Computer Modeling in
Engineering & Sciences

Tech Science Press

Doi:10.32604/cmes.2025.068628

ARTICLE

A Novel Approach Based on Recuperated Seed Search Optimization for Solving Mechanical Engineering Design Problems

Sumika Chauhan[1], **Govind Vashishtha**[1,*], **Riya Singh**[2] and **Divesh Bharti**[3]

[1]Faculty of Geoengineering, Mining and Geology, Wroclaw University of Science and Technology, Na-Grobli 15, Wroclaw, 50-421, Poland
[2]Department of Mechanical Engineering, GLA University, Mathura, 281406, India
[3]Precision Metrology Laboratory, Department of Mechanical Engineering, Sant Longowal Institute of Engineering and Technology, Punjab, 148106, India

*Corresponding Author: Govind Vashishtha. Email: govindyudivashishtha@gmail.com or govind.vashishtha@pwr.edu.pl

Received: 02 June 2025; Accepted: 02 July 2025; Published: 31 July 2025

ABSTRACT: This paper introduces a novel optimization approach called Recuperated Seed Search Optimization (RSSO), designed to address challenges in solving mechanical engineering design problems. Many optimization techniques struggle with slow convergence and suboptimal solutions due to complex, nonlinear natures. The Sperm Swarm Optimization (SSO) algorithm, which mimics the sperm's movement to reach an egg, is one such technique. To improve SSO, researchers combined it with three strategies: opposition-based learning (OBL), Cauchy mutation (CM), and position clamping. OBL introduces diversity to SSO by exploring opposite solutions, speeding up convergence. CM enhances both exploration and exploitation capabilities throughout the optimization process. This combined approach, RSSO, has been rigorously tested on standard benchmark functions, real-world engineering problems, and through statistical analysis (Wilcoxon test). The results demonstrate that RSSO significantly outperforms other optimization algorithms, achieving faster convergence and better solutions. The paper details the RSSO algorithm, discusses its implementation, and presents comparative results that validate its effectiveness in solving complex engineering design challenges.

KEYWORDS: Local search; Cauchy mutation; opposition-based learning; exploration; exploitation

1 Introduction

Nature-inspired algorithms have gained significant popularity among researchers and scientists for addressing a wide range of challenges in engineering fields, including medical image processing, signal processing, cloud computing, feature selection, deep learning, text mining, photovoltaic models, and urban planning [1]. These algorithms are categorized into swarm intelligence (SI), physics-based, and evolutionary algorithms (EA). Notably, swarm-based optimization algorithms are renowned for their ability to mimic the social behaviour of organisms found in nature [2,3].

Swarm-based optimization algorithms excel in solving real-world optimization problems due to several key characteristics: (i) The search process mimics natural phenomena, which are inherently random. This randomness enables swarm-based optimization algorithms to avoid getting trapped in local optima; (ii) these algorithms, also known as population-based optimization, explore all potential solutions by updating the positions of individuals within the search space to achieve the global optimal solution. Notable swarm-based optimization algorithms include the particle swarm optimizer (PSO) [4], arithmetic optimization algorithm

(AOA) [5], ant lion optimizer (ALO) [6], dragonfly algorithm (DA) [7], grey wolf optimization (GWO) [8], moth-flame optimization (MFO) [9], multi-verse optimization (MVO) [10], salp swarm algorithm (SSA) [11], rabbits optimization algorithm (ROA) [12] and sine cosine algorithm (SCA) [11]. Sperm swarm optimization, a novel approach proposed by Shehadeh et al. [13], emulates the movement of sperm or seed during fertilization.

In recent years, researchers have been focused on developing innovative methods to enhance the performance of swarm-based optimization algorithms. These mechanisms include the hybridization of two algorithms, the incorporation of an opposition-based concept, the addition of various mutation strategies, and the application of weightage during position updating. For example, Wang et al. [14] improved kill herd optimization by integrating OBL and heavy-tailed CM to boost the convergence rate of the basic kill herd. Kumari et al. [15] enhanced the chimp optimization algorithm by incorporating elements from the spotted hyena optimizer, aiming to improve both the exploration and exploitation phases of the optimization process. Kaucic et al. [16] tackled the limitations of basic optimization techniques for high-dimensional problems by combining the level-based learning swarm optimizer (LLSO) with PSO. Alruwais et al. [17] hybridized the moth flame optimization (MFO) with deep learning concepts for automatic fabric inspection. Zhou et al. [18] refined the slime mould algorithm (SMA) by incorporating mutation and neighborhood search strategies, enhancing both exploration and exploitation capabilities, especially for complex combinatorial functions. Yu et al. [19] integrated grey wolf optimization (GWO) and differential evolution (DE) to improve UAV path planning. Vashishtha and Kumar [20] utilized AO to determine the optimal filter length for minimum entropy deconvolution (MED), a technique used to diagnose bearing defects in Francis turbines. Chauhan et al. [21] explored various mutation strategies within a diversity-driven multi-parent evolutionary algorithm to address area coverage issues in wireless sensor networks. Wang et al. [22] enhanced the Golden Jackal Optimization algorithm with multi-strategy mixing, incorporating chaotic initialization, dynamic inertia weight, and Gaussian mutation. It outperforms existing methods on benchmark functions and industrial problems, effectively balancing exploration and exploitation, avoiding local optima, and demonstrating high robustness and applicability for complex real-world optimization challenges.

Swarm-based optimization algorithms offer numerous advantages, yet no single algorithm can efficiently and effectively solve all optimization problems. This notion is supported by the no-free-lunch (NFL) theorem, which underscores the importance of developing innovative and unique algorithms for different optimization challenges. This insight has inspired a modification to the existing sperm swarm optimization (SSO), which mimics the movement of sperm toward fertilizing an egg. While the basic SSO is adept at handling a range of optimization problems, it does face limitations, such as slow convergence and a tendency to get trapped in local optima when addressing high-dimensional problems. To tackle these issues, OBL and CM strategies have been integrated into the basic SSO. The OBL enhances the diversity of SSO solutions by generating a new solution that is the opposite of the current one, thereby accelerating convergence. Meanwhile, the CM strategy balances the exploration and exploitation search capabilities.

The quality of solutions and the convergence rate of the basic SSO were improved by competitively incorporating the benefits of OBL and CM operators throughout the exploratory phase. The innovations of this study are summarized as follows:

1. The basic SSO was refined using OBL and CM strategies to boost its performance. This enhanced algorithm is called recuperated seed swarm optimization (RSSO).
2. The fundamental SSO was enhanced through the application of OBL and CM strategies to improve its performance. This upgraded algorithm is known as recuperated seed swarm optimization (RSSO).

3. To demonstrate its effectiveness, RSSO was compared with other popular optimization algorithms developed in recent years. This comparison highlights RSSO's advantages and showcases its potential to outperform existing methods.

The rest of this paper is organized as follows. Section 2 provides the background information for this research. Section 3 describes the proposed RSSO-based optimization algorithm. Section 4 presents experimental results, analysis, and discussion. Section 6 summarizes the key conclusions and findings of this study.

2 Background

2.1 Sperm Swarm Optimization (SSO)

Shehadeh et al. [13] introduced the SSO algorithm, inspired by the natural process of sperm fertilization. This algorithm mimics the journey of sperm as they move from the cooler cervix to the warmer fallopian tubes in search of the egg. In this optimization model, the 'sperm' or 'seed' seeks the 'egg' or 'ovum' within an optimal environment. Numerous seeds, or solutions, exist within the search space, but only one successfully fertilizes an egg, becoming the winner or global solution. However, this optimization faces certain limitations, such as the seeds not having a low pH level during fertilization. Therefore, it is recommended to maintain a pH value between 7 and 14 to ensure an alkaline and non-toxic environment for ovulation. This concept is represented by Eq. (1).

$$v_{ini} = DF \times v_{seed} \times log_{10}\left(ph\_r_1\right) \tag{1}$$

where the parameter DF represents the damping factor of velocity which is randomly generated in the range of [0, 1], the parameter v_{seed} is the velocity of the seed. And, the parameter ph_r_1 indicates the pH value in the range of 7 to 14.

The algorithm keeps track of the best solution found thus far, which is represented by the position of the most successful 'seed.' To identify the best seed, the algorithm constantly compares the current position of the best seed with its previously recorded position of the best seed. Thus, the current position replaces the previous position if the latest seed provides a superior solution to that of the last seed. The solution of the best seed can be obtained by Eq. (2).

$$S_{CB} = \log_{10}\left(ph\_r_2\right) \times \log_{10}\left(T\_r_1\right) \times \left(S_B - S_C\right) \tag{2}$$

where the parameter S_B indicates the best solution, ph_r_2 is the pH value of the visited area in the range of 7 to 14. Whereas the parameter T_r_1 randomly generated between 35.1 to 38.5 indicating the temperature of the visited area. The global best solution obtained from the seed can be computed using Eq. (3).

$$S_G = \log_1\left(ph\_r_3\right) \times \log_{10}\left(T\_r_2\right) \times \left(S_{GB} - S_C\right) \tag{3}$$

where the parameter S_{GB} represents the global best solution obtained by the seed. The parameter ph_r_3 indicates the pH value of the visited area generated in the range of 7 to 14 and the T_r_2 indicates the temperature of the visited area generated in the range of 35.1 to 38.5. The S_C represents the current solution of the seed which can be calculated through Eq. (4).

$$S_C = S_C + v_{seed} \tag{4}$$

where the velocity of the seed is represented by v_{seed} that can be obtained by Eq. (5).

$$v_{seed} = v_{ini} + S_{CB} + S_G \tag{5}$$

2.2 Cauchy Mutation (CM) Strategy

The algorithm uses a mathematical technique called Cauchy mutation to introduce randomness into the search process. Cauchy mutation is a type of probability distribution that depends on two factors: location (x) and scaling (t) [23,24]. These factors determine the shape of the Cauchy distribution, which is defined by the following density function: $f(x)$.

$$f(x) = \frac{1}{\pi}\left(\frac{t}{t^2 + x^2}\right), -\infty < x < \infty \tag{6}$$

where the parameter $t > 0$ is a scaling parameter. Whereas, Cauchy distribution function $F_t(x)$ is defined as:

$$F_t(x) = \frac{1}{2} + \frac{1}{\pi} arctan\left(\frac{x}{t}\right) \tag{7}$$

The CM operator increases the probability of not being trapped in a local optimum solution and overcomes premature convergence problems by performing small controlled steps in the search space. The solution of the Cauchy distributed random number $C(x)$ is computed by Eq. (7), whereas the mutation in the proposed methodology is performed through Eq. (8).

$$X_i^{t+1} = X_i + C(x) \times \Delta X_i^{t+1} \tag{8}$$

where ΔX_i^{t+1} is the step vector.

2.3 Opposition-Based Learning (OBL)

Every optimization algorithm starts by randomly choosing a starting point for the solution. The positions of the individual elements within the algorithm are then updated based on their 'intelligence' (meaning their ability to improve the overall solution). This means that the initial guess significantly affects the time required by the algorithm to find a solution. To improve the speed and efficiency, the algorithm can explore both the initial guess and its opposite. This implies calculating the solution based on the initial guess and then calculating the solution based on the opposite of that guess. A better solution of these two is then used to start the optimization process, which leads to faster convergence. This process of exploring both the initial guess and its opposite is repeated throughout the optimization process. Eq. (9) outlines the specific method used for initializing the solution within the algorithm.

$$x_{i+NP,j}^{t} = x_j^{min} + x_j^{max} - x_{ij}^{t}\,(i = 1, 2, \ldots, NP; j = 1, 2, \ldots, D), \tag{9}$$

where NP is the population size and D is the dimension of the search space.

3 Proposed Algorithm

The concept of OBL and CM has been incorporated in the basic SSO to develop RSSO whose pseudo-code is given in Algorithm 1. Along with the OBL and CM, the weighting factor has also been introduced while computing the current solution. The proposed algorithm is accomplished through the following steps which are also shown in Fig. 1.

Step 1. Initialization

The population is the RSSO has been initialized randomly through Eq. (10).

$$X_{ij} = X_j^{min} + r_{ij}\left(X_j^{max} + X_j^{min}\right); (i = 1, 2, \ldots, NP; j = 1, 2, \ldots, D) \tag{10}$$

where X_{ij} is the initial population having an upper bound X_j^{max} and lower bound X_j^{min}, respectively. The parameter r_{ij} is uniformly and randomly distributed number between [0, 1].

Step 2. Incorporating the OBL

The opposite solution has also been explored to improve the diversity in the population which increases the convergence rate. This step can be attained through Eq. (9).

Step 3. Position clamping in RSSO

The velocity of the "seed" (representing a potential solution) is crucial in finding the optimal solution within the RSSO algorithm, as outlined in Algorithm 1. To enhance the algorithm's exploration capabilities, a weighting factor has been introduced. The current position of the seed (S_C) is updated using Eq. (11) within the RSSO framework.

$$S_C = S_C \times 0.3 + v_{seed} \times 0.7 \tag{11}$$

Step 4. Utilizing the CM operator

Finding the optimal solution in any optimization algorithm requires a delicate balance between exploration (searching for new areas of the solution space) and exploitation (refining existing solutions). An effective balance helps to speed up the convergence process and prevents the algorithm from getting stuck in suboptimal solutions (local optima). The Cauchy Mutation (CM) operator has been incorporated into the algorithm to address this balance. Eq. (8) demonstrates how CM is implemented to achieve this balance.

