15E+02	2.81E+00	6.13E+02	3.04E+00
F_7	7.00E+00	1.23E−02	7.00E+02	1.99E−02	7.00E+02	1.30E−02	7.00E+02	1.82E−02
F_8	8.22E+02	1.10E+01	8.29E+02	1.30E+01	8.51E+02	2.26E+01	8.44E+02	1.54E+01
F_9	9.76E+02	1.83E+01	9.83E+02	1.95E+01	1.00E+03	2.34E+01	9.97E+02	2.50E+01
F_{10}	2.47E+03	7.86E+02	1.28E+03	2.48E+02	1.42E+03	2.57E+02	1.23E+03	2.20E+02
F_{11}	5.97E+03	1.05E+03	4.21E+03	7.84E+02	3.99E+03	7.51E+02	3.71E+03	6.02E+02
F_{12}	1.20E+03	3.56E−01	1.20E+03	1.33E−01	1.20E+03	1.66E−01	1.20E+03	1.46E−01
F_{13}	1.30E+03	6.36E−02	1.30E+03	5.92E−02	1.30E+03	9.60E−02	1.30E+03	7.59E−02
F_{14}	1.40E+03	4.29E−02	1.40E+03	5.32E−02	1.40E+03	9.78E−02	1.40E+03	7.36E−02
F_{15}	1.51E+03	2.59E+00	1.51E+03	4.23E+00	1.51E+03	1.96E+00	1.51E+03	2.67E+00
F_{16}	1.61E+03	7.25E−01	1.61E+03	6.32E−01	1.61E+03	5.43E−01	1.61E+03	5.55E−01
F_{17}	1.34E+04	9.95E+03	1.22E+04	1.07E+04	8.52E+03	4.26E+03	9.52E+03	5.45E+03
F_{18}	2.51E+07	1.37E+08	3.23E+03	1.89E+03	2.56E+03	8.95E+02	2.50E+03	8.99E+02
F_{19}	1.91E+03	1.07E+01	1.91E+03	1.21E+00	1.91E+03	1.51E+01	1.91E+03	1.76E+00
F_{20}	2.29E+03	2.76E+02	2.09E+03	5.72E+01	2.11E+03	1.05E+02	2.08E+03	6.11E+01
F_{21}	1.01E+04	1.33E+04	6.13E+03	3.52E+03	5.56E+03	4.27E+03	4.73E+03	1.78E+03
F_{22}	2.43E+03	1.23E+02	2.39E+03	7.30E+01	2.46E+03	1.08E+02	2.45E+03	1.09E+02
F_{23}	2.62E+03	1.31E−12	2.62E+03	2.11E−12	2.62E+03	1.74E−12	2.62E+03	1.47E−12
F_{24}	2.63E+03	4.15E+00	2.63E+03	3.88E+00	2.63E+03	7.09E+00	2.63E+03	5.54E+00
F_{25}	2.71E+03	4.86E+00	2.71E+03	2.52E+00	2.71E+03	4.86E+00	2.71E+03	4.63E+00
F_{26}	2.70E+03	4.76E−01	2.70E+03	1.04E−01	2.71E+03	2.53E+01	2.70E+03	6.12E−02
F_{27}	3.62E+03	2.41E+02	3.31E+03	1.36E+02	3.29E+03	1.76E+02	3.25E+03	1.42E+02
F_{28}	4.04E+03	3.22E+02	3.79E+03	2.69E+02	3.85E+03	1.67E+02	3.76E+03	8.18E+01
F_{29}	7.82E+06	1.99E+06	5.08E+06	4.04E+06	5.63E+06	4.08E+06	4.26E+06	4.48E+06
F_{30}	7.58E+04	7.27E+04	6.45E+03	1.12E+03	5.64E+03	1.51E+03	5.57E+03	1.27E+03

Note: BSA, backtracking search algorithm; IBSA-1, the first variant of IBSA; IBSA-2, the second variant of IBSA; IBSA, improved backtracking search algorithm; MEAN, mean solution; STD, standard deviation.

Based on the above analysis, IBSA is significantly superior to BSA, IBSA-1 and IBSA-2. The performance of IBSA-1 and IBSA-2 is significantly superior to that of BSA. Put differently, the search strategies built upon the local and global elite populations together help IBSA achieve excellent global search ability. This indicates that the two elite-population-based search strategies designed by IBSA are not a simple mechanical

combination, but an organic integration, jointly contributing to IBSA's outstanding global search capability. This also demonstrates the effectiveness of the designed search strategies from another perspective.

5 Performance Evaluation in Engineering Optimization Problems

This section serves to test the effectiveness of IBSA for engineering optimization tasks. The algorithms applied here are identical to those in Section 4. A population size of 40, a maximum iteration limit of 2000, and 30 independent trials per problem are adopted uniformly across all algorithms. Other parameters of the comparison algorithms are taken from their respective source literature. This section addresses the constraints associated with the engineering problems to be solved by using penalty functions as done in [38]. In addition, this section also introduces some recently reported results to assess IBSA's capability in addressing practical engineering problems. In this section, WORST, MEAN, MEDIAN, BEST, and STD are defined as the worst, mean, median, and best solutions and the standard deviation obtained over 30 independent executions, respectively.

