

## SLIME MOULD ALGORITHM-BASED MUTATED CHAOTIC LOCAL SEARCH FOR GLOBAL OPTIMIZATION AND ENGINEERING APPLICATIONS

ADEL GOT<sup>1</sup>, DJAAFAR ZOUACHE<sup>2</sup>, NAILA AZIZA HOUACINE<sup>1,\*</sup>  AND HABIBA DRIAS<sup>1</sup> 

**Abstract.** Swarm intelligence has gained increasing interest, with the Slime Mould Algorithm (SMA) standing out as a promising yet convergence-prone metaheuristic. To address this limitation, we propose an improved variant, called mSMACLS, which integrates a mutated Chaotic Local Search (mCLS) strategy into SMA. The mCLS mechanism selectively perturbs dimensions of the global best solution using chaotic sequences, enhancing local exploitation while maintaining global exploration. We evaluate 10 chaotic maps to identify the most effective configuration and conduct extensive experiments on 23 standard benchmark functions and 10 more complex optimization problems. The proposed algorithm is further validated on 15 challenging benchmark problems from the CEC2017 competition on constrained single-objective optimization. Results show that the Logistic chaotic map provides the best performance within the mCLS framework. Overall, mSMACLS consistently outperforms six well-known metaheuristics across most test cases, demonstrating improved convergence speed and solution quality. Finally, the algorithm’s applicability is tested on 3 real-world engineering design problems, and the results were very encouraging. The source code is publicly available at <https://www.mathworks.com/matlabcentral/fileexchange/182303-slime-mould-algorithm-based-mutated-chaotic-local-search>.

**Mathematics Subject Classification.** 68T20.

Received December 6, 2024. Accepted October 5, 2025.

### 1. INTRODUCTION

Solving an optimization problem consists of minimizing or maximizing, depending on the context, a mathematical formula generally known as objective function, and it is expressed by a set of decision variables or dimensions. Knowing that the objective function is the mathematical description of the problem to be solved, the set of variables represents the size of this problem. Therefore, the main goal when solving any optimization problem is to find the best configuration of variables that leads to the global best objective value. However, the real-world optimization problems are generally defined in non-linear and high-dimensional search spaces that comprises several local optima, leading in serious challenges to find the global optimum. Hence, dealing with these challenges *via* traditional and deterministic methods seems to be not only very hard, but totally impracticable. For this reason, population-based metaheuristics have become an efficient and powerful choice

---

*Keywords.* Swarm intelligence, slime mould algorithm, chaotic local search, mutation, optimization problems.

<sup>1</sup> LRIA, Faculty of Informatics, USTHB, Algiers, Algeria.

<sup>2</sup> Computer Science Department, University of Mohamed El Bachir El Ibrahimi, Bordj Bou Arreridj, Algeria.

\*Corresponding author: [naila.houacine@usthb.edu.dz](mailto:naila.houacine@usthb.edu.dz); [nhouacine@usthb.dz](mailto:nhouacine@usthb.dz)

for solving complex optimization problems [1]. Indeed, one of the strengths of metaheuristics is their ability to approximate the best solution in reasonable time, and their efficiency to deal with any optimization problem whatever its nature (non-linear, non-convex, high-dimensional, etc). Broadly speaking, and during its optimization process, any metaheuristic performs two tasks universally known as exploration and exploitation [2]. The aim of exploration is to visit unexplored regions in the search space in order to increase the diversity of solutions, while the aim of exploitation is to perform some local searches in the neighborhood of a promising region in order to enhance the quality of the current best solution [3].