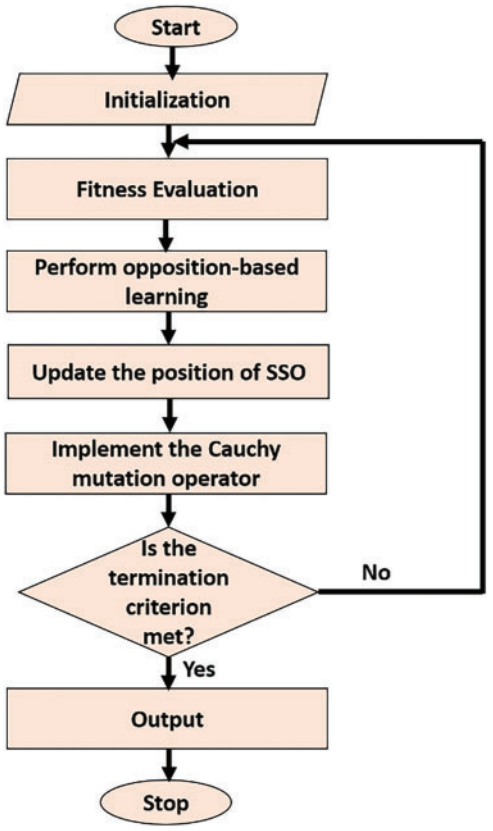
Start
Initialization
Fitness Evaluation
Perform opposition-based learning
Update the position of SSO
Implement the Cauchy mutation operator
Is the termination criterion met?
No
Yes
Output
Stop

Figure 1: Proposed RSSO algorithm

Algorithm 1: Proposed algorithm

```
Inputs: The population size (NP), maximum number of iterations (T), UB, LB.
Outputs: Optimum fitness value and the position of destination point
Step 1. Initialize the population randomly within search space
Step 2. Apply opposition best learning
Step 3. Evaluate fitness function for each search agent obtained from S2
Step 4. Sort the best fitness values from step 3 and obtained best NP members
While
    Step 5. For $i = 1: NP$
        Step 6. Evaluate fitness function for each swarm
            If Obtained fitness > best solution of the seed
                Assign the current value as the best solution of the seed
            end If
    end For
    Step 7. Assign the global best solution depends on the winner
    Step 8. For $i = 1: NP$
        Perform the velocity update rule update the position of the seed on the problem search space using Eq. (11)
    end for
    Step 9. Generate Cauchy mutation using Eq. (7).
    Step 10. Perform Cauchy mutation using Eq. (8).
    Step 11. $t = t + 1$.
Step 12. End While
Step 13. Return $X^*$
```

4 Results and Discussion

4.1 Classical Benchmark Functions

Three sets of classical benchmark functions viz., unimodal, multi-modal, and fixed-dimensional multi-modal have been used to evaluate the RSSO's performance. Applying the one best global solution of each approach to the unimodal benchmark functions F1–F7 improves their capabilities. Alternatively, optimizers can test their algorithms' diversification capabilities using fixed-dimension multi-modal benchmark functions F14–F23 and multimodal benchmark functions F8–F13. Tables 1–3 present the definitions of the three types of benchmark functions.

Table 1: Function definition for unimodal benchmark functions

No.	Type	Function	Dimension	Range	F_{min}
F1	US	$F(x) = \sum_{i=1}^{D} x_i^2$	30	[−100, 100]	0
F2	UN	$F(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	30	[−10, 10]	0
F3	UN	$F(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j^2\right)$	30	[−100, 100]	0
F4	US	$F(x) = \max_i \{\lvert x_i \rvert, 1 \le i \le D\}$	30	[−100, 100]	0
F5	UN	$F(x) = \sum_{i=1}^{D-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	30	[−30, 30]	0
F6	US	$F(x) = \sum_{i=1}^{D} \left(\lfloor x_i + 0.5 \rfloor\right)^2$	30	[−100, 100]	0

(Continued)

Table 1 (continued)

No.	Type	Function	Dimension	Range	F_{min}
F7	US	$F(x) = \sum_{i=1}^{D} ix_i^4 + \text{random}[0,1]$	30	[−1.28, 1.28]	0

Note: Where, US is uni-modal separable and UN represents uni-modal non-separable.

Table 2: Function definition for multi-modal benchmark functions

No.	Type	Function	Dimension	Range	F_{min}
F8	MS	$F(x) = -\sum_{i=1}^{D}\left(x_i \sin\sqrt{\lvert x_i\rvert}\right)$	10	[−500, 500]	−418.9829 ∗ D
F9	MS	$F(x) = 10D + \sum_{i=1}^{D}\left(x_i^2 - 10\cos(2\pi x_i)\right)$	30	[−5.12, 5.12]	0
F10	MN	$F(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	30	[−32, 32]	0
F11	MN	$F(x) = 1/4000\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[−600, 600]	0
F12	MN	$F(x) = \frac{\pi}{D}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2 \left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_D - 1)^2\right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$, where $y_i = 1 + \frac{x_i+1}{4}$, and $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m; x_i > a \\ 0; -a < x_i < a \\ k(-x_i - a)^m; x_i < -a \end{cases}$	30	[−50, 50]	0
F13	MN	$F(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{D}(x_i - 1)^2 \left[1 + \sin^2(3\pi x_{i+1})\right] + (x_D - 1)^2\left[1 + \sin^2(2\pi x_D)\right]\right\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	30	[−50, 50]	0

Note: Where, MS is multi-modal separable and MN represents multi-modal non-separable.

Table 3: Function definition for fixed dimension multi-modal benchmark functions

No.	Type	Function	Dimension	Range	F_{min}
F14	MS	$F(x) = \left[1/500 + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{D}(x_i - a_{ij})^6}\right]$	2	[−65.536, 65.536]	0.998004
F15	MN	$F(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	[−5, 5]	0.0003075
F16	MN	$F(x) = 4x_1^2 - 2.1x_1^4 + 1/3x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[−5, 5]	−1.0316285
F17	MN	$F(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	[−5, 5]	0.398
F18	MN	$F(x) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times \left[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	2	[−2, 2]	3

(Continued)

Table 3 (continued)

No.	Type	Function	Dimension	Range	F$_{\min}$
F19	MN	$F(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right]$	3	[−5, 5]	−3.862782
F20	MN	$F(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right]$	6	[−5, 5]	−3.32236
F21	MN	$F(x) = -\sum_{i=1}^{5}\left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$	4	[−5, 5]	−10.1532
F22	MN	$F(x) = -\sum_{i=1}^{7}\left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$	4	[−5, 5]	−10.4029
F23	MN	$F(x) = -\sum_{i=1}^{10}\left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$	4	[−5, 5]	−10.5364

Note: Where, MS is multi-modal separable and MN represents multi-modal non-separable.

The results obtained by RSSO at benchmark functions have been compared with well-known algorithms of recent times.

4.2 Parametric Settings

The research was conducted using MATLAB 2019b software on a specific computer with the following specifications: processor: AMD Ryzen 5 4600 with Radeon graphics 3.00 GHz, RAM: 8.00 GB and operating system: 64-bit Windows 11. For all the algorithms tested, the population size was set to 30 and the maximum number of iterations was set to 1000. The specific settings for each algorithm are summarized in Table 4.

Table 4: Parameter settings of different optimization algorithms

Algorithm	Parameter	Value
RSSO	DF	$random$
	ph_r_1	$random$
	ph_r_2	$random$
	ph_r_3	$random$
	T_r_1	$random$
	T_r_2	$random$
AOA	r_1	$random$
	r_2	$random$
	r_3	$random$
	α	5
	μ	0.499
ALO	a	$\min\left(allvariblesatl^{th}iteration\right)$
	b	$\max\left(allvariblesatl^{th}iteration\right)$
	I	$10^{w}\left(\frac{l}{L}\right)$

(Continued)

Table 4 (continued)

Algorithm	Parameter	Value
DA	s	$2 * random * mc$
	a	$2 * random{*}mc$
	c	$2 * random * mc$
	f	$2 * random$
	e	mc
	mc	$0.1 - l * \left(\frac{0.1}{\left(\frac{L}{2}\right)}\right)$
GWO	$convergenceparameter\,(a)$	$Linearreductionfrom2to0$
MFO	$Convergenceconstant\,(a)$	$[-2\text{–}1]$
	$spiralfactor\,(b)$	1
MVO	r_1	$random$
	r_2	$random$
	r_3	$random$
	p	6
SSA	c_1	$2 * e^{-\left(\frac{4l}{L}\right)^2}$
	c_2	$random$
	c_3	$random$
SCA	a	2
	r_1	$a - l\left(\frac{a}{L}\right)$
	r_2	$2 * \pi * random$
	r_3	$2 * random$
	r_4	$random$

4.3 Quantitative Analysis of RSSO

The proposed RSSO algorithm was compared against other recently developed optimization algorithms using standard benchmark functions. The comparison focused on the average and standard deviation of the results, which are presented in a Table 5.

Table 5: Results of RSSO at classical benchmark functions

Functions	Parameter	AOA	ALO	DA	GWO	MFO	MVO	SCA	SSA	SSO	RSSO
***F*1**	Average	3.73E−64	7.03E−06	1478.518	1.13E−16	1666.667	0.285773	0.333526	1.29E−08	1.73E−197	**0**
	Std	2.04E−63	6.22E−06	766.2623	4.96E−17	3790.49	0.095841	1.714933	3.65E−09	0	0
***F*2**	Average	0	35.34462	13.55902	5.39E−08	40.33356	0.43547	3.96E−05	1.114872	2.64E−120	**0**
	Std	0	46.88196	5.043681	1.84E−08	24.70265	0.13303	7.20E−05	1.260599	7.30E−120	0
***F*3**	Average	0.002365	1151.9	15356.97	443.0136	16766.67	46.24936	4044.797	368.6763	8.19E−92	**0**
	Std	0.00881	666.1661	10095.45	188.0379	11242.6	18.8267	3880.241	254.8877	2.15E−91	0
***F*4**	Average	0.021292	11.55389	23.06015	1.104124	66.64077	1.086291	18.0148	8.17505	8.38E−73	**2.866E−319**
	Std	0.019539	3.317957	6.965639	1.064798	8.528802	0.554548	9.687705	3.482576	3.75E−72	0
***F*5**	Average	**28.25658**	172.1007	147484.5	54.16325	3517.504	170.2001	92095.15	109.1053	28.341713	28.582865
	Std	0.406265	392.8285	142560.8	68.85537	16368.69	200.7955	374414.4	164.4389	0.52367199	0.37061133
***F*6**	Average	2.80945	1.06E−05	1120.928	0	1656.709	0.31324	0.363056	**1.27E−08**	5.1365183	6.1008991
	Std	0.233264	9.56E−06	509.9442	0	3767.988	0.070063	0.128503	2.79E−09	0.19066466	0.23019444

(Continued)

Table 5 (continued)

Functions	Parameter	AOA	ALO	DA	GWO	MFO	MVO	SCA	SSA	SSO	RSSO
F7	Average	4.86E−05	0.09513	0.016716	0.057969	1.63063	0.018222	0.031431	0.096089	3.44E−05	**2.18E−05**
	Std	4.52E−05	0.035386	0.013431	0.024491	2.99583	0.00738	0.028207	0.032296	2.64E−05	1.66E−05
F8	Average	−3202.86	−2413.53	−5754.17	−1587.84	−3140.14	−3025.94	−2324.05	−2883.38	−5815.8841	**−5839.7256**
	Std	240.9948	496.7874	591.5846	300.7621	347.8142	290.339	202.7524	311.6512	75.940633	92.768738
F9	Average	0	80.45882	162.1103	25.37144	161.282	106.9613	17.15178	57.17687	0	**0**
	Std	0	25.80236	38.31689	5.746865	43.39013	32.2269	22.96072	22.52042	0	0
F10	Average	**8.88E−16**	1.973304	9.232492	8.40E−09	14.00294	1.283419	13.96645	2.165787	1.07E−15	**8.88E−16**
	Std	2.01E−31	0.611315	1.44599	2.20E−09	7.950571	0.728616	8.91743	0.895879	7.94E−16	**0**
F11	Average	0.101451	0.01314	12.75532	9.167486	15.07443	0.550285	0.29508	0.006158	0	0
	Std	0.079528	0.014247	6.049923	5.613008	53.42284	0.110778	0.275885	0.006458	0	0
F12	Average	**0.390268**	11.79996	337.437	0.147818	0.842388	1.671162	100.5139	6.031526	0.59370526	0.83335773
	Std	0.05154	3.893738	1274.537	0.190267	1.086202	1.03551	528.101	3.243773	0.061096459	0.080747258
F13	Average	2.775434	3.566452	60001.51	**0.039062**	0.686478	0.057376	107.5754	2.688221	2.5764391	2.7841001
	Std	0.105293	11.30312	157203.5	0.163974	1.111076	0.033767	475.3857	8.903576	0.10194472	0.04152751
F14	Average	9.305394	1.428617	0.998004	3.920338	1.691591	**0.998004**	1.655152	1.031138	3.6534025	4.0440541
	Std	3.82843	0.564551	6.39E−08	2.721899	1.165462	6.78E−16	1.87623	0.181484	3.1813031	3.5201559
F15	Average	0.010175	0.004013	0.00371	0.002483	0.001622	0.003847	0.000862	0.001416	**0.000551011**	0.000595998
	Std	0.022732	0.007439	0.005836	0.00146	0.003566	0.007011	0.000335	0.003584	0.000278742	0.000148682
F16	Average	**−1.03163**	−1.03163	−1.03163	−1.03163	−1.03163	−1.03163	−1.0316	−1.03163	−1.031427	−1.031552
	Std	9.32E−08	0	3.66E−06	0	0	1.04E−07	3.03E−05	0	0.000135908	6.11E−05
F17	Average	0.405421	**0.397887**	0.397895	0.397887	0.397887	0.397888	0.399022	0.397887	0.39936786	0.6827929
	Std	0.005929	1.69E−16	3.56E−05	1.69E−16	1.69E−16	2.06E−07	0.001427	1.69E−16	0.003070781	0.000027446
F18	Average	10.0868	3	3.000007	3	3	3.000001	3.000021	3	3	**3**
	Std	11.95653	0	2.24E−05	0	0	7.04E−07	3.49E−05	0	0	0
F19	Average	−3.85342	−3.86278	−3.8627	−3.86278	−3.86252	−3.86278	−3.85498	**−3.86278**	−3.7299897	−3.7483059
	Std	0.002905	1.36E−15	0.000138	1.36E−15	0.001439	3.97E−07	0.001994	1.36E−15	0.072521232	0.077402915
F20	Average	−3.09355	**−3.27839**	−3.24973	−3.322	−3.2341	−3.25042	−2.91013	−3.22916	−2.4727819	−2.532164
	Std	0.085726	0.058284	0.077816	2.26E−15	0.060746	0.059439	0.377887	0.052154	0.35304148	0.29383272
F21	Average	−3.74402	−6.53424	−7.35707	−5.59336	−6.14456	−7.46314	−2.82024	−8.13991	−3.1786562	**−10.1523**
	Std	0.788914	2.932695	2.673835	3.575389	3.448257	3.023758	1.808205	2.980425	0.70870283	0.001507
F22	Average	−4.0787	−7.66281	−8.67404	−10.27	−8.42414	−9.16675	−3.43843	−10.2271	−3.015563	**−10.4018**
	Std	1.044784	3.22496	2.679209	0.728055	3.141982	2.278944	2.01902	0.962918	0.60410782	0.001336
F23	Average	−3.96481	−7.02753	−8.36917	−10.4662	−7.89781	−9.40438	−4.84152	−8.96732	−3.0950104	**−10.5341**
	Std	1.469116	3.438882	2.905126	0.384508	3.584177	2.618164	2.075541	3.192975	0.71336859	0.005092

Note: The bold text is showing the best values obtained by the optimization algorithm.