5.1 The Welded Beam Design Problem

For the welded beam design problem, the objective function is given by:

$$f(\mathbf{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \quad (10)$$

The related constraints can be summarized as follows.

$$g_1(\mathbf{x}) = \tau(\mathbf{x}) - \tau_{\max} \leq 0, \quad g_2(\mathbf{x}) = \sigma(\mathbf{x}) - \sigma_{\max} \leq 0, \quad g_3(\mathbf{x}) = x_1 - x_4 \leq 0, \quad (11a)$$

$$g_4(\mathbf{x}) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \leq 0, \quad g_5(\mathbf{x}) = 0.125 - x_1 \leq 0, \quad (11b)$$

$$g_6(\mathbf{x}) = \delta(\mathbf{x}) - \delta_{\max} \leq 0, \quad g_7(\mathbf{x}) = P - P_c(\mathbf{x}), \quad (11c)$$

where $\tau(\mathbf{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$, $\tau' = \frac{P}{\sqrt{2}x_1x_2}$, $\tau'' = \frac{MR}{J}$, $M = P(L + \frac{x_2}{2})$, $R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1+x_3}{2})^2}$, $\sigma(\mathbf{x}) = \frac{6PL}{x_4x_3^3}$, $\delta(\mathbf{x}) = \frac{4PL^3}{Ex_3^3x_4}$, $J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\}$, $P_c(\mathbf{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$, $P = 6000$ lb, $L = 14$ in, $E = 30 \times 10^6$ psi, $G = 12 \times 10^6$ psi. In addition, $0.1 \leq x_1 \leq 2$, $0.1 \leq x_2 \leq 10$, $0.1 \leq x_3 \leq 10$, and $0.1 \leq x_4 \leq 2$.

The statistical results obtained by IBSA and the compared algorithms in solving the welded beam design problem have been presented in Table 5. From Table 5, IBSA, JAYA, BSLO, and GNDO can find the best solution, which is 1.72485231. The optimal solutions obtained by BSA is 1.72485412. That is to say, the optimal solution obtained by BSA is very close to those obtained by IBSA, JAYA, BSLO and GNDO. The optimal solution obtained by SCA is significantly inferior to those of other algorithms. Table 6 shows the optimal parameters corresponding to the optimal solutions obtained by IBSA and the compared algorithms. Fig. 2 describes the convergence curves of IBSA and the compared algorithms. Looking at Fig. 2, as shown in Tables 5 and 6, the numerical solutions obtained by IBSA and the comparative algorithms do not differ significantly; therefore, the convergence curves are also very similar. Table 7 presents a comparison between the best solution obtained by IBSA and the solutions reported in the literature for the welded beam design problem. In Table 7, the solution of IBSA is superior to the solutions of the four algorithms it is compared with. The optimal solution of GSO is 1.72491. It is very close to the optimal solution of IBSA (i.e., 1.72485231), indicating the excellent performance of GSO in solving this problem. The optimal solutions obtained by the other comparison algorithms show a significant gap compared to IBSA.

Table 5: The statistical results obtained by IBSA and the compared algorithms in solving the welded beam design problem.

Algorithm	WORST	MEAN	MEDIAN	BEST	STD
JAYA	1.72485231	1.72485231	1.72485231	1.72485231	9.03E−16
BSA	1.75906358	1.72890658	1.72499463	1.72485412	8.41E−03
SCA	1.90347905	1.84374026	1.83531584	1.76915318	3.21E−02
BSLO	4.04699528	2.11202598	1.93375793	1.72485231	5.08E−01
GNDO	1.72485231	1.72485231	1.72485231	1.72485231	9.03E−16
IBSA	1.82464715	1.73484169	1.72489725	1.72485231	2.25E−02

Note: JAYA, Jaya algorithm; BSA, backtracking search algorithm; SCA, sine cosine algorithm; BSLO, blood-sucking leech optimizer; GNDO, generalized normal distribution optimization; IBSA, improved backtracking search algorithm; BEST, the best solution; MEAN, mean solution; STD, standard deviation; WORST, the worst solution; MEDIAN, the median solution.

Table 6: The best results obtained by IBSA and the compared algorithms in solving the welded beam design problem.

Algorithm	x_1	x_2	x_3	x_4	$f(x^*)$
JAYA	0.20572964	3.47048867	9.03662391	0.20572964	1.72485231
BSA	0.20572951	3.47049614	9.03662336	0.20572978	1.72485412
SCA	0.19891913	3.67981015	9.09498698	0.20789896	1.76915318
BSLO	0.20572964	3.47048867	9.03662391	0.20572964	1.72485231
GNDO	0.20572964	3.47048867	9.03662391	0.20572964	1.72485231
IBSA	0.20572964	3.47048867	9.03662391	0.20572964	1.72485231

Note: JAYA, Jaya algorithm; BSA, backtracking search algorithm; SCA, sine cosine algorithm; BSLO, blood-sucking leech optimizer; GNDO, generalized normal distribution optimization; IBSA, improved backtracking search algorithm; x^* , the optimal solution; $f(x^*)$, the optimal value.

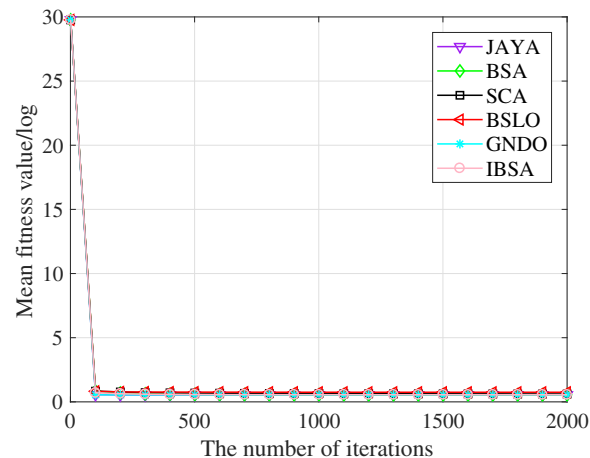
**Figure 2:** Convergence behavior for the welded beam design problem; JAYA, Jaya algorithm; BSA, backtracking search algorithm; SCA, sine cosine algorithm; BSLO, blood-sucking leech optimizer; GNDO, generalized normal distribution optimization; IBSA, improved backtracking search algorithm.