Inspired mainly from the collective behavior of animals that live in groups, Swarm Intelligence (SI) are one of the most popular metaheuristics which have attracted much attention [4, 5]. Indeed, many SI-based algorithms have been developed in the last decade such as Grey Wolf Optimizer (GWO) [6], Whale Optimization Algorithm (WOA) [7], Manta Ray Foraging Optimizer (MRFO) [8], Aquila Optimizer (AO) [9], and more recently Artificial Rabbits Optimization (ARO) [10], Coati Optimization Algorithm (COA) [11], Sand Cat Swarm Optimization (SCSO) [12], Greylag Goose Optimization (GGO) [13], Moth Flame Optimizer (MFO) and its variants [14, 15], the prominent Parrot Optimizer (PO) inspired by parrots' (*pyrrhura molinaes*' ) behavior [16], the interesting Artemisinin Optimization (AO) which mimics the process of artemisinin medicine therapy for malaria [17], and the Polar Lights Optimization (PLO) introduced in [18] and the well-reputed RIME algorithm proposed in [19]. These algorithms have received significant attention and they have been widely adopted to handle many practical problems such as multi-objective and engineering design problems [20–23], machine learning [24–26], wireless sensor networks [27], renewable energy [28], ambulance dispatching and relocation problem [29–31], online sequential learning machine [32], the diagnose faults of rolling bearings [33], the detection of faults in gears [34], etc. However, despite this effectiveness in solving various real-world situations, and due to their stochastic behavior, the improvement of Swarm Intelligence metaheuristics is still an open topic for many scholars until now, as the recent Non-dominated sorting advanced butterfly optimizer introduced in [35] to deal with multi-objective problems, the MCF-FFA algorithm proposed in [36] to solve the Traveling Salesman Problem (TSP) *via* an improved Farmland Fertility Algorithm, the prominent Multi-objective imperialist competitive algorithm (NSICA) which treats feature selection as a multi-objective problem for effective cardiac arrhythmia diagnosis [37], the improved Artificial Rabbits Optimizer through CLS and OBL strategies (COARO) applied to different engineering design problems and its binary version BCOARO applied in Breast Cancer Problem [38]. Under this context, non-linear dynamics, and chaos theory in particular, has been widely adopted to improve the performance of several SI-based metaheuristics [39–41]. Indeed, the ergodicity and the non-repetition characterizing the chaotic maps allow them to increase the exploration and exploitation abilities of SI algorithms. Typically speaking, the exploration is often enhanced by replacing some random parameters of the algorithm with chaotic sequences. Here, it is worth mentioning that some studies have adopted the chaos theory to improve the diversity of the initial population, hence, improving the exploration ability. On the other hand, the exploitation is often enhanced by incorporating a Chaotic Local Search (CLS) into the algorithm itself.

For instance, the authors in [42] proposed the Chaotic Whale Optimization Algorithm (CWOA) which replaces the controlling exploration/exploitation parameter  $p$  in the basic WOA by a sequence of chaotic numbers. Among the 10 investigated chaotic maps in this study, the results shown that only Circle and Iterative maps which have failed to improve the basic WOA, while the Tent map has significantly improved the performance of WOA. However, we haven't any idea on its performance on more complex functions and practical problems as it was benchmarked only on twenty classical test functions. The same idea was applied in [43] to adjust the key parameter  $a$  of GWO, where Chebyshev map is selected as the best chaos map. The results shown that the proposed CGWO clearly outperforms its competitors in benchmark functions and real-world problems. In CGOA [44], chaos theory is embedded to refine the parameters  $c_1$  and  $c_2$  of the standard GOA algorithm. The authors have drew an interesting conclusion, which suggests that refining  $c_1$  or  $c_2$  separately cannot improve the performance of the GOA considerably. In contrast, Circle map was the best chaotic GOA for updating  $c_1$  and  $c_2$  parameters. In [45], Tent chaotic map is employed to initialize the dragonflies to maintain a good diversity along the initial population at the hope of improving the search efficiency of the PDA algorithm. The obtained results have demonstrated that the proposed TPDA can converge to the global optima with a

TABLE 1. Some developed chaotic metaheuristics.