4.4 *Statistical Analysis of Proposed Optimization RSSO*

While no direct comparison of RSSO with other optimization algorithms has been published independently, the study conducted 30 independent runs of RSSO and compared its performance (in terms of average and standard deviation) to other algorithms. To rule out the possibility that the superior performance of RSSO was simply due to chance, a statistical test called the Wilcoxon rank-sum test was performed. Table 6 summarizes the p-values obtained from this test for each benchmark function. A p-value less than 0.05 indicates strong support for the hypothesis that RSSO is significantly better than the other algorithms. Since RSSO cannot be compared to itself, "N/A" (Not Applicable) is used in these cases.

Table 6: Statistical results obtained by Wilcoxon rank sum test in terms of p value

Functions	AOA	ALO	DA	GWO	MFO	MVO	SCA	SSA	SSO	RSSO
F1	4.50×10^{-11}	4.50×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	2.98×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	**NaN**
F2	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	**NaN**
F3	3.34×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	**NaN**
F4	3.34×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	**NaN**

(Continued)

Table 6 (continued)

Functions	AOA	ALO	DA	GWO	MFO	MVO	SCA	SSA	SSO	RSSO
F5	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	**NaN**
F6	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	**NaN**	1.20×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F7	0.032651	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	**NaN**
F8	3.02×10^{-11}	2.99×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	2.99×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	**NaN**
F9	**NaN**	1.21×10^{-12}	1.21×10^{-12}	1.20×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.17×10^{-12}	**NaN**
F10	**NaN**	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.20×10^{-12}	1.19×10^{-12}	**NaN**
F11	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	NaN	**NaN**
F12	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	0.379036	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	0.003501	0.003501
F13	3.02×10^{-11}	4.98×10^{-11}	3.02×10^{-11}	**NaN**	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	0.662682	0.662572	4.98×10^{-11}
F14	8.41×10^{-13}	0.000132	0.333711	4.57×10^{-12}	0.00031	**NaN**	4.50×10^{-12}	0.333711	0.022124	0.021561
F15	7.20×10^{-05}	6.52×10^{-09}	3.69×10^{-11}	3.02×10^{-11}	7.32×10^{-11}	1.09×10^{-10}	1.33×10^{-10}	6.07×10^{-11}	**NaN**	**NaN**
F16	2.39×10^{-12}	**NaN**	1.24×10^{-07}	**NaN**	**NaN**	3.71×10^{-13}	1.21×10^{-12}	**NaN**	1.21×10^{-12}	1.21×10^{-12}
F17	1.21×10^{-12}	**NaN**	1.27×10^{-05}	**NaN**	**NaN**	4.48×10^{-12}	1.21×10^{-12}	**NaN**	1.21×10^{-12}	1.21×10^{-12}
F18	0.002787	**NaN**	1.27×10^{-05}	**NaN**	**NaN**	5.70×10^{-11}	1.21×10^{-12}	**NaN**	**NaN**	**NaN**
F19	1.21×10^{-12}	**NaN**	4.57×10^{-12}	**NaN**	0.333711	1.67×10^{-08}	1.21×10^{-12}	**NaN**	1.21×10^{-12}	1.21×10^{-12}
F20	1.21×10^{-12}	0.000313	4.57×10^{-12}	**NaN**	2.92×10^{-08}	1.21×10^{-12}	1.21×10^{-12}	5.85×10^{-09}	4.79×10^{-08}	1.21×10^{-12}
F21	3.02×10^{-11}	0.075837	2.87×10^{-06}	0.074498	0.183369	0.147367	3.02×10^{-11}	0.024255	0.001489	**NaN**
F22	3.02×10^{-11}	0.372965	0.001679	4.56×10^{-11}	0.006661	0.003938	3.02×10^{-11}	4.56×10^{-11}	3.81×10^{-05}	**NaN**
F23	3.02×10^{-11}	0.660515	8.19×10^{-07}	4.56×10^{-11}	0.072549	0.000168	3.02×10^{-11}	3.81×10^{-05}	0.006675	**NaN**

Note: The bold text is showing the best values obtained by the optimization algorithm.

4.5 Qualitative Analysis of the RSSO

Using convergence analysis and box plots, the suggested RSSO has been qualitatively analyzed. Fig. 2 displays the RSSO algorithm's convergence graphs for the classical benchmark functions. Comparing it to the basic SSO and other algorithms, it appears that the RSSO approach demonstrates faster convergence. Due to the stochastic character of metaheuristic algorithms, Box plots have been used to verify RSSO's stability by comparing the means obtained by various algorithms, as shown in Fig. 3. To further compare RSSO outcomes to different optimization procedures, box plots of the top individuals' fitness from the last generation are provided. This further demonstrates RSSO's superior performance and convergence capabilities. If all trials yield the same result, the RSSO algorithm can strike a balance between the exploration and efficiency phases.

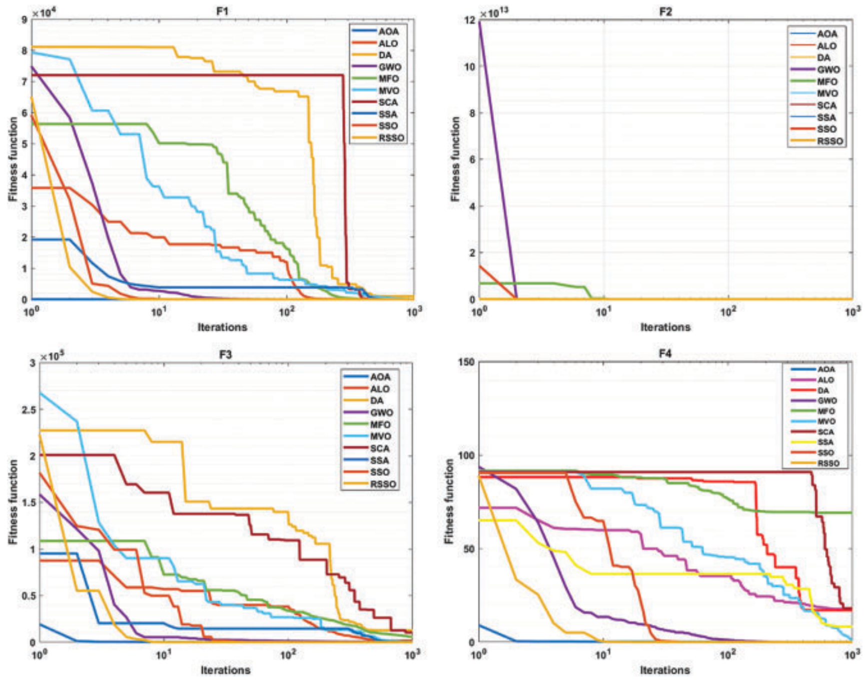

Figure 2: (Continued)

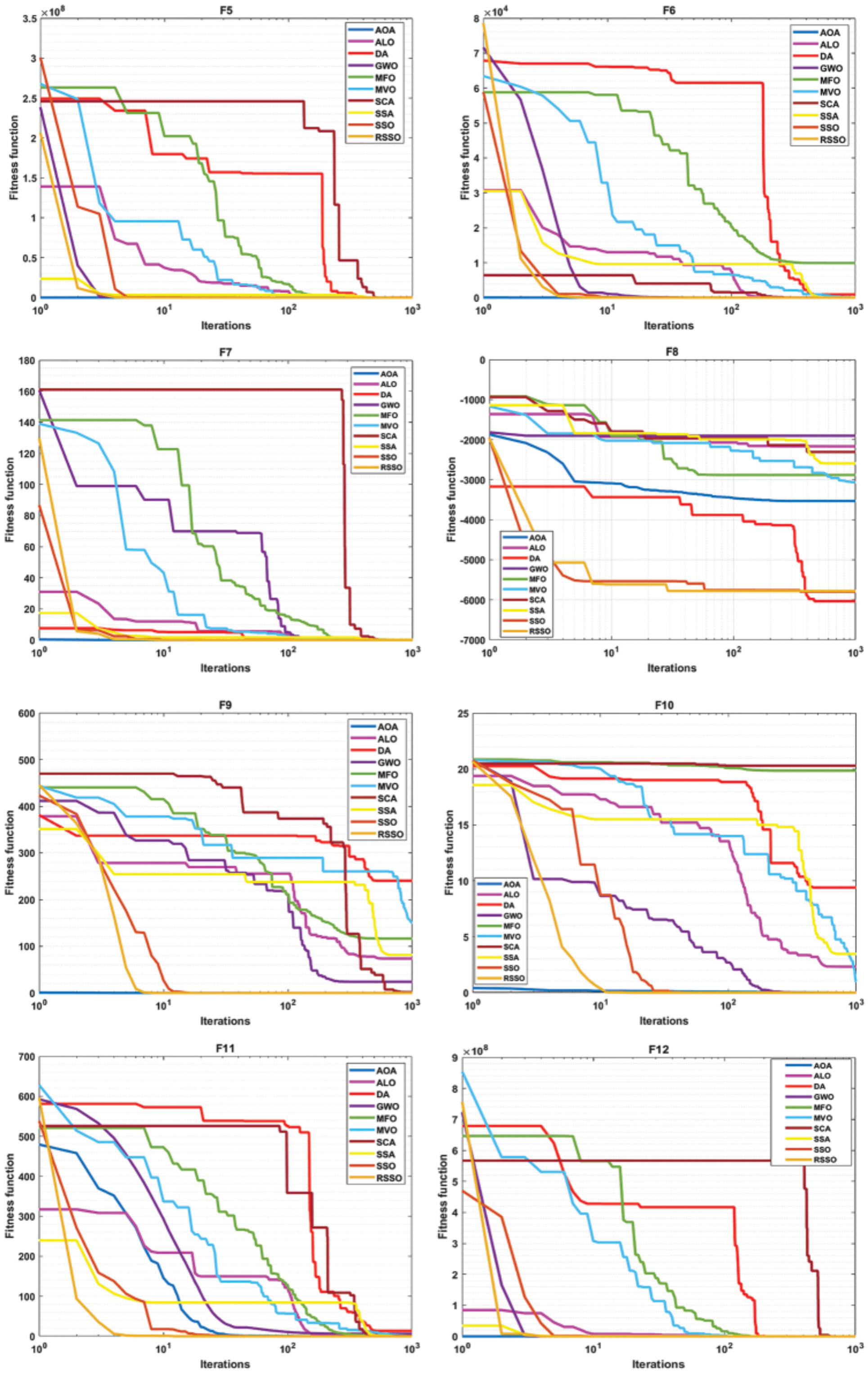

Figure 2: (Continued)

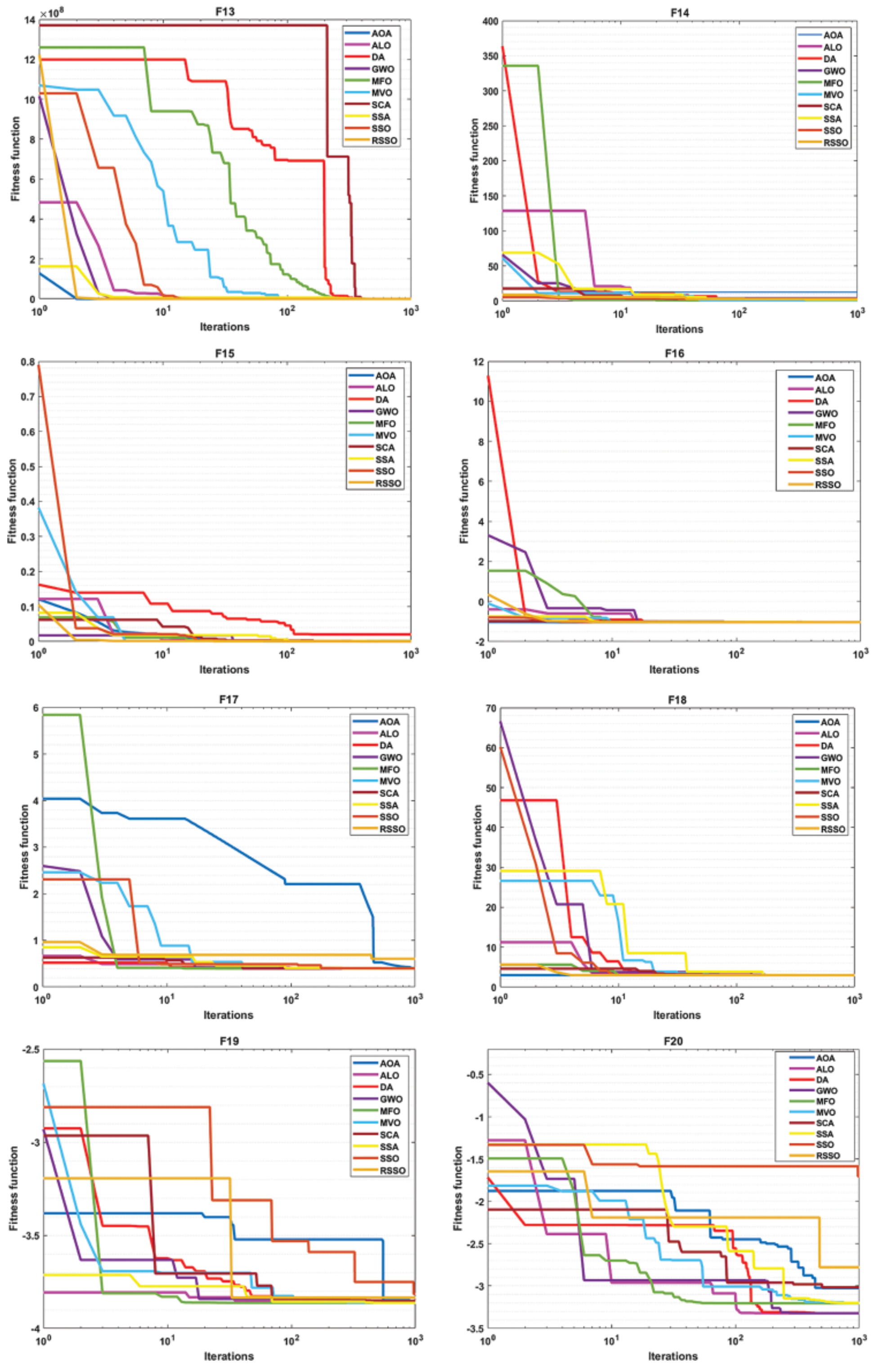

Figure 2: (Continued)