Table 7: The best solution of IBSA and the reported solutions in solving the welded beam design problem.

Algorithm	x_1	x_2	x_3	x_4	$f(x^*)$
DBA [39]	0.20491	3.483873	9.047859	0.20568	1.726943
AQODSBKA [40]	0.2057	3.4712	9.0366	0.2057	1.7251
QOBLBKA [40]	0.2119	3.4411	8.9426	0.2149	1.7593
GSO [41]	0.205726	3.470329	9.037232	0.205728	1.72491
IBSA	0.20572964	3.47048867	9.03662391	0.20572964	1.72485231

Note: DBA, detective behavior algorithm; AQODSBKA, adaptive quasi-opposition and dynamic switching in black-winged kite algorithm; QOBLBKA, quasi-opposition black-winged kite algorithm; GSO, glider snake optimize; IBSA, improved backtracking search algorithm; x^* , the optimal solution; $f(x^*)$, the optimal value.

5.2 The Tension/Compression Spring Design Problem

For the tension/compression spring design problem, the objective function is given by:

$$f(x) = (x_3 + 2)x_2x_1^2 \quad (12)$$

The related constraints can be summarized as follows.

$$g_1(x) = 1 - \frac{x_2^3x_3}{71785x_1^4} \leq 0, \quad g_2(x) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} \leq 0, \quad (13a)$$

$$g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0, \quad g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0, \quad (13b)$$

where $0.05 \leq x_1 \leq 2$, $0.25 \leq x_2 \leq 1.3$, and $2.00 \leq x_3 \leq 15.0$.

The statistical results of IBSA and the compared algorithms in solving the tension/compression spring design problem have been shown in Table 8. According to Table 8, IBSA, BSLO, and GNDO are capable of finding the optimal solution, which is 0.012665. The optimal solution of JAYA is 0.012667. That is, the optimal solution obtained by JAYA is very close to those obtained by IBSA, BSLO and GNDO. The optimal solutions from BSA and SCA are significantly inferior to those of other algorithms. Table 9 lists the optimal parameter values that correspond to the best solutions obtained by IBSA and the competing algorithms. Meanwhile, Fig. 3 illustrates the convergence behavior of IBSA and its rival methods on the tension/compression spring design problem. As the numerical solutions obtained by IBSA and the comparison algorithm differ very little, as shown in Tables 8 and 9, the convergence curves are very similar in Fig. 3. Table 10 shows a comparison between the best solution of IBSA and those reported in previous studies. As shown in Table 10, IBSA outperforms the four algorithms it was compared with. The optimal solution of IBSA is 0.012665, whereas the optimal solution of DBSA and ETO are 0.0126658 and 0.012666338, respectively. In other words, the solutions of DBSA and ETO are very close to that of IBSA. The solutions of the remaining algorithms are significantly inferior to that of IBSA.

Table 8: The statistical solutions obtained by IBSA and the compared algorithms in solving the tension/compression spring design problem.

Algorithm	WORST	MEAN	MEDIAN	BEST	STD
JAYA	0.012725	0.012697	0.012694	0.012667	2.07E−05
BSA	0.012719	0.012702	0.012705	0.012675	1.74E−05
SCA	0.013147	0.012949	0.012953	0.012753	1.03E−04
BSLO	0.017747	0.014035	0.012791	0.012665	1.93E−03
GNDO	0.012665	0.012665	0.012665	0.012665	7.42E−14
IBSA	0.013623	0.012788	0.012701	0.012665	2.06E−04

Note: JAYA, Jaya algorithm; BSA, backtracking search algorithm; SCA, sine cosine algorithm; BSLO, blood-sucking leech optimizer; GNDO, generalized normal distribution optimization; IBSA, improved backtracking search algorithm; BEST, the best solution; MEAN, mean solution; STD, standard deviation; WORST, the worst solution; MEDIAN, the median solution.

Table 9: The best solutions obtained by IBSA and the compared algorithms in solving the tension/compression spring design problem.

Algorithm	x_1	x_2	x_3	$f(x^*)$
JAYA	0.051674	0.356351	11.312108	0.012667
BSA	0.051959	0.363176	10.927569	0.012675
SCA	0.050000	0.317312	14.076199	0.012753
BSLO	0.051716	0.357372	11.250735	0.012665
GNDO	0.051689	0.356718	11.288966	0.012665
IBSA	0.051705	0.357104	11.266455	0.012665

Note: JAYA, Jaya algorithm; BSA, backtracking search algorithm; SCA, sine cosine algorithm; BSLO, blood-sucking leech optimizer; GNDO, generalized normal distribution optimization; IBSA, improved backtracking search algorithm; x^* , the optimal solution; $f(x^*)$, the optimal value.