Chaotic version	Improved algorithm	Improved strategy	Optimal chaotic map
CKHA [48]	Krill Herd Algorithm	Exploration (param.chaos)	Singer map
CWOA [42]	Whale Optimization Algorithm	Exploration (param.chaos)	Tent map
CGWO [43]	Grey Wolf Optimizer	Exploration (param.chaos)	Chebyshev map
CGOA [44]	Grasshopper Optimization Algorithm	Exploration (param.chaos)	Circle map
CSSA [49]	Salp Swarm Algorithm	Exploration (param.chaos)	Logistic map
CMRFO [50]	Manta Ray Foraging Optimizer	Exploration (param.chaos)	Singer map
ACPSO [51]	Particle Swarm Optimization	Exploration (param.chaos)	Logistic map
CSCPS [52]	Sand Cat Optimizer	Exploration (param.chaos)	Logistic map
CSPSO [53]	Particle Swarm Optimization	Exploration (pop.init)	Sinusoidal map
IAHA [54]	Artificial Hummingbird Algorithm	Exploration (pop.init)	Chebyshev map
eHHO [55]	Harris Hawks Optimizer	Exploration (pop.init)	Logistic map
AChOA [56]	Chimp Optimization Algorithm	Exploration (pop.init)	Tent map
IRSA [57]	Reptile Search Algorithm	Exploration (pop.init)	Circle map
CASFO [58]	Sailfish Optimizer	Exploration (pop.init)	Tent map
TPDA [45]	Dragonfly Algorithm	Exploration (pop.init)	Tent map
ISSA [59]	Sparrow Search Algorithm	Exploration (pop.init)	Tent map
CJADE-M [60]	Adaptive Differential Evolution	Exploitation (CLS)	Multiple chaotic maps
QOCALO [47]	Ant Lion Optimizer	Exploitation (quasi-oppositional CLS)	Logistic map
OCS-GWO [61]	Grey Wolf Optimizer	Exploitation (opposition-based CLS)	Logistic map
DECLS [62]	Differential Evolution	Exploitation (CLS)	Logistic map
CLSGMFO [46]	Moth Flame Optimizer	Exploitation (CLS)	Logistic map
mBES [63]	Bald Eagle Search	Exploitation (CLS)	Logistic map

good robustness and stability. However, the used population classification evolution strategy has increased the computational complexity of the algorithm. Regarding the use of Chaotic Local Search (CLS), CLSGMFO algorithm is developed in [46] by incorporating a CLS Logistic-based map into the MFO algorithm to handle local search more efficiently. The algorithm was applied to perform financial predictions, and the results were very satisfactory in terms of accuracy. In [47], the CLS strategy is applied around the global best solution of ALO algorithm, which leads in better exploitation ability. Indeed, the findings on benchmark functions and three real structural problems show the high performance of the proposed QOCALO compared to the standard ALO and other metaheuristics.

Obviously, apart from the above mentioned references, there is a lot of interesting works in this research direction in literature, and to sum up, Table 1 summarizes some prominent contributions on the improvement of metaheuristics through chaos theory.

Slime Mould Algorithm (SMA) is a recent SI-based metaheuristic which has been developed by simulating a special social behavior observed in Slime mould [64]. Despite its recent appearance, SMA has shown high performance in various optimization situations. However, it is still able to be improved. Hence, this paper presents a new improved version of the SMA called mSMACLS (for mutated Chaotic Local Search SMA). The main contributions of the proposed chaotic-based algorithm are summarized as follows:

- A novel and effective local search strategy based on chaos theory is proposed and integrated into the standard SMA algorithm to improve its performance.
- The proposed mCLS strategy acts as a mutation operator on the current best solution to improve the exploitation ability of SMA.

- The ten most popular chaotic maps are investigated and compared to select the appropriate chaotic map for the proposed mSMACLS algorithm.
- Expensive numerical and graphical comparison is performed against thirteen well-known algorithms on twenty-three classical and twenty-five complex test functions are employed.
- The performance of the proposed mSMACLS in handling real-world problems is tested on three practical engineering problems.

The remainder of paper is structured as follows: Section 2 presents the basic SMA algorithm. Section 3 describes, with detailed explanations, the proposed mSMACLS algorithm. Section 4 outlines the experimental simulations. At the end, Section 5 concludes the the paper.

## 2. SLIME MOULD ALGORITHM

As mentioned above, Slime Mould Algorithm is a recent population-based Swarm Intelligence metaheuristic inspired by the oscillation strategy adopted by slime moulds during their attempts to find the optimal path towards the source-food in nature [64]. Moreover, SMA algorithm employs adaptive weights to model the positive and negative feedback generated during the propagation wave of slime moulds. The following subsections summarize the mathematical model of SMA algorithm.

### 2.1. Approach food mechanism

In this phase, slime moulds