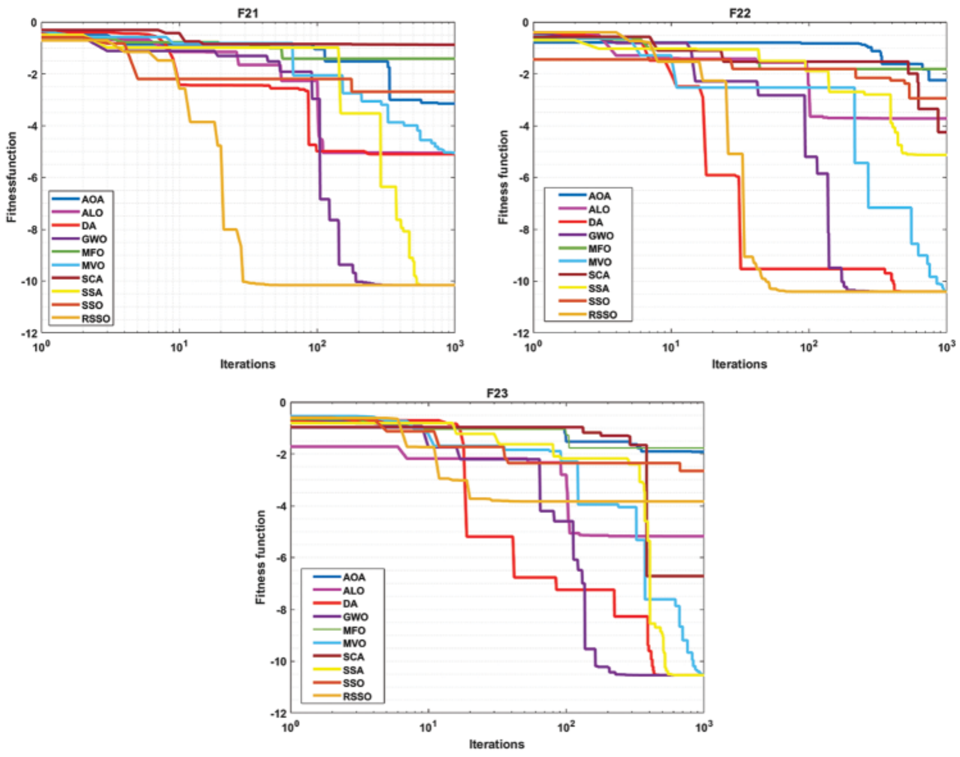

Figure 2: Convergence behaviour for functions F1–F23

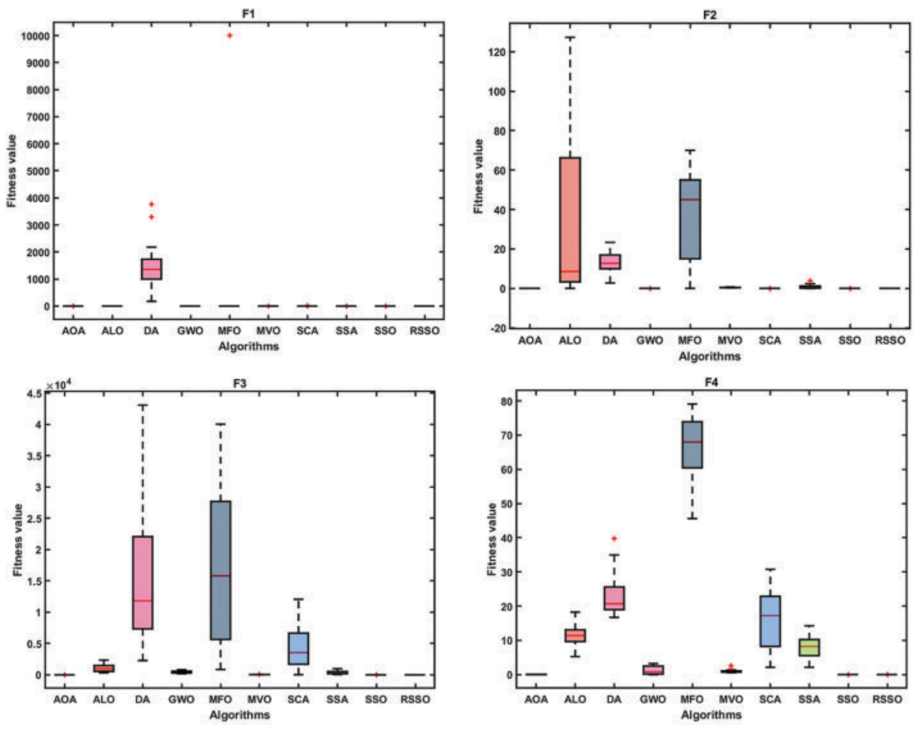

Figure 3: (Continued)

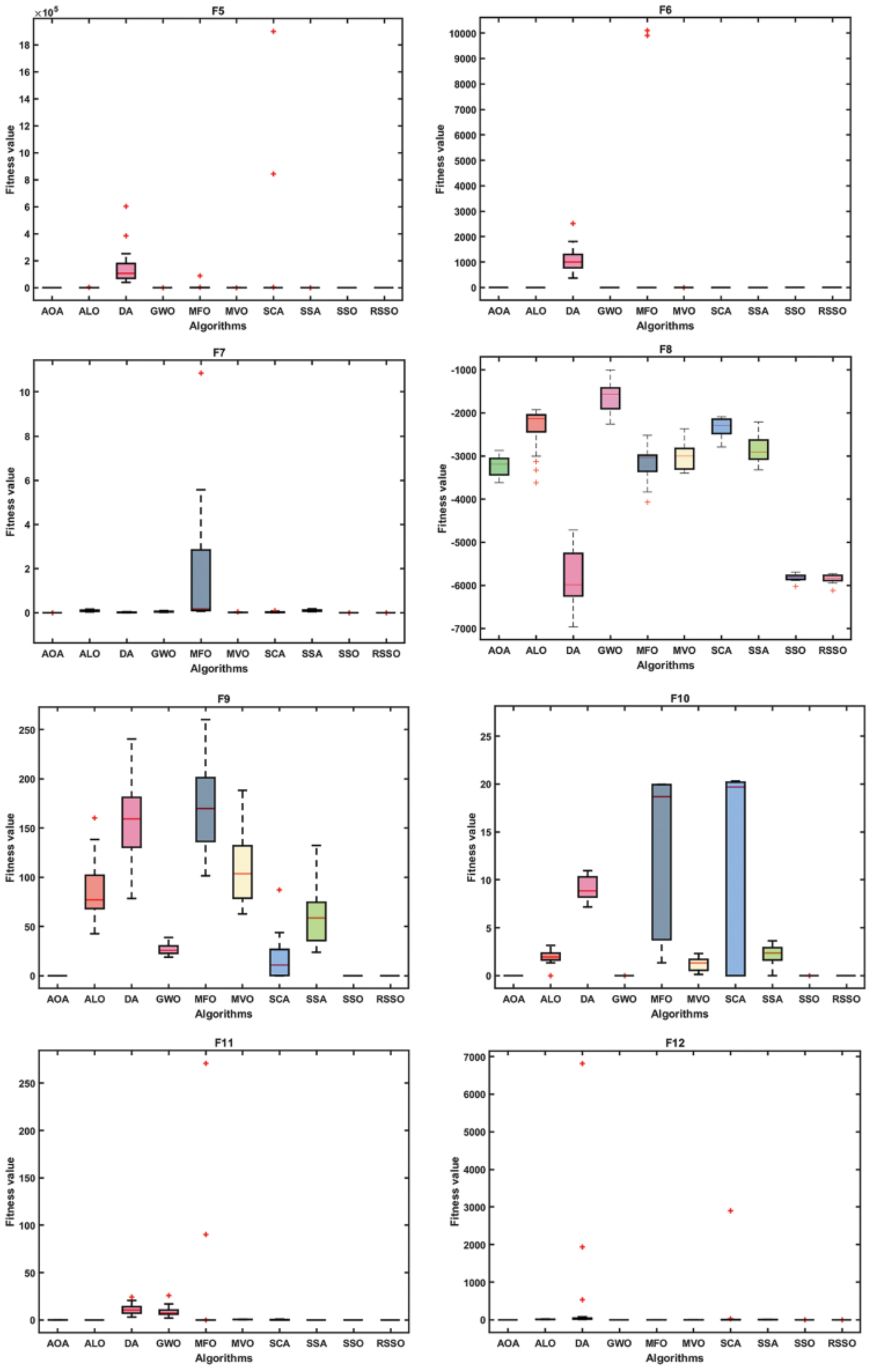

Figure 3: (Continued)

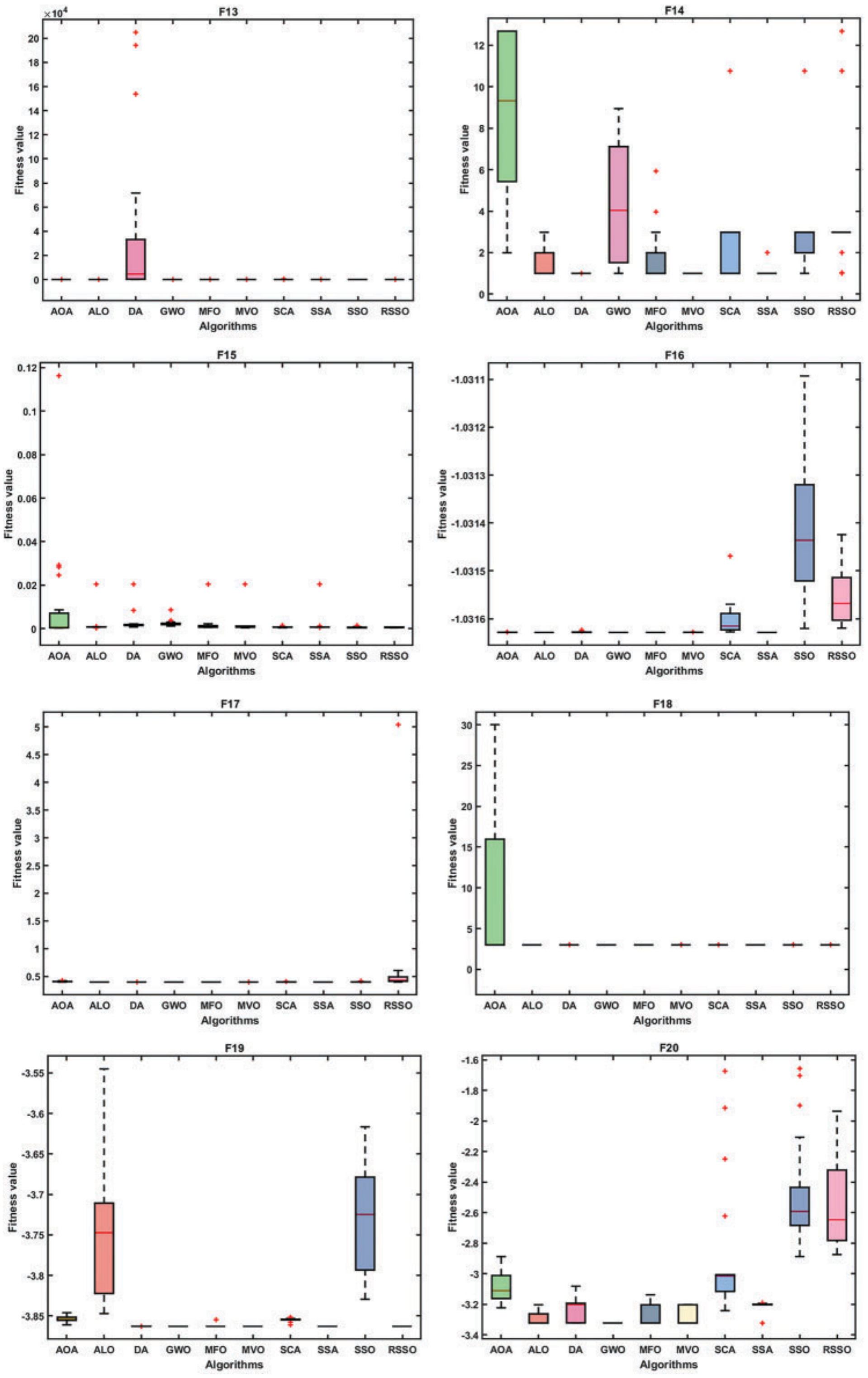

Figure 3: (Continued)

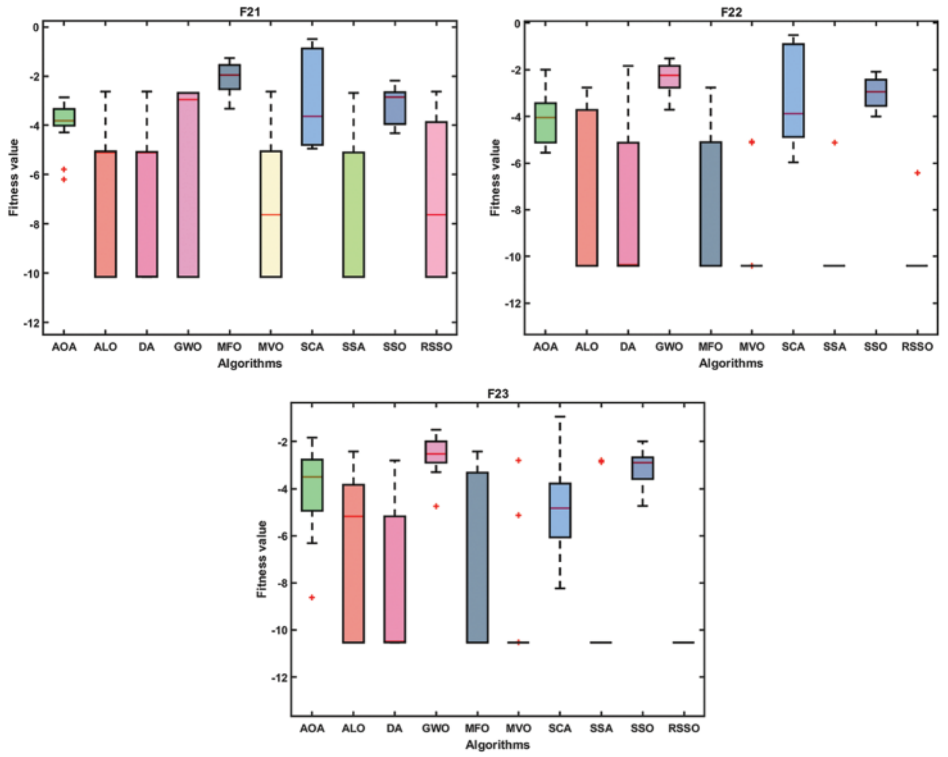

Figure 3: Box plots for functions F1–F23

4.6 Quantitative Analysis of CEC 2018 Test Functions

The robustness of the proposed algorithms has been evaluated using the CEC 2018 test functions. Additionally, these algorithms have been compared with other contemporary algorithms, with results sourced from Ref. [25]. As shown in Table 7, RSSO delivers optimal results for most functions, including F1, F3–F6, F9, F11, F14–F17, F20–F22, F25, and F27–F28. In contrast, MFO achieves optimal results for functions such as F7, F10, F19, and F30. Meanwhile, AOA provides optimal outcomes for F8, F12, F18, F23, F26, and F29.