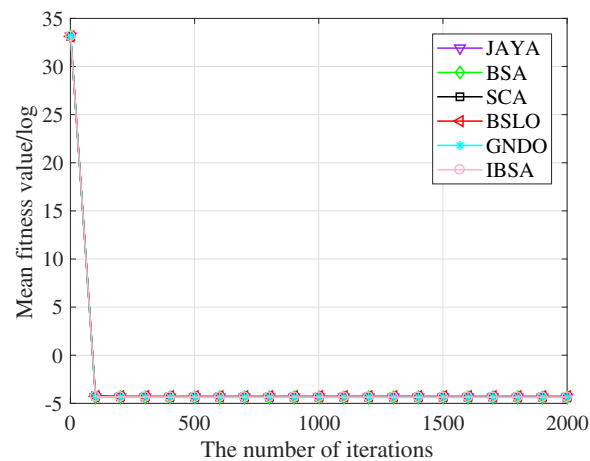
**Figure 3:** Convergence behavior for the tension/compression spring design problem; JAYA, Jaya algorithm; BSA, backtracking search algorithm; SCA, sine cosine algorithm; BSLO, blood-sucking leech optimizer; GNDO, generalized normal distribution optimization; IBSA, improved backtracking search algorithm.

Table 10: The best solution of IBSA and the reported solutions in solving the tension/compression spring design problem.

Algorithm	x_1	x_2	x_3	$f(x^*)$
AOO [42]	0.0515	0.3530	11.5202	0.01268
DBSA [39]	0.051683	0.356565	11.297842	0.0126658
TJO [43]	0.05109	0.34252	12.18639	0.01269
ETO [44]	0.051443	0.350838	11.642213	0.012666338
IBSA	0.051705	0.357104	11.266455	0.012665

Note: AOO, animated oat optimization; DBSA, detective behavior algorithm; TJO, traffic jam optimizer; ETO, expansion-trajectory optimization; IBSA, improved backtracking search algorithm; x^* , the optimal solution; $f(x^*)$, the optimal value.

5.3 The Pressure Vessel Design Problem

For the pressure vessel design problem, the objective function is given by:

$$f(x) = 0.6224x_1x_3x_4 + 1.7881x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \quad (14)$$

The related constraints can be summarized as follows.

$$g_1(x) = -x_1 + 0.0193x_3 \leq 0, g_2(x) = -x_2 + 0.00954x_3 \leq 0, \quad (15a)$$

$$g_3(x) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 129600 \leq 0, g_4(x) = x_4 - 240 \leq 0, \quad (15b)$$

where $0 \leq x_1 \leq 99$, $0 \leq x_2 \leq 99$, $10 \leq x_3 \leq 200$, and $10 \leq x_4 \leq 200$.

The statistical solutions produced by IBSA and the compared algorithms in solving the pressure vessel design problem have been displayed in Table 11. By observing Table 11, IBSA, JAYA, and GNDO can obtain the best solution, which is 5885.332774. The optimal solutions of BSA, BSLO are 5885.334082 and 5885.343329, respectively. Clearly, the optimal solutions obtained by BSA and BSLO are very close to those obtained by IBSA, JAYA, and GNDO. The optimal solution from SCA is 6021.688551. That is, the best solution of SCA is significantly inferior to those of other algorithms. Table 12 presents the optimal parameters corresponding to the best solutions obtained by IBSA and the comparative algorithms in solving the pressure vessel design problem. Fig. 4 displays the convergence trajectories of IBSA and the competing algorithms when tackling the pressure vessel design problem. From Fig. 4, SCA and BSLO are significantly inferior to JAYA, BSA, GNDO, and IBSA. Moreover, the convergence performance of JAYA, BSA, GNDO, and IBSA is relatively similar. Table 13 presents the optimal solutions obtained by IBSA and the reported algorithms for the pressure vessel design problem. From this table, it is clear that the solutions from the five competing algorithms are markedly worse than that of IBSA.

Table 11: The statistical solutions of IBSA and the baseline methods when tackling the pressure vessel design problem.

Algorithm	WORST	MEAN	MEDIAN	BEST	STD
JAYA	5885.332878	5885.332777	5885.332774	5885.332774	1.90E−05
BSA	5886.465362	5885.446409	5885.365665	5885.334082	2.28E−01
SCA	7737.253909	6570.872754	6511.709108	6021.688551	3.94E+02
BSLO	7319.000666	6389.809448	6207.057196	5885.343329	4.91E+02
GNDO	5885.332774	5885.332774	5885.332774	5885.332774	5.07E−13
IBSA	6219.167777	5918.883428	5885.441888	5885.332774	7.41E+01

Note: JAYA, Jaya algorithm; BSA, backtracking search algorithm; SCA, sine cosine algorithm; BSLO, blood-sucking leech optimizer; GNDO, generalized normal distribution optimization; IBSA, improved backtracking search algorithm; BEST, the best solution; MEAN, mean solution; STD, standard deviation; WORST, the worst solution; MEDIAN, the median solution.

Table 12: The best solutions of IBSA and the baseline methods when tackling the pressure vessel design problem.

Algorithm	x_1	x_2	x_3	x_4	$f(x^*)$
JAYA	0.778169	0.384649	40.319619	200.000000	5885.332774
BSA	0.778169	0.384649	40.319620	200.000000	5885.334082
SCA	0.794010	0.412230	41.092874	191.444819	6021.688551
BSLO	0.778175	0.384652	40.319938	199.995550	5885.343329
GNDO	0.778169	0.384649	40.319619	200.000000	5885.332774
IBSA	0.778169	0.384649	40.319619	200.000000	5885.332774

Note: JAYA, Jaya algorithm; BSA, backtracking search algorithm; SCA, sine cosine algorithm; BSLO, blood-sucking leech optimizer; GNDO, generalized normal distribution optimization; IBSA, improved backtracking search algorithm; x^* , the optimal solution; $f(x^*)$, the optimal value.