Table 7: Comparison of QRGM-AOA with well-known optimization at CEC 2018 test functions

Fn	Parameter	CSA	PSO	MFO	SMFO	AOA	SSO	RSSO
1	**Average**	3.246E+10	5.907E+10	6.278E+09	3.119 E+10	9.986E+10	2.4478346E+10	**5.268354E+09**
	Std	2.130E+10	3.270E+10	1.027E+09	1.734 E+10	5.602E+09	2.4958522E+09	2.51242E+09
3	**Average**	9.600E+04	1.308E+05	9.453E+04	8.300 E+04	9.453E+04	6.3570338E+04	**6.0054458E+04**
	Std	5.473E+04	1.039E+05	1.203E+04	7.186 E+04	2.104E+04	8.2872876E+03	8.065423E+03
4	**Average**	5.726E+03	1.183E+04	8.558E+02	5.612 E+03	7.225E+02	3.1067644E+03	**7.99852E+02**
	Std	3.185E+03	7.302E+03	4.991E+02	2.322 E+03	3.852E+02	9.5901826E+02	5.95926E+02
5	**Average**	8.509E+02	9.635E+02	6.740E+02	8.725 E+02	5.640E+02	8.3439158E+02	**5.52198E+02**
	Std	8.017E+02	8.845E+02	6.114E+02	8.105 E+02	5.225E+02	1.9876379E+01	1.887531E+01
6	**Average**	6.683E+02	6.921E+02	6.260E+02	6.814 E+02	6.912E+02	6.7132678E+02	**6.195792E+02**
	Std	6.554E+02	6.828E+02	6.113E+02	6.571 E+02	4.227E+02	5.8798258E+00	5.71118E+00
7	**Average**	1.730E+03	2.511E+03	**1.007E+03**	1.359 E+03	1.152E+03	1.2035080E+03	1.009120E+03
	Std	1.552E+03	2.201E+03	8.311E+02	1.198 E+03	5.125E+02	4.0996442E+01	3.995421E+01
8	**Average**	1.134E+03	1.220E+03	9.895E+02	1.093 E+03	**9.995E+02**	1.0636021E+03	1.042513E+03
	Std	1.105E+03	1.163E+03	9.126E+02	1.052 E+03	4.186E+02	2.0778253E+01	1.676781E+01

(Continued)

Table 7 (continued)

Fn	Parameter	CSA	PSO	MFO	SMFO	AOA	SSO	RSSO
9	**Average**	1.040E+04	1.735E+04	6.219E+03	9.431 E+03	5.129E+03	7.6133748E+03	**5.092154E+03**
	Std	7.223E+03	1.271E+04	3.323E+03	7.359E+103	2.189E+03	6.5544697E+02	5.905312E+02
10	**Average**	8.279E+03	8.218E+03	**5.259E+03**	8.272 E+03	5.527E+03	7.2382791E+03	5.562151E+03
	Std	7.738E+03	7.661E+03	4.231E+03	7.449 E+03	4.882E+03	3.3771434E+02	3.214524E+02
11	**Average**	4.700E+03	1.018E+04	3.967E+03	5.799 E+03	3.634E+03	3.4017747E+03	**3.36592E+03**
	Std	3.395E+03	7.488E+03	1.370E+03	2.547 E+03	1.224E+03	8.3572634E+02	5.62356E+02
12	**Average**	2.979E+09	6.824E+09	9.043E+07	4.342 E+09	**8.294E+07**	3.4094207E+09	9.243121E+08
	Std	1.516E+09	3.870E+09	7.305E+04	2.607 E+09	6.189E+04	8.6961284E+08	9.658732E+08
13	**Average**	9.478E+08	**3.156E+04**	4.593E+06	7.405E+08	5.125E+06	1.1928068E+09	2.251521E+08
	Std	5.211E+08	5.760E+08	1.003E+04	1.145 E+08	2.112E+04	7.9669524E+08	2.998542E+08
14	**Average**	4.342E+05	7.227E+05	6.942E+04	1.715 E+06	5.124E+04	6.5681555E+05	**4.99845E+04**
	Std	1.482E+05	1.041E+05	5.450E+03	7.879 E+04	4.250E+03	4.2636747E+05	6.23586E+04
15	**Average**	7.286E+07	2.025E+08	3.090E+04	4.161 E+07	3.124E+04	6.6094562E+05	**3.015249E+04**
	Std	2.378E+07	1.064E+07	5.117E+03	1.868 E+06	5.236E+03	3.3070797E+03	2.325923E+03
16	**Average**	3.989E+03	4.452E+03	2.956E+03	4.223 E+03	2.956E+03	3.7529462E+03	**2.956E+03**
	Std	3.147E+03	3.827E+03	2.398E+03	3.565E+103	1.426E+03	2.5093522E+02	1.20120E+03
17	**Average**	2.628E+03	3.298E+03	2.349E+03	2.788E+103	3.259E+03	2.6133274E+03	**2.31653E+03**
	Std	2.256E+03	2.755E+03	1.975E+03	2.359 E+03	1.126E+03	2.0465634E+02	2.065321E+02
18	**Average**	7.890E+06	6.473E+06	2.830E+06	5.330 E+07	**1.259E+05**	4.1228460E+06	2.265124E+06
	Std	2.022E+06	6.593E+05	7.725E+04	2.825 E+06	6.512E+04	3.3306227E+06	3.215462E+06
19	**Average**	1.358E+08	2.509E+08	**4.261E+06**	7.588 E+07	5.665E+06	3.2431428E+07	3.236532E+06
	Std	6.263E+07	3.341E+07	1.293E+04	5.192 E+06	0.843E+04	3.3495209E+07	3.264213E+06
20	**Average**	2.759E+03	2.847E+03	2.537E+03	2.837 E+03	3.246E+03	2.6347378E+03	**2.53125E+03**
	Std	2.476E+03	2.574E+03	2.215E+03	2.454E+03	2.216E+03	1.2882269E+02	1.32654E+02
21	**Average**	2.625E+03	2.707E+03	2.472E+03	2.630E+03	2.795E+03	2.5989179E+03	**2.469562E+03**
	Std	2.587E+03	2.615E+03	2.420E+03	2.363E+03	2.129E+03	1.7175886E+01	1.365213E+01
22	**Average**	6.831E+03	8.759E+03	6.353E+03	8.681E+03	5.239E+03	5.5538834E+03	**5.036521E+03**
	Std	5.558E+03	6.900E+03	3.223E+03	5.677E+03	4.268E+03	1.2200116E+03	1.320012E+01
23	**Average**	3.143E+03	3.239E+03	2.811E+03	3.273E+03	**2.126E+03**	3.1611422E+03	2.821652E+03
	Std	3.061E+03	3.101E+03	2.740E+03	3.027E+03	2.852E+03	6.3662237E+01	3.265315E+01
24	**Average**	3.319E+03	3.539E+03	2.979E+03	3.482E+03	2.216E+03	3.4117165E+03	3.002512E+03
	Std	3.206E+03	3.253E+03	2.926E+03	3.217E+03	2.249E+03	7.2482310E+01	2.326591E+01
25	**Average**	4.890E+03	7.655E+03	3.181E+03	3.972E+03	3.958E+03	3.4732912E+03	**3.171523E+03**
	Std	4.344E+03	5.843E+03	2.895E+03	3.467E+103	2.129E+03	1.9152232E+02	1.32654E+02
26	**Average**	8.661E+03	8.709E+03	5.650E+03	9.093 E+03	**5.112E+03**	8.1569419e+03	7.659832E+03
	Std	7.772E+03	6.500E+03	4.921E+03	5.057 E+03	2.672E+03	4.8626737e+02	5.1579234E+02
27	**Average**	3.690E+03	3.827E+03	3.233E+03	3.754 E+03	3.111E+03	3.6988474e+03	**3.111E+03**
	Std	3.537E+03	3.591E+03	3.206E+03	3.538 E+03	3.139E+03	1.2144742e+02	1.35621E+02
28	**Average**	5.474E+03	6.851E+03	3.756E+03	5.462 E+03	3.813E+03	4.5653177e+03	**3.7559E+03**
	Std	4.541E+03	5.611E+03	3.263E+03	4.419 E+03	3.129E+03	4.7777496e+02	4.03265E+02
29	**Average**	5.253E+03	5.426E+03	4.014E+03	5.639 E+03	**3.098E+03**	5.2867215e+03	5.112621E+03
	Std	4.952E+03	4.907E+03	3.499E+03	4.728 E+03	2.559E+03	2.9107176e+02	2.365321E+02
30	**Average**	1.084E+08	2.946E+08	**2.524E+05**	2.326 E+08	4.198E+05	1.2701093e+08	1.112651E+07
	Std	4.274E+07	8.913E+07	7.219E+03	2.468 E+07	6.772E+03	3.8396778e+07	2.365214E+06

Note: The bold text is showing the best values obtained by the optimization algorithm.

4.7 Comparison of Computational Complexity

The computational complexity of the RSSO algorithm has been analyzed using Big-O notation and compared to the basic SSO algorithm. The results, presented in Table 8, show that both algorithms have similar time complexity. In other words, they take roughly the same amount of time to execute.

Table 8: Computational complexity of SSO and RSSO

Functions	SSO	RSSO	Functions	SSO	RSSO
***F*1**	4.0404864	3.9900107	***F*13**	5.28386	5.0602747
***F*2**	4.0412297	4.0192187	***F*14**	2.9023463	2.72382
***F*3**	5.225257	4.8509542	***F*15**	1.3209882	1.2065279
***F*4**	4.633608	4.1676389	***F*16**	0.8762344	0.8542739
***F*5**	4.0746033	4.2390084	***F*17**	1.181482	0.8933553
***F*6**	4.0494139	4.1275828	***F*18**	0.916449	0.8567044
***F*7**	4.4197981	4.3274047	***F*19**	1.0240062	1.1327767
***F*8**	4.4731884	4.1521003	***F*20**	1.6778817	1.4735105
***F*9**	4.0829756	4.0392895	***F*21**	1.2996001	1.3685306
***F*10**	4.1208045	4.2581151	***F*22**	1.2898021	1.3745246
***F*11**	4.3691686	4.7054779	***F*23**	1.6077693	2.222182
***F*12**	5.0312676	5.1842126			

Note: The bold text is showing the best values obtained by the optimization algorithm.

The computational complexity is evaluated based on Algorithms 1 and 2 using the following Eq. (12).

$$O\left(NP \times D \times \max\_iter\right) \tag{12}$$

5 Application of the Proposed RSSO Algorithm to the Real-World Application

The proposed algorithm proved its efficiency efficient through quantitative and qualitative analysis of the benchmark problems. The efficacy of the proposed algorithm has also been validated on real-world optimization problems through well-known engineering design problems.

5.1 Gear Train Design Problem

Fig. 4 depicts the gear train design problem as an unconstrained case study with decision variables with discrete values. The following is a mathematical representation of these choice variables as train gears.

$$\text{Minimize } f\left(n\right) = \left(\frac{1}{6.931} - \frac{n_C n_D}{n_A n_B}\right)^2 \tag{13}$$

subjected to

$$12 \leq n_A, n_B, n_C, n_D \leq 60$$

The obtained results are tabulated in Table 9. The parameters in each optimization algorithm are set to similar values for fair comparison. From Table 8, it can be observed that the proposed RSSO algorithm gives superior results when compared to other algorithms.

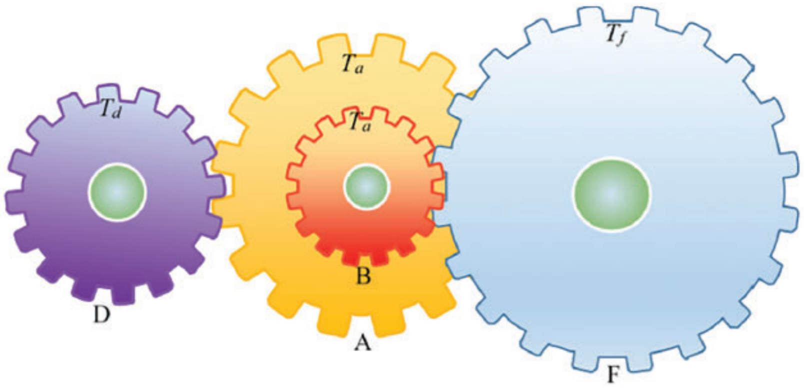

Figure 4: Gear train design problem

Table 9: Results for the gear train design

Algorithms	Optimized parameters				Optimum value
	n_A	n_B	n_C	n_D	
BDE	59	15	21	37	3.0676E−10
SinDE	53	13	20	34	2.3078E−11
IDE	53	13	30	51	2.3078E−11
CMA-ES	60	12	12	12	3.1048E−03
OBSCA	52	15	30	60	2.3576E−09
m-SCA	53	13	20	34	2.3078E−11
WOA	55	14	17	30	1.3616E−09
GWO	54	17	22	48	1.1661E−10
MFO	51	16	23	50	1.1834E−09
PSO	54	17	22	48	1.1661E−10
SCA	55	17	14	30	1.3616E−09
MG-SCA	43	16	19	49	2.7009E−12
RSSO	43	16	20	48	2.6999E-12

5.2 Speed Reducer Design Problem

Speed reducer as shown in Fig. 5 is designed to minimize the weight of the reducer. The face width (n_1), a module of teeth (n_2), number of teeth on the pinion (n_3), length of the first shaft between bearings (n_4), length of the second shaft between bearings (n_5), the diameter of the first shaft (n_6), and the diameter of the first shaft (n_7) are the key variables to be optimized.

$$\text{Minimize } f(n)7 = 0.785n_1n_2^2\left(3.333n_3^2 + 14.9334n_3 - 42.0934\right) - 1.508n_1\left(n_6^2 + n_7^2\right) + 7.4777n_1\left(n_6^3 + n_7^3\right) + 1.508n_1\left(n_4n_6^2 + n_5n_7^2\right) \tag{14}$$

subject to:

$$g_1(n) = \frac{21}{n_1n_2^2n_3} - 1 \le 0; \tag{15}$$

$$g_2(n) = \frac{397.5}{n_1 n_2 n_3^2} - 1 \le 0; \tag{16}$$

$$g_3(n) = \frac{1.93 n_4^3}{n_1 n_3 n_6^4} - 1 \le 0; \tag{17}$$

$$g_4(n) = \frac{1.93 n_4^3}{n_1 n_3 n_7^4} - 1 \le 0; \tag{18}$$

$$g_5(n) = \frac{1}{110 n_6^3}\sqrt{\left(\frac{745 n_4}{n_2 n_3}\right)^2 + 16.9 \times 10^6 - 1 \le 0}; \tag{19}$$

$$g_6(n) = \frac{1}{85 n_7^3}\sqrt{\left(\frac{745 n_4}{n_2 n_3}\right)^2 + 157.5 \times 10^6 - 1 \le 0}; \tag{20}$$

$$g_7(n) = \frac{n_2 n_3}{40} - 1 \le 0; \tag{21}$$

$$g_8(n) = \frac{5 n_2}{n_1} - 1 \le 0; \tag{22}$$

$$g_9(n) = \frac{n_1}{12 n_2} - 1 \le 0; \tag{23}$$

$$g_{10}(n) = \frac{1.5 n_6 + 1.9}{n_4} - 1 \le 0; \tag{24}$$

$$g_{11}(n) = \frac{1.1 n_7 + 1.9}{n_5} - 1 \le 0; \tag{25}$$

variable range: $2.6 \le n_1 \le 3.6; 0.7 \le n_2 \le 0.8; 17 \le n_3 \le 28;$
$7.3 \le n_4 \le 8.3; 7.8 \le n_5 \le 8.3; 2.9 \le n_6 \le 3.9; 5 \le n_7 \le 5.5.$

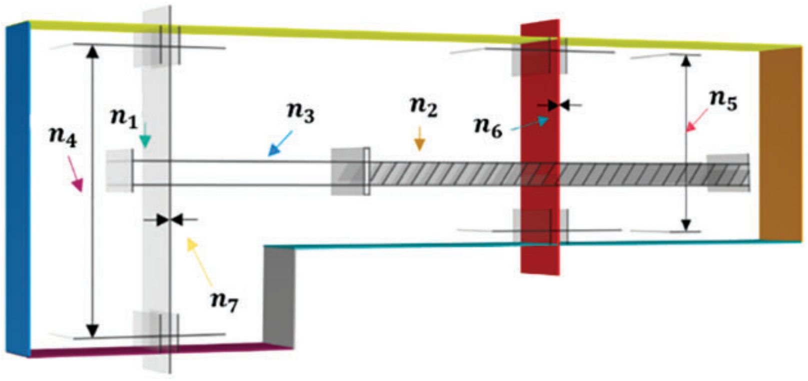

Figure 5: Speed reducer design problem

The performance of the proposed RSSO algorithm was compared to several other optimization algorithms, including MBFPA, WCA, PSODE, MDE, HEAA, and PVS. Table 10 shows that RSSO consistently produces superior solutions, particularly when dealing with complex constraints.

Table 10: Comparison of optimization results for speed reducer [26,27]

Algorithms	n_1	n_2	n_3	n_4	n_5	n_6	n_7	Optimal cost
MBFPA	3.500	0.7	17	7.3	7.7153199122	3.35021466	5.28665446	2994.341315
WCA	3.500	0.7	17	7.3	7.715319	3.350214	5.286654	2994.471066
PSODE	3.500	0.7	17	7.3	7.800000	3.350214	5.2866832	2996.348167
MDE	3.50001	0.7	17	7.300156	7.800027	3.350221	5.286685	2996.356689
HEAA	3.500022	0.7	17.000012	7.300427	7.715377	3.350230	5.286663	2994.499107
PVS	3.49999	0.6999	17	7.3	7.8	3.3502	5.2866	2996.3481
RSSO	3.3594	0.5987	17	7.2	7.8	3.3024	5.12974	2992.8452

5.3 Parameter Estimation for Frequency-Modulated Synthesizer

The decision variables of a frequency-modulated synthesizer are estimated in this problem. This is an unconstrained, multimodal, complicated problem which contains six decision variables, a_1, a_2, a_3, n_1, n_2 and n_3. The problem is formulated in the following manner

$$\text{Minimize } f(x) = \sum_{t=1}^{100} \left(X(t,\vec{x}) - X_0(t,\vec{x})\right)^2; \tag{26}$$

$\vec{x} = a_1, a_2, a_3, n_1, n_2, n_3$ and

$\overline{x} = (1, 5, 1.5, 4.8, 2, 4.9)$

subjected to $-6.4 \le a_1, a_2, a_3, n_1, n_2, n_3 \le 6.35$

where

$$X(t,\vec{x}) = a_1 \sin\left(n_1 t\theta + a_2 \sin\left(n_2 t\theta + a_3 \sin\left(n_3 t\theta\right)\right)\right), \tag{27}$$

$$X_0(t,\overline{x}) = \sin\left(5t \times \frac{2\pi}{100} + 1.5 \sin\left(4.8t \times \frac{2\pi}{100} + 2\sin\left(4.9t \times \frac{2\pi}{100}\right)\right)\right). \tag{28}$$

A comprehensive comparison was conducted to evaluate the performance of the proposed RSSO algorithm against various other optimization techniques, including BDE, SinDE, IDE, CMA-ES, OBSCA, m-SCA, WOA, GWO, MFO, PSO, SCA, and MG-SCA. The comparison considered several metrics: best solution, worst solution, standard deviation, and average performance. All algorithms were tested using 10^5 function evaluations, a population size of 30, and 30 independent runs. The results consistently showed that RSSO outperforms all other optimization algorithms tested as shown in Table 11.

Table 11: Results for parameter estimation for frequency-modulated [28]

Algorithms	Best	Worst	Standard deviation	Average	Statistical decision
BDE	14.81	28.38	3.27	22.29	+
SinDE	0.00	16.96	6.45	5.60	−
IDE	0.00	16.15	5.98	5.69	−
CMA-ES	24.52	29.46	1.02	28.45	+
OBSCA	4.48	21.15	5.12	9.65	=
m-SCA	10.90	22.84	4.15	15.94	+
WOA	11.38	25.10	4.42	18.12	+

(Continued)

Table 11 (continued)

Algorithms	Best	Worst	Standard deviation	Average	Statistical decision
GWO	8.42	25.10	5.16	14.77	+
MFO	11.58	26.82	4.41	21.57	+
PSO	25.24	29.65	1.17	27.63	+
SCA	10.31	22.20	3.48	14.53	=
MG-SCA	0.001949	19.91	4.58	12.25	=
RSSO	10.05	14.9209	2.48	12.2445	

5.4 Three-Bar Truss Design Problem

Fig. 6 illustrates the three-bar truss design problem, which involves minimizing the weight of a three-bar truss structure. This optimization problem includes constraints related to stress, deflection, and buckling, ensuring that the final design meets structural integrity requirements.

$$[n] = [n_1, n_2] = [A_1, A_2] \tag{29}$$

$$\text{Minimize } f(n) = \left(2\sqrt{2}n_1 + n_2\right) \times l. \tag{30}$$

Subject to:

$$g_1(n) = \sqrt{2}n_1 + n_2/\sqrt{2}n_1^2 + 2n_1n_2P - \sigma \leq 0; \tag{31}$$

$$g_2(n) = n_2/\sqrt{2}n_1^2 + 2n_1n_2P - \sigma \leq 0; \tag{32}$$

$$g_3(n) = 1/\sqrt{2}n_2 + n_1P - \sigma \leq 0; \tag{33}$$

variable range $0 \leq n_1, n_2 \leq 1$.
where $l = 100$ cm; $P = 2$ kN/cm^2; $r = 2$ kN/cm^2.

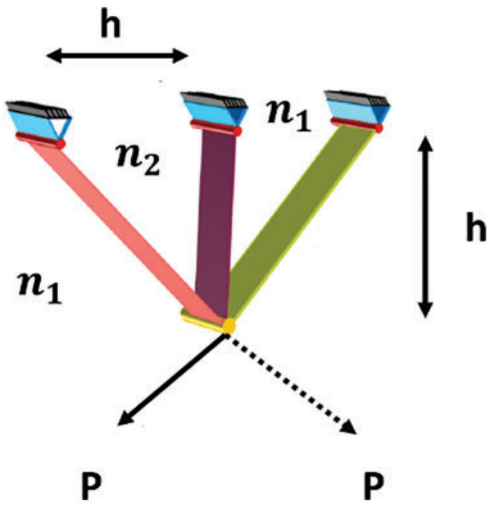

Figure 6: Three-bar truss design problem

The results are tabulated in Table 12 which suggests that the RSSO perform better in this problem when compared to other algorithms.

Table 12: Comparison of results for the three-bar truss design problem [29]

Algorithm	n_1	n_2	Optimal weight
MBFPA	0.788675132828	0.408248295461	263.895843376
DEDS	0.78867513	0.40824828	263.8958434
MVO	0.78860276	0.408453070000000	263.8958499
GOA	0.788897555578973	0.407619570115153	263.895881496069
MFO	0.788244771	0.409466905784741	263.8959797
PSO-DE	0.7886751	0.4082482	263.8958433
SSA	0.788665414	0.408275784444547	263.8958434
MBA	0.885650	0.4085597	263.8958522
WCA	0.788651	0.408316	263.895843
RSSO	0.7886509	0.408305	263.8958420

5.5 Tension/Compression Spring Design

The suggested RSSO lightens the tension/compression spring's load in this case. Shear stress, frequency, and deflection must all be considered when designing the ideal spring configuration. Wire diameter (d), mean coil diameter (D), and number of active coils (N) are the design characteristics displayed in Fig. 7. Subsequent sections outline the challenge of minimization.

$$X = [x_1, x_2, x_3] = [d\ D\ N] \tag{34}$$

$$\text{Minimize } f(X) = (x_3 + 2)\, x_2 x_1^2 \tag{35}$$

$$\text{Subject to } g_1(X) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \le 0 \tag{36}$$

$$g_2(X) = \frac{4x_2^2 - x_1 x_2}{12566 x_1^4 (x_2 x_3^2 - x_1^4)} + \frac{1}{5108 x_1^2} \le 0 \tag{37}$$

$$g_3(X) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \le 0 \tag{38}$$

$$g_4(X) = \frac{x_1 + x_2}{1.5} - 1 \le 0 \tag{39}$$

where x_1, x_2 and x_3 are in the range of:

$0.05, \le x_1 \le 2.00;$

$0.05 \le x_2 \le 1.30;$

$2.00 \le x_3 \le 15.$

The findings of the proposed RSSO along with other algorithms considering both tension and compression of the spring are shown in Table 13 which suggests that the proposed RSSO outperforms the other research.

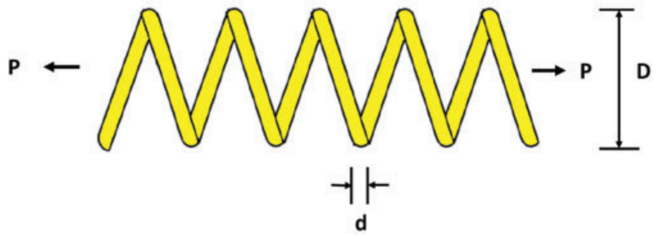

Figure 7: Tension/Compression spring design problem

Table 13: Comparison of optimization results for tension/compression spring design problem

Algorithm	*d*	*D*	*N*	Optimum weight
MFO	0.0518945	0.3642093	10.864219	0.0127669
ALO	0.0518808	0.3647768	10.8891826	0.0125672
MVO	0.0501000	0.316916	14.1454336	0.0128772
SCA	0.0501000	0.3161590	14.2132000	0.0129107
GOA	0.0513728	0.3468863	11.8989078	0.0128705
GWO	0.0517900	0.3567470	11.288950	0.0126660
PSO	0.0521701	0.3718567	10.5067588	0.0125713
WOA	0.0513070	0.3453150	12.0041320	0.0129763
SSA	0.0514070	0.3462150	12.0040420	0.0121763
ESSAWOA	0.0516675	0.3558191	11.3490195	0.0129655
RSSO	0.069221234	0.08457352	3.23350236	0.01211208

5.6 *Pressure Vessel Design*

In order to minimize the total cost while taking material, forming, and welding limits into account, the RSSO is employed to address the design issues associated with pressure vessels. Fig. 8 shows the pressure vessel's structural design, with design parameters such shell thickness (t), head thickness (T), inner radius (R), and length of the head-free cylindrical portion (L) shown. The following is the mathematical formulation of the four restrictions that the design is subject to:

$$X = [x_1, x_2, x_3, x_4] = [T_s, T_h, R, L], \tag{40}$$

$$f(X) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3. \tag{41}$$

Subject to

$$g_1(X) = -x_1 + 0.0193x_3 \le 0, \tag{42}$$

$$g_2(X) = -x_3 + 0.00954x_3 \le 0, \tag{43}$$

$$g_3(X) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0, \tag{44}$$

$$g_4(X) = x_4 - 240 \le 0. \tag{45}$$

variable range: $0 \le x_1, x_2 \le 99, 10 \le x_3, x_4 \le 200.$

Table 14 displays the outcomes produced by the suggested method in addition to other optimization techniques. The optimal solution was attained by the proposed RSSO at the lowest cost.