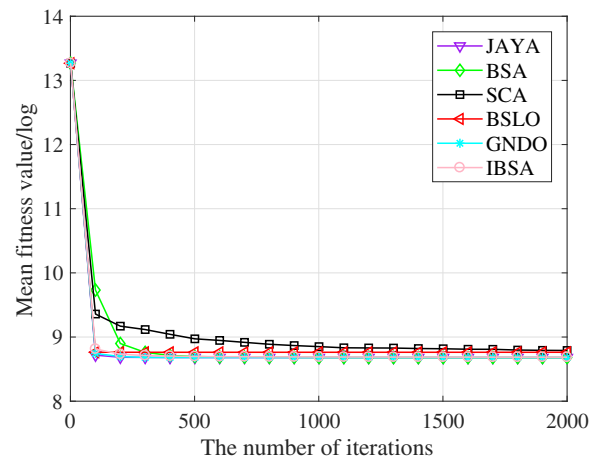
**Figure 4:** Convergence behavior for the pressure vessel design problem; JAYA, Jaya algorithm; BSA, backtracking search algorithm; SCA, sine cosine algorithm; BSLO, blood-sucking leech optimizer; GNDO, generalized normal distribution optimization; IBSA, improved backtracking search algorithm.

Table 13: The best solution of IBSA and the reported solutions in solving the pressure vessel design problem.

Algorithm	x_1	x_2	x_3	x_4	$f(x^*)$
AOO [42]	0.7792	0.3866	40.3645	199.4199	5893.2564
DBSA [39]	0.779915	0.385796	40.400294	198.918863	5891.30258
TJO [43]	13.1138	7.1123	42.0984	176.6384	6059.7450
ETO [44]	13.501686	7.628282	42.098445	176.636598	6059.714353
NDE [45]	0.8125	0.4375	42.0984455958	176.63659584330	6059.714335
IBSA	0.778169	0.384649	40.319619	200.000000	5885.332774

Note: AOO, animated oat optimization; DBSA, detective behavior algorithm; TJO, traffic jam optimizer; ETO, expansion-trajectory optimization; NDE, novel differential evolution; IBSA, improved backtracking search algorithm; x^* , the optimal solution; $f(x^*)$, the optimal value.

5.4 The Speed Reducer Design Problem

For the speed reducer design problem, the objective function is given by:

$$f(\mathbf{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2) \quad (16)$$

The related constraints can be summarized as follows.

$$f_{\text{cost}}(\mathbf{x}) = 0.7854x_1x_2^2(3.3x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.477(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2) \quad (17)$$

where $2.6 \leq x_1 \leq 3.6$, $0.7 \leq x_2 \leq 0.8$, $17 \leq x_3 \leq 28$, $7.3 \leq x_4 \leq 8.3$, $7.8 \leq x_5 \leq 8.3$, $2.9 \leq x_6 \leq 3.9$, and $5.0 \leq x_7 \leq 5.5$.

Table 14 presents the statistical solutions obtained by IBSA and the compared algorithms in solving the speed reducer design problem. As can be seen from Table 14, IBSA, JAYA, BSA, and GNDO can obtain the best solution, which is 2983.255664. The optimal solution achieved by SCA is 3030.759751. Clearly, the best solution of SCA is significantly inferior to those of other algorithms. Table 15 gives the optimal parameter settings that correspond to the best solutions of IBSA and the compared algorithms for the speed reducer design problem. Meanwhile, Fig. 5 illustrates the convergence trajectories of IBSA and its competitors when tackling the same problem. As shown in Fig. 5, given that the numerical solutions obtained by IBSA and the comparative algorithm in addressing speed reducer design problems are quite close, their convergence performance is therefore very similar. Table 16 presents a comparison between the optimal solution of IBSA and those from previously reported algorithms on the speed reducer design problem. From this table, IBSA is clearly superior to the competing methods.

Table 14: The statistical solutions of IBSA and the baseline methods when tackling the speed reducer design problem.

Algorithm	WORST	MEAN	MEDIAN	BEST	STD
JAYA	2983.255664	2983.255664	2983.255664	2983.255664	0.00E+00
BSA	3022.162714	2985.849468	2983.255664	2983.255664	9.87E+00
SCA	3118.099308	3065.130166	3059.558467	3030.759751	2.45E+01
BSLO	2983.255688	2983.255670	2983.255668	2983.255666	4.80E−06
GNDO	2983.255664	2983.255664	2983.255664	2983.255664	0.00E+00
IBSA	2983.255664	2983.255664	2983.255664	2983.255664	0.00E+00

Note: JAYA, Jaya algorithm; BSA, backtracking search algorithm; SCA, sine cosine algorithm; BSLO, blood-sucking leech optimizer; GNDO, generalized normal distribution optimization; IBSA, improved backtracking search algorithm; BEST, the best solution; MEAN, mean solution; STD, standard deviation; WORST, the worst solution; MEDIAN, the median solution.

Table 15: The best solutions of IBSA and the baseline methods when tackling the speed reducer design problem.

Algorithm	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$f(x^*)$
JAYA	3.500000	0.700000	17.000000	7.300000	7.800000	3.350215	5.286683	2983.255664
BSA	3.500000	0.700000	17.000000	7.300000	7.800000	3.350215	5.286683	2983.255664
SCA	3.600000	0.700000	17.000000	7.300000	7.800000	3.379893	5.288256	3030.759751
BSLO	3.500000	0.700000	17.000000	7.300000	7.800000	3.350215	5.286683	2983.255666
GNDO	3.500000	0.700000	17.000000	7.300000	7.800000	3.350215	5.286683	2983.255664
IBSA	3.500000	0.700000	17.000000	7.300000	7.800000	3.350215	5.286683	2983.255664

Note: JAYA, Jaya algorithm; BSA, backtracking search algorithm; SCA, sine cosine algorithm; BSLO, blood-sucking leech optimizer; GNDO, generalized normal distribution optimization; IBSA, improved backtracking search algorithm; x^* , the optimal solution; $f(x^*)$, the optimal value.

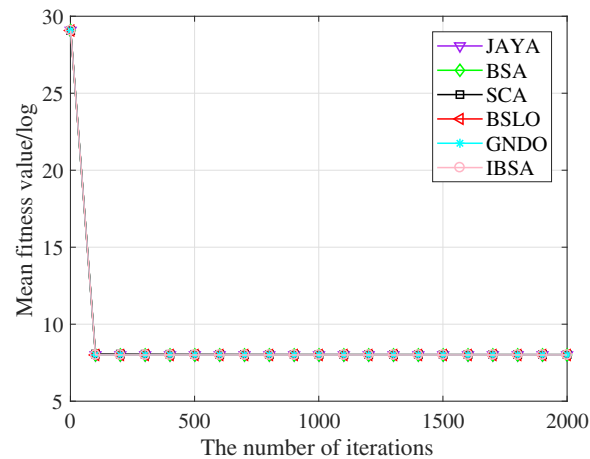
**Figure 5:** Convergence behavior for the speed reducer design problem; JAYA, Jaya algorithm; BSA, backtracking search algorithm; SCA, sine cosine algorithm; BSLO, blood-sucking leech optimizer; GNDO, generalized normal distribution optimization; IBSA, improved backtracking search algorithm.

Table 16: The best solution of IBSA and the reported solutions in solving the speed reducer design problem.

Algorithm	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$f(x^*)$
LSO [46]	3.5000	0.7	17.0	7.6707	7.7192	3.3509	5.2866	2998.008
CAO [47]	3.500000	0.70	17	7.300000	7.7153199	3.350215	5.286654	2994.471
AGOA [48]	3.500000	0.700000	17.000000	7.300000	7.715320	3.350541	5.286654	2994.424466
AOO [42]	3.5000	0.7000	17.0000	7.3125	7.7313	3.3503	5.2867	2994.9764
ETO [44]	3.5	0.7	17.2074	7.3	7.7154	3.3502	5.2867	2994.47292
IBSA	3.500000	0.700000	17.000000	7.300000	7.800000	3.350215	5.286683	2983.255664

Note: LSO, latin sampling and crossover; CAO, competition-amensalism optimization; AGOA, adversarial game optimization algorithm; AOO, animated oat optimization; ETO, expansion-trajectory optimization; IBSA, improved backtracking search algorithm; x^* , the optimal solution; $f(x^*)$, the optimal value.

When addressing practical engineering problems, compared to other statistical indicators, more attention should be paid to obtaining the optimal solutions. Based on the comparison results mentioned above, IBSA has demonstrated outstanding performance in obtaining the optimal solutions to these engineering problems, which can be described as follows:

- IBSA is able to achieve the same optimal solutions as some comparative algorithms in certain engineering problems. Although IBSA attains the same optimal solutions as some comparative algorithms, this conclusion is still meaningful: it indicates that when addressing the same engineering problem, IBSA provides an alternative perspective, making it a viable option for similar engineering problems.
- IBSA can achieve solutions slightly better than the comparison algorithms in some engineering problems. This conclusion is also meaningful: for engineering problems, even minor improvements can result in significant changes in engineering quality and economic benefits.

6 Conclusion

This paper aims to propose a new variant of BSA, namely IBSA, for solving complex optimization problems. Compared to BSA, the significant feature of IBSA is the introduction of two dynamic elite populations into the search strategy. The characteristics of these two dynamic elite populations are as follows: one updates with the renewal of individuals; One is to update with the update of the optimal solution obtained by each iteration of the population. This improvement mechanism can significantly enhance the directionality of the population during the search process, thereby improving the search quality of the population. The performance of IBSA has been validated in the classic CEC 2014 test suite and four constrained engineering optimization problems. IBSA outperforms the competing algorithms in overall performance, which also validates the efficacy of the proposed improvement strategy. In addition, we also study the performance of IBSA in the CEC 2017 test set, with further details available in the supplementary material (Supplementary Tables S1 and S2).

It is worth mentioning that among the four engineering design problems in the main text, IBSA achieved outstanding optimization performance. However, this does not imply that IBSA can demonstrate superior performance in solving other engineering problems. This is mainly because the objective functions, decision variables, and constraints of each practical engineering problem are complex and variable. Therefore, the performance of IBSA in solving engineering problems should not be overstated. Encouraged by this, future efforts will attempt to use IBSA to address more complex optimization problems in order to expand the applications of IBSA, such as unmanned aerial vehicle path optimization problems during large-scale Internet of Things data collection and robot path planning problems in complex environments.

Acknowledgement: Not applicable.