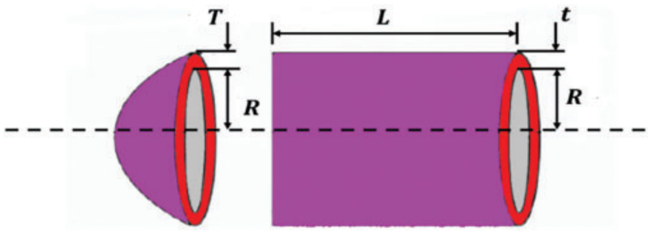

Figure 8: Pressure vessel design problem

Table 14: Comparison of optimization results for pressure vessel design problem

Algorithm	T_s	T_h	R	L	Optimal cost
MFO	0.8126000	0.4385000	42.0994450	176.6375960	6059.7154000
ALO	0.7899151	0.3999634	40.8774169	192.4040606	5904.21989670
MVO	0.8135000	0.4575000	42.0927382	176.7396900	6060.807450
SCA	0.8139210	0.4684392	40.3599386	200.0022000	6272.4912900
GOA	0.8623648	0.42455574	44.5762983	148.1776879	6042.4662583
GWO	0.8135000	0.4355000	42.0892810	176.7598531	6051.5699400
PSO	0.8494716	0.4394000	43.9626591	154.8365465	6016.6894000
WOA	0.8135000	0.4377000	42.0985699	176 0.639998	6059.7412000
SSA	0.7916780	0.3928340	40.9679388	195.91835200	6012.1899500
ESSAWOA	0.7827639	0.3865301	40.5058956	197.4652899	5892.3568603
RSSO	0.07400238	0.002850599	35.868695704	196.40353077	5652.679909778

5.7 Welded Beam Design

Fig. 9 illustrates how the proposed RSSO algorithm is used to optimize the design of a welded beam, with the goal of minimizing its fabrication cost. By optimizing various design parameters, such as shear stress (τ), bending stress (σ), buckling load (Pc), deflection (δ), and side constraints, the overall production cost can be reduced. This optimization problem involves four key design variables: bar thickness (b), height (t), length (l), and weld thickness (h).

$$X = [x_1, x_2, x_3, x_4] = [h\ l\ T\ b] \tag{46}$$

$$f(X) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \tag{47}$$

Subject to

$$g_1(X) = \tau(X) - \tau_{max} \le 0 \tag{48}$$

$$g_2(X) = \sigma - \sigma_{max} \le 0 \tag{49}$$

$$g_3(X) = \delta - \delta_{max} \le 0 \tag{50}$$

$$g_4(X) = x_1 - x_4 \le 0 \tag{51}$$

$$g_5(X) = P - P_C(X) \le 0 \tag{52}$$

$$g_6(X) = 0.125 - x_1 \le 0 \tag{53}$$

$$g_7(X) = 1.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5.0 \le 0 \tag{54}$$

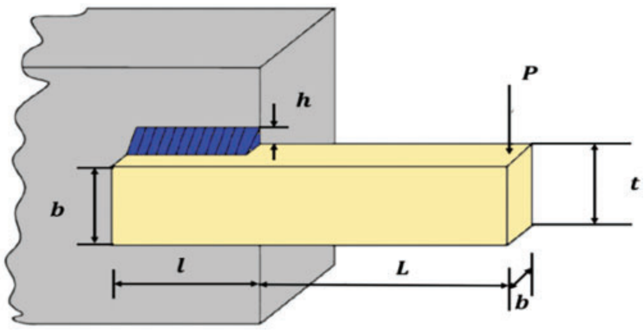

Figure 9: Welded beam design problem

The variable range is taken as follows $0.1 \leq x_1, x_4 \leq 2$ and $0.1 \leq x_2, x_3 \leq 10$, where

$$\tau(X) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \tag{55}$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2} \tag{56}$$

$$\tau'' = \frac{MR}{J} \tag{57}$$

$$M = P\left(L + \frac{x_2}{2}\right) \tag{58}$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \tag{59}$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \tag{60}$$

$$\sigma(X) = \frac{6PL}{Ex_3^2x_4} \tag{61}$$

$$\delta(X) = \frac{6PL^3}{Ex_3^2x_4} \tag{62}$$

$$P_C(X) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \tag{63}$$

$P = 6000lb, L = 14\ in, E = 30 \times 10^6 psi, G = 12 \times 10^6 psi,$
$\delta_{max} = 0.25in, \tau_{max} = 13600\ psi, \sigma_{max} = 30000\ psi.$

The results of the proposed algorithm and other optimization are tabulated in Table 15 which indicates proposed RSSO outperforms the majority of the existing methods.

Table 15: Comparison of optimization results for a welded beam design problem [30]

Algorithm	*h*	*l*	*T*	*b*	Optimal cost
RANDOM	0.457700	4.736900	5.085550	0.6677200	4.856850
SIMPLEX	0.279400	5.622600	7.756900	0.299600	2.537860
APPROX	0.244500	6.216900	8.295600	0.25500	2.3819300
HHO	0.204239	3.535661	9.0277863	0.216187	1.7859906
MFO	0.2057200	3.476930	9.0366650	0.216700	1.7946200
ALO	0.2039936	3.5126449	9.0366268	0.2167896	1.7773787
MVO	0.20634630	3.4736930	9.0455420	0.20568950	1.7823600
SCA	0.2025987	3.6596450	9.2859949	0.204262	1.78892134
GOA	0.20756334	3.449599	9.1118962	0.2911324	1.73746220
GWO	0.2058960	3.4785780	9.1568100	0.2154690	1.7465300
PSO	0.2256801	3.2405599	8.7590948	0.2348262	1.7942319
WOA	0.20569960	3.4845950	9.1364260	0.21685327	1.7359390
SSA	0.2058900	3.4788000	9.0569800	0.2178960	1.7245189
ESSAWOA	0.20569751	3.4755616	9.523665	0.2187923	1.7261397
RSSO	0.228392203	4.372097314	5.263547690	0.2563605396	1.444628808

5.8 *Multi-Plate Disc Clutch Brake Design Problem*

The research also focused on optimizing the design of a multi-plate disc brake system to reduce its weight. The optimization targeted minimizing several factors, including the actuation force required to engage the brakes, the inner and outer radii of the discs, the friction surfaces, and the thickness of the discs. Fig. 10 provides a visual representation of a multi-plate disc clutch, which serves as an analogy for the disc brake system being optimized. The mathematical formulation of this optimization problem is provided below.

$$[n] = [n_1, n_2, n_3, n_4, n_5] = [r_i, r_o, t, F, Z] \tag{64}$$

$$\text{Minimize } f(n) = \pi\left(n_2^2 - n_1^2\right) n_3 (n_5 + 1)\rho \tag{65}$$

Subject to:

$$g_1(n) = n_2 - n_1 - \Delta R \geq 0 \tag{66}$$

$$g_2(n) = L_{max} - (n_5 + 1)(n_3 + \delta) \geq 0 \tag{67}$$

$$g_3(n) = P_{max} - P_{rz} \geq 0 \tag{68}$$

$$g_4(n) = P_{max} v_{srmax} - P_{rz} v_{sr} \geq 0 \tag{69}$$

$$g_5(n) = v_{srmax} - v_{sr} \geq 0 \tag{70}$$

$$g_6(n) = T_{max} - T \geq 0 \tag{71}$$

$$g_7(n) = M_h - sM_s \geq 0 \tag{72}$$

$$g_8(n) = T \geq 0 \tag{73}$$

where $M_h = 2/3\mu n_4 n_5 \frac{n_2^3 - n_1^3}{n_2^2 - n_1^2}$, $w = \frac{\pi n_q}{30}$ rad/s;

$A = \pi\left(n_2^2 - n_1^2\right)$ mm^2, $P_{rz} = \frac{n_4}{A} \frac{\text{N}}{\text{mm}^2}$, $v_{sr} = \frac{\pi R_{rs} n_q}{30}$ mm/s,

$R_{sr} = 2/3\frac{n_2^3-n_1^3}{n_2^2 n_1^2}$, $T = \frac{I_z \pi n_q}{30\left(M_h+M_f\right)}$mm, $\Delta R = 20$ mm, $L_{max} = 30$ mm,
$\mu = 0.6$, $T_{max} = 15s$, $\delta = 0.5$ mm, $s = 1.5$, $M_s = 40$ Nm, $P_{max} = 1$ MPa,
$\rho = 0.0000078$ kg/mm^3, $v_{srmax} = 10$m/s, $T_{max} = 15s$, $n_q = 250$ rpm,
$I_z = 55$ kg m^2, $M_s = 40$ Nm, $M_f = 3$ Nm,
variable range $60 \le n_1 \le 80; 90 \le n_2 \le 110; 1 \le n_3 \le 3;$
$60 \le n_4 \le 1000; 2 \le n_5 \le 9; i = 1, 2, 3, 4, 5.$

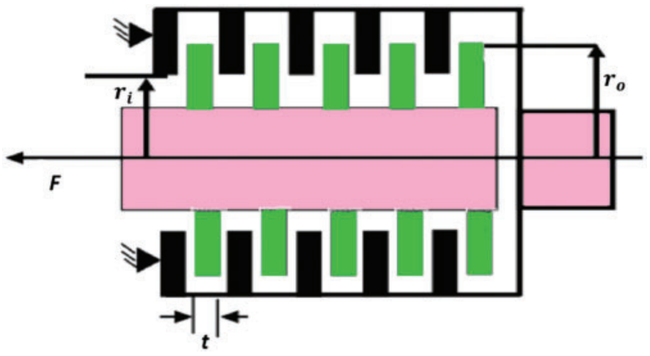

Figure 10: Multi-plate disc clutch brake design problem

As shown in Table 16, the suggested RSSO is included with the results obtained by several algorithms, including TLBO, HHO, WCA, PVS, and AVOA. When compared to the other optimization techniques, the RSSO clearly came out on top.

Table 16: Comparison of results for multi-plate disc clutch brake

Algorithm	TLBO	HHO	WCA	PVS	AVOA	SCA	SSA	RSSO
r_i	70	69.99999999	70	70	69.9999999	69.99999	69.9999	70
r_o	90	90	90	90	90	90	90	90
t	1	1	1	1	1	1	1	1
F	810	1000	910	980	1000	900	900	800
Z	3	2.312781994	3	3	2.312781982	2.653	2.7123	2
Optimal cost	0.313656	0.259768993	0.313656	0.31366	0.259768992	0.28952	0.2967318	0.2352424579

5.9 Rolling Element Bearing Design Problem

The research also investigated the optimization of rolling element bearings, which are commonly used in various mechanical systems. This design problem involved optimizing ten variables while adhering to nine constraints, as shown in Fig. 11. The objective was to maximize the load-carrying capacity of the bearing, which was calculated using a specific formula.

$$\text{Maximize } C_d = \begin{cases} f_c Z^{2/3} D_b^{1.8}, & \text{if } D \le 25.4 \text{ mm} \\ 3.647 f_c Z^{2/3} D_b^{1.4}, & \text{if } D > 25.4 \text{ mm} \end{cases} \tag{74}$$

Subject to:

$$g_1(\vec{z}) = \frac{\varphi_0}{2sin^{-1}(D_b/D_m)} - Z + 1 \geq 0 \tag{75}$$
$$g_2(\vec{z}) = 2D_b - K_{Dmin}(D-d) > 0 \tag{76}$$
$$g_3(\vec{z}) = K_{Dmax}(D-d) - 2D_b \geq 0 \tag{77}$$
$$g_4(\vec{z}) = \zeta B_w - D_b \leq 0 \tag{78}$$
$$g_5(\vec{z}) = D_m - 0.5(D+d) \geq 0 \tag{79}$$
$$g_6(\vec{z}) = (0.5+e)(D+d) - D_m \geq 0 \tag{80}$$
$$g_7(\vec{z}) = 0.5(D - D_m - D_b) - \varepsilon D_b \geq 0 \tag{81}$$
$$g_8(\vec{z}) = f_i \geq 0.515 \tag{82}$$
$$g_9(\vec{z}) = f_o \geq 0.515 \tag{83}$$

where

$$f_c = 37.91\left[1+\left\{1.04\left(\frac{1-\gamma}{1+\gamma}\right)^{1.72}\left(\frac{f_i(2f_o-1)}{f_o(2f_i-1)}\right)^{0.41}\right\}^{10/3}\right]^{-0.3} \times \left[\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{(1+\gamma)^{1/3}}\right]\left[\frac{2f_i}{2f_i-1}\right]^{0.41},$$

$$x = \left[\{(D-d)/2 - 3(T/4)\}^2 + \{D/2 - T/4 - D_b\}^2 - \{d/2 + T/4\}^2\right],$$

$$y = 2\{(D-d)/2 - 3(T/4)\}\{D/2 - T/4 - D_b\},$$

$$\varphi_0 = 2\pi - cos^{-1}\left(\frac{x}{y}\right),$$

$$\gamma = \frac{D_b}{D_m}, f_i = \frac{r_i}{D_b}, f_o = \frac{r_o}{D_b}, T = D - d - 2D_b,$$

$$D = 160, d = 90, B_w = 30, r_i = r_o = 11.0330,$$

$$0.5(D+d) \leq D_m \leq 0.6(D+d),$$

$$0.15(D-d) \leq D_b \leq 0.45(D-d),$$

$$4 \leq Z \leq 50, 0.515 \leq f_i \ and \ f_o \leq 0.6,$$

$$0.4 \leq K_{Dmin} \leq 0.5, 0.6 \leq K_{Dmax} \leq 0.7,$$

$$0.3 \leq \varepsilon \leq 0.4, 0.02 \leq e \leq 0.1,$$

$$0.6 \leq \zeta \leq 0.85.$$

The suggested RSSO has been compared to other algorithms' outputs, including AVOA, PVS, TLBO, and GA. Table 17 displays the results of the comparison, which indicate that the SSCA outperformed the other algorithms.