Funding Statement: The project was supported in part by the Special Fund of the Taishan Scholar Program under Grant tsqn202408187, and in part by the Shandong Provincial Natural Science Foundation under Grant ZR2025QC1548.

Availability of Data and Materials: The data and materials used to support the findings of this study are available from the corresponding author upon reasonable request.

Ethics Approval: Not applicable.

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

Supplementary Materials: The supplementary material is available online at <https://www.techscience.com/doi/10.32604/cmc.2026.087316/sl>.

References

1. Abdel-Basset M, Mohamed R, Azeem SAA, Jameel M, Abouhawwash M. Kepler optimization algorithm: a new metaheuristic algorithm inspired by Kepler's laws of planetary motion. *Knowl-Based Syst.* 2023;268:110454.
2. Yang XS, Deb S. Cuckoo search via Lévy flights. In: *Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*; 2009 Dec 9–11; Coimbatore, India. p. 210–4.
3. Mirjalili S, Lewis A. The whale optimization algorithm. *Adv Eng Softw.* 2016;95(12):51–67. doi:10.1016/j.advengsoft.2016.01.008.
4. Rao RV, Savsani VJ, Vakharia DP. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Comput Des.* 2011;43(3):303–15.
5. Zhang Y, Jin Z. Group teaching optimization algorithm: a novel metaheuristic method for solving global optimization problems. *Expert Syst Appl.* 2020;148:113246.
6. Yu M, Yang H, Zhang J, Ouyang K, Fu S, Tan P, et al. Bounty hunter optimizer: a novel metaheuristic with an application to multi-UAV mobile edge computing and path planning. *Knowl-Based Syst.* 2026;341:115836.
7. Sadollah A, Sayyaadi H, Yadav A. A dynamic metaheuristic optimization model inspired by biological nervous systems: neural network algorithm. *Appl Soft Comput.* 2018;71(2):747–82. doi:10.1016/j.asoc.2018.07.039.
8. Simon D. Biogeography-based optimization. *IEEE Trans Evol Comput.* 2008;12(6):702–13. doi:10.1109/tevc.2008.919004.
9. Hatamlou A. Black hole: a new heuristic optimization approach for data clustering. *Inf Sci.* 2013;222:175–84.
10. Ghorbani N, Babaei E. Exchange market algorithm. *Appl Soft Comput.* 2014;19(1):177–87. doi:10.1016/j.asoc.2014.02.006.
11. Rashedi E, Nezamabadi-pour H, Saryazdi S. GSA: a gravitational search algorithm. *Inf Sci.* 2009;179(13):2232–48.
12. Hashim FA, Houssein EH, Hussain K, Mabrouk MS, Al-Atabany W. Honey badger algorithm: new metaheuristic algorithm for solving optimization problems. *Math Comput Simul.* 2022;192(2):84–110. doi:10.1016/j.matcom.2021.08.013.
13. Mohamed AW, Hadi AA, Mohamed AK. Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm. *Int J Mach Learn Cybern.* 2020;11(7):1501–29. doi:10.1007/s13042-019-01053-x.
14. Storn R, Price K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J Glob Optim.* 1997;11(4):341–59. doi:10.1023/a:1008202821328.
15. Mohamed AW. RDEL: restart differential evolution algorithm with local search mutation for global numerical optimization. *Egypt Inform J.* 2014;15(3):175–88.
16. Zhong L, Wang Y, Wang K, Sui Q, Lei Z, Gao S. Binary tree population structure-based differential evolution. *IEEE Trans Emerg Top Comput Intell.* 2025;9(6):3849–61. doi:10.1109/tetci.2025.3551940.
17. Wang K, Gao S, Zhou MC, Zhan ZH, Cheng J. Fractional order differential evolution. *IEEE Trans Evol Comput.* 2024;29(3):822–35. doi:10.1109/tevc.2024.3382047.