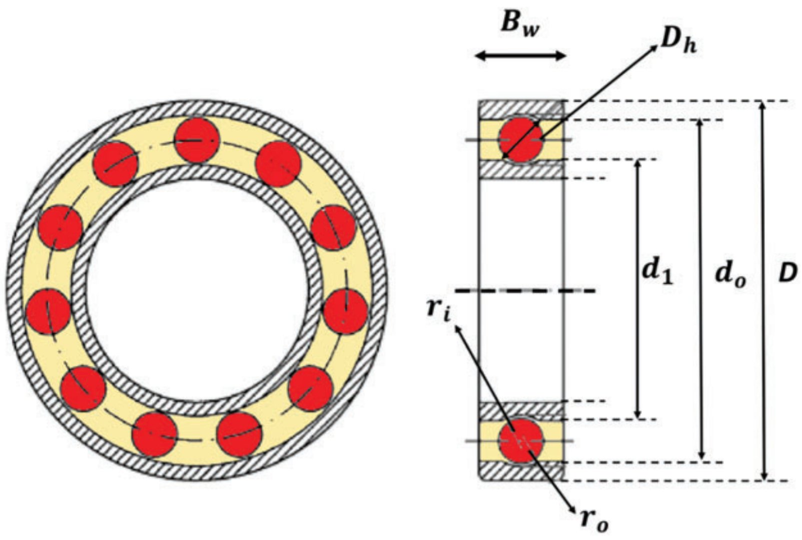

Figure 11: Rolling element bearing design problem

Table 17: Comparison of optimization results for rolling element bearing design problem

Algorithms	PVS	TLBO	GA	AVOA	SSA	SCA	RSSO
D_m	125.719060	125.7191	125.717100	125.722717	125.6523	125.6823	125.50000
D_b	21.425590	21.42559	21.423000	21.423294	21.421236	21.41256	21.50000
Z	11.000000	11.000000	11.000000	11.001162	11.005236	11.000126	12.50000
f_i	0.515000	0.515000	0.515000	0.515000	0.51426	0.513269	0.5154
f_o	0.515000	0.515000	0.515000	0.515000	0.510368	0.108712	0.5187
K_{Dmin}	0.400430	0.424266	0.415900	0.404428	0.403526	0.41365	0.4373
K_{Dmax}	0.680160	0.633948	0.651000	0.618679	0.62589	0.63541	0.6552
ε	0.300000	0.300000	0.300043	0.300000	0.35123	0.36234	0.3936
e	0.079990	0.068858	0.022300	0.0691299	0.075631	0.0758531	0.0809
ξ	0.700000	0.799498	0.751000	0.602470	0.746321	0.75498	0.8394
Maximum cost	81859.741210	81859.74	81843.30	85539.15785	86524.8461	86684.43618	87790.506682

5.10 Cantilever Structure Problem

The cantilever beam used in this work has five hollow square cross-sections, as shown in Fig. 12. To find the most economical solution, we'll assume that the thickness is constant and look at the other six parameters in Fig. 12. Following are the mathematical concepts used in this design issue:

$$\text{Minimize } f(n) = 0.6224\left(n_1 + n_2 + n_3 + n_4 + n_5\right) \tag{84}$$

Subject to:

$$g(n) = \frac{61}{n_1^3} + \frac{37}{n_2^3} + \frac{19}{n_3^3} + \frac{7}{n_4^3} + \frac{1}{n_5^3} \le 1 \tag{85}$$

variable range $0.01 \le n_1, n_2, n_3, n_4, n_5 \le 100$.

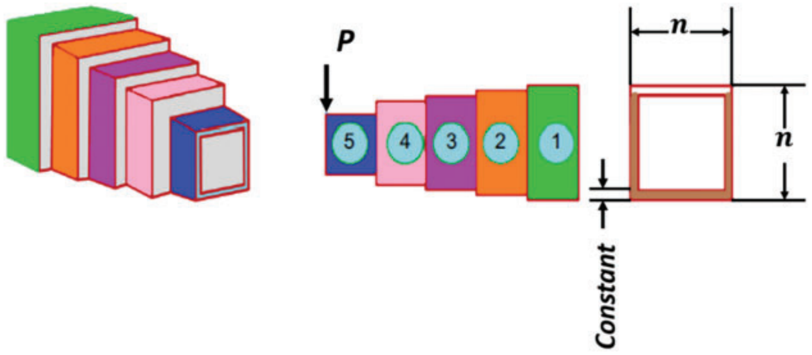

Figure 12: Cantilever design problem

Table 18 shows the results of comparing the proposed RSSO with different optimization strategies. The superior performance of the RSSO compared to the other optimization techniques is clearly demonstrated in Table 18.

Table 18: Comparison of optimization results for cantilever structure problem [31]

Algorithms	n_1	n_2	n_3	n_4	n_5	Optimum cost
SMA	6.017756	5.310893	4.493758	3.501107	2.150158	1.339958
MFO	5.9831	5.3168	4.4974	3.5137	2.1617	1.33999
SOS	6.0189	5.3032	4.4959	3.4901	2.1557	1.33997
CS	6.0090	5.3048	4.5024	3.5078	2.1506	1.33998
MMA	6.0101	5.3001	4.4901	3.4901	2.1501	1.3410
GCA	6.0101	5.3011	4.4901	3.4901	2.1501	1.3410
SSA	6.0101	5.3024	4.46813	3.4823	2.1506	1.3325
SCA	6.0100	5.3055	4.46277	3.48922	2.1513	1.32956
RSSO	6.10078650	5.31100200	4.249327158	3.51306250	2.15015900	1.3272267447

6 Conclusion

This research focused on enhancing the search and convergence capabilities of the SSO algorithm. To achieve this, three key modifications were made to the original algorithm. The following conclusions have been drawn from the above research.

- The proposed optimization method incorporates three key features: OBL, the Cauchy mutation strategy, and position clamping. These features enhance the balance between exploration and exploitation phases compared to the basic SSO algorithm, resulting in a more effective search process.
- The proposed RSSO algorithm demonstrated superior performance compared to other leading optimization techniques when evaluated on various benchmark functions, including unimodal, multi-modal, and fixed-dimensional multi-modal functions. The results, measured by average and standard deviation, showed that RSSO effectively avoids getting stuck in local optima (exploitation) while simultaneously exploring a wider range of potential solutions (exploration).
- RSSO proves to be a powerful tool for solving complex optimization problems. Both qualitative and statistical analyses demonstrate that RSSO outperforms other algorithms in terms of both speed of

convergence and quality of the final solution. This makes RSSO a valuable tool for computer-aided design and engineering applications.

- The effectiveness of the RSSO algorithm has been validated not only on theoretical benchmark problems but also on a range of real-world engineering design problems. In all these cases, RSSO has demonstrated superior performance compared to other algorithms, highlighting its broad applicability and potential for solving complex engineering challenges.
- Future research directions could include investigating adaptive parameter tuning mechanisms to further enhance RSSO's performance across diverse problem domains. Developing hybrid algorithms that integrate RSSO with other metaheuristic techniques to capitalize on their respective strengths. Applying RSSO to emerging fields such as machine learning hyperparameter optimization and quantum computing optimization problems.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Sumika Chauhan: Conceptualisation, Investigation, Data curation, Writing—original draft. Govind Vashishtha: Conceptualisation, Investigation, Writing—original draft. Riya Singh: Conceptualisation, Investigation, Supervision, Writing—original draft, review & editing. Divesh Bharti: Supervision, Writing—review & editing. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data will be available on reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Ding Z, Huang Z, Pang M, Bai G, Wang Q, Han B. Optimized design of uniform-field coils system with a large uniform region by particle swarm optimization algorithm. Measurement. 2024;234(2):114614. doi:10.1016/j.measurement.2024.114614.
2. Li D, Li C, Yang J, Chen Z, Liu X, Wang X, et al. Bayesian optimization-attention-feedforward neural network based train traction motor-gearbox coupled noise prediction. Measurement. 2024;238:115323. doi:10.1016/j.measurement.2024.115323.
3. Lu J, Yue J, Zhu L, Wang D, Li G. An improved variational mode decomposition method based on the optimization of salp swarm algorithm used for denoising of natural gas pipeline leakage signal. Measurement. 2021;185(454):110107. doi:10.1016/j.measurement.2021.110107.
4. Khunkitti S, Siritaratiwat A, Premrudeepreechacharn S, Chatthaworn R, Watson NR. A hybrid DA-PSO optimization algorithm for multiobjective optimal power flow problems. Energies. 2018;11(9):2270. doi:10.3390/en11092270.
5. Abbassi A, Ben Mehrez R, Touaiti B, Abualigah L, Touti E. Parameterization of photovoltaic solar cell double-diode model based on improved arithmetic optimization algorithm. Optik. 2022;253(2):168600. doi:10.1016/j.ijleo.2022.168600.
6. Vashishtha G, Kumar R. Feature selection based on gaussian ant lion optimizer for fault identification in centrifugal pump. In: Gupta VK, Amarnath C, Tandon P, Ansari MZ, editors. Recent advances in machines and mechanisms. Singapore: Springer Nature; 2023. p. 295–310. doi:10.1007/978-981-19-3716-3_23.
7. Chen J, Aurangzeb M, Iqbal S, Shafiullah M, Harrison A. Advancing EV fast charging: addressing power mismatches through P2P optimization and grid-EV impact analysis using dragonfly algorithm and reinforcement learning. Appl Energy. 2025;394(2):126157. doi:10.1016/j.apenergy.2025.126157.

8. Gai J, Shen J, Hu Y, Wang H. An integrated method based on hybrid grey wolf optimizer improved variational mode decomposition and deep neural network for fault diagnosis of rolling bearing. Measurement. 2020;162:107901. doi:10.1016/j.measurement.2020.107901.
9. Anslam Sibi S, Sherly Puspha Annabel L. Network lifetime improvement in wireless sensor networks using energy-efficient bat-moth flame optimization technique. Sci Rep. 2025;15(1):18065. doi:10.1038/s41598-025-88550-y.
10. Grisales-Noreña LF, Botero-Gómez V, Bolaños RI, Moreno-Gamboa F, Sanin-Villa D. An effective parameter estimation on thermoelectric devices for power generation based on multiverse optimization algorithm. Results Eng. 2025;25(1):104408. doi:10.1016/j.rineng.2025.104408.
11. Neggaz N, Ewees AA, Elaziz MA, Mafarja M. Boosting salp swarm algorithm by sine cosine algorithm and disrupt operator for feature selection. Expert Syst Appl. 2020;145(1181):113103. doi:10.1016/j.eswa.2019.113103.
12. Özbay F, Özbay E, Gharehchopogh F. An improved artificial rabbits optimization algorithm with chaotic local search and opposition-based learning for engineering problems and its applications in breast cancer problem. Comput Model Eng Sci. 2024;141(2):1067–110. doi:10.32604/cmes.2024.054334.
13. Shehadeh HA, Ahmedy I, Idris MYI. Sperm swarm optimization algorithm for optimizing wireless sensor network challenges. In: Proceedings of the 6th International Conference on Communications and Broadband Networking. New York, NY, USA: Association for Computing Machinery; 2018. p. 53–9. doi:10.1145/3193092.3193100.
14. Wang GG, Deb S, Gandomi AH, Alavi AH. Opposition-based krill herd algorithm with Cauchy mutation and position clamping. Neurocomputing. 2016;177:147–57. doi:10.1016/j.neucom.2015.11.018.
15. Kumari CL, Kamboj VK, Bath SK, Tripathi SL, Khatri M, Sehgal S. A boosted chimp optimizer for numerical and engineering design optimization challenges. Eng Comput. 2022;39(4):2463–514. doi:10.1007/s00366-021-01591-5.
16. Kaucic M, Piccotto F, Sbaiz G, Valentinuz G. A hybrid level-based learning swarm algorithm with mutation operator for solving large-scale cardinality-constrained portfolio optimization problems. Inf Sci. 2023;634(4):321–39. doi:10.1016/j.ins.2023.03.115.
17. Alruwais N, Alabdulkreem E, Mahmood K, Marzouk R, Assiri M, Abdelmageed AA, et al. Hybrid mutation moth flame optimization with deep learning-based smart fabric defect detection. Comput Electr Eng. 2023;108(3):108706. doi:10.1016/j.compeleceng.2023.108706.
18. Zhou X, Chen Y, Wu Z, Heidari AA, Chen H, Alabdulkreem E, et al. Boosted local dimensional mutation and all-dimensional neighborhood slime mould algorithm for feature selection. Neurocomputing. 2023;551(6):126467. doi:10.1016/j.neucom.2023.126467.
19. Yu X, Jiang N, Wang X, Li M. A hybrid algorithm based on grey wolf optimizer and differential evolution for UAV path planning. Expert Syst Appl. 2023;215:119327. doi:10.1016/j.eswa.2022.119327.
20. Vashishtha G, Kumar R. Autocorrelation energy and aquila optimizer for MED filtering of sound signal to detect bearing defect in Francis turbine. Meas Sci Technol. 2021;33(1):15006. doi:10.1088/1361-6501/ac2cf2.
21. Chauhan S, Singh M, Aggarwal AK. Investigative analysis of different mutation on diversity-driven multi-parent evolutionary algorithm and its application in area coverage optimization of WSN. Soft Comput. 2023;27(14):9565–959. doi:10.1007/s00500-023-08090-3.
22. Wang J, Wang W, Chau K, Qiu L, Hu X, Zang H, et al. An improved golden jackal optimization algorithm based on multi-strategy mixing for solving engineering optimization problems. J Bionic Eng. 2024;21(2):1092–115. doi:10.1007/s42235-023-00469-0.
23. Abualigah L, Elaziz MA, Khasawneh AM, Alshinwan M, Ibrahim RA, Al-Qaness MAA, et al. Meta-heuristic optimization algorithms for solving real-world mechanical engineering design problems: a comprehensive survey, applications, comparative analysis, and results. Neural Comput Appl. 2022;34(6):4081–110. doi:10.1007/s00521-021-06747-4.
24. Das A, Namtirtha A, Dutta A. Lévy-Cauchy arithmetic optimization algorithm combined with rough K-means for image segmentation. Appl Soft Comput. 2023;140(1):110268. doi:10.1016/j.asoc.2023.110268.
25. Nadimi-Shahraki MH, Zamani H, Fatahi A, Mirjalili S. MFO-SFR: an enhanced moth-flame optimization algorithm using an effective stagnation finding and replacing strategy. Mathematics. 2023;11(4):862. doi:10.3390/math11040862.

26. Savsani P, Savsani V. Passing vehicle search (PVS): a novel metaheuristic algorithm. Appl Math Model. 2016;40:3951–78. doi:10.1016/j.apm.2015.10.040.
27. Shehadeh HA. A hybrid sperm swarm optimization and gravitational search algorithm (HSSOGSA) for global optimization. Neural Comput Appl. 2021;33(18):11739–52. doi:10.1007/s00521-021-05880-4.
28. Gupta S, Deep K, Mirjalili S, Kim JH. A modified sine cosine algorithm with novel transition parameter and mutation operator for global optimization. Expert Syst Appl. 2020;154(3):113395. doi:10.1016/j.eswa.2020.113395.
29. Wang S, Liu Q, Liu Y, Jia H, Abualigah L, Zheng R, et al. A hybrid SSA and SMA with mutation opposition-based learning for constrained engineering problems. Comput Intell Neurosci. 2021;2021(1):6379469. doi:10.1155/2021/6379469.
30. Wang Y, Cai Z, Zhou Y, Fan Z. Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. Struct Multidiscip Optim. 2009;37(4):395–413. doi:10.1007/s00158-008-0238-3.
31. Li S, Chen H, Wang M, Heidari AA, Mirjalili S. Slime mould algorithm: a new method for stochastic optimization. Future Gener Comput Syst. 2020;111:300–23. doi:10.1016/j.future.2020.03.055.