18. Civicioglu P. Backtracking search optimization algorithm for numerical optimization problems. *Appl Math Comput.* 2013;219(15):8121–44. doi:10.1016/j.amc.2013.02.017.
19. Guney K, Durmus A, Basbug S. Backtracking search optimization algorithm for synthesis of concentric circular antenna arrays. *Int J Antennas Propag.* 2014;2014(1):250841. doi:10.1155/2014/250841.
20. Jin YF, Yin ZY. Enhancement of backtracking search algorithm for identifying soil parameters. *Int J Numer Anal Methods Geomech.* 2020;44(9):1239–61. doi:10.1002/nag.3059.
21. Zou F, Chen D, Li S, Lu R, Lin M. Community detection in complex networks: multi-objective discrete backtracking search optimization algorithm with decomposition. *Appl Soft Comput.* 2017;53(7127):285–95. doi:10.1016/j.asoc.2017.01.005.
22. Guney K, Durmus A. Pattern nulling of linear antenna arrays using backtracking search optimization algorithm. *Int J Antennas Propag.* 2015;2015(1):713080. doi:10.1155/2015/713080.
23. Shaheen AM, El-Sehiemy RA, Farrag SM. Integrated strategies of backtracking search optimizer for solving reactive power dispatch problem. *IEEE Syst J.* 2018;12(1):424–33. doi:10.1109/jsyst.2016.2573799.
24. Ahandani MA, Ghiasi AR, Kharrati H. Parameter identification of chaotic systems using a shuffled backtracking search optimization algorithm. *Soft Comput.* 2018;22(24):8317–39. doi:10.1007/s00500-017-2779-0.
25. Wang Z, Zeng YR, Wang S, Wang L. Optimizing echo state network with backtracking search optimization algorithm for time series forecasting. *Eng Appl Artif Intell.* 2019;81(3):117–32. doi:10.1016/j.engappai.2019.02.009.
26. Wang L, Peng L, Wang S, Liu S. Advanced backtracking search optimization algorithm for a new joint replenishment problem under trade credit with grouping constraint. *Appl Soft Comput.* 2020;86(5):105953. doi:10.1016/j.asoc.2019.105953.
27. Hu Z, Dai C, Su Q. Adaptive backtracking search optimization algorithm with a dual-learning strategy for dynamic economic dispatch with valve-point effects. *Energy.* 2022;248(1):123558. doi:10.1016/j.energy.2022.123558.
28. Zhang Y. Elite archives-driven particle swarm optimization for large scale numerical optimization and its engineering applications. *Swarm Evol Comput.* 2023;76(6):101212. doi:10.1016/j.swevo.2022.101212.
29. Zhao S, Wang D. Elite-ordinary synergistic particle swarm optimization. *Inf Sci.* 2022;609(3):1567–87. doi:10.1016/j.ins.2022.07.131.
30. Yang Q, Yan JQ, Gao XD, Xu DD, Lu ZY, Zhang J. Random neighbor elite guided differential evolution for global numerical optimization. *Inf Sci.* 2022;607(3):1408–38. doi:10.1016/j.ins.2022.06.029.
31. Wang L, Huang C. A novel elite opposition-based Jaya algorithm for parameter estimation of photovoltaic cell models. *Optik.* 2018;155(7):351–6. doi:10.1016/j.jllo.2017.10.081.
32. Deng S, Liu H, Zhu W, Li K, Li M. Elite neighborhood search and dynamic feature probability-guided crow search algorithm for high-dimensional feature selection. *Expert Syst Appl.* 2025;274(3):127012. doi:10.1016/j.eswa.2025.127012.
33. Liang JJ, Qu BY, Suganthan PN. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. Zhengzhou, China: Computational Intelligence Laboratory, Zhengzhou University; Singapore: Nanyang Technological University; 2013. Technical Report 210311.
34. Rao RV, Waghmare GG. A new optimization algorithm for solving complex constrained design optimization problems. *Eng Optim.* 2017;49(1):60–83. doi:10.1080/0305215x.2016.1164855.
35. Mirjalili S. SCA: a sine cosine algorithm for solving optimization problems. *Knowledge-Based Syst.* 2016;96:120–33.
36. Bai J, Nguyen-Xuan H, Atroshchenko E, Kosec G, Wang L, Wahab MA. Blood-sucking leech optimizer. *Adv Eng Softw.* 2024;195:103696. doi:10.1016/j.advengsoft.2024.103696.
37. Zhang Y, Jin Z, Mirjalili S. Generalized normal distribution optimization and its applications in parameter extraction of photovoltaic models. *Energy Convers Manag.* 2020;224(9):113301. doi:10.1016/j.enconman.2020.113301.
38. She YX. Engineering optimization: an introduction with metaheuristic applications. Hoboken, NJ, USA: John Wiley & Sons, Inc.; 2010.
39. Cheng J, Waele DW. Detective behavior algorithm (DBA): a new metaheuristic for design and engineering optimization. *Knowledge-Based Syst.* 2026;338:115434.

40. Rajasekar P, Jayalakshmi M. Adaptive quasi-opposition and dynamic switching in black-winged kite algorithm for global optimization and constrained engineering designs. *Alex Eng J.* 2025;130(1):969–94. doi:10.1016/j.aej.2025.09.058.
41. El-kenawy ESM, Khodadadi N, Mirjalili S, Zaki AM, Ibrahim A, Alhussan AA, et al. Glider snake optimizer (GSO): a nature-inspired metaheuristic algorithm for global and engineering optimization problems. *Artif Intell Rev.* 2026;59(3):91. doi:10.1007/s10462-026-11504-x.
42. Wang RB, Hu RB, Geng FD, Xu L, Chu SC, Pan JS, et al. The animated oat optimization algorithm: a nature-inspired metaheuristic for engineering optimization and a case study on wireless sensor networks. *Knowl-Based Syst.* 2025;318:113589.
43. Wang J, Shang Z. Traffic jam optimizer: a novel swarm-based metaheuristic algorithm for solving global optimization problems. *Appl Math Model.* 2026;150:116410.
44. Cuevas E, González-Sánchez OA, Jiménez OF, Zaldívar D, Rodríguez-Vazquez A, Sarkar R. Expansion-trajectory optimization (ETO): a dual-operator metaheuristic for balanced global and local search. *Appl Soft Comput.* 2025;183:113642.
45. Mohamed AW. A novel differential evolution algorithm for solving constrained engineering optimization problems. *J Intell Manuf.* 2018;29(3):659–92. doi:10.1007/s10845-017-1294-6.
46. Cuevas E, González-Sánchez OA, Escobar H, Ayala E, Zaldívar D, Pérez-Cisneros M, et al. Overcoming center-bias behavior: a metaheuristic algorithm with dual operators for optimized search and refinement. *Syst Soft Comput.* 2026;8:200436.
47. Yilma AA, Woldegiorgis BH. Competition-amensalism optimization: a novel metaheuristic optimization algorithm. *Results Eng.* 2026;30:111048.
48. Li C, Zhang Q, Li J, Tao S, Oliva D. Adversarial game optimization: a game-theoretic metaheuristic for efficient complex optimization and engineering applications. *Inf Sci.* 2026;735:123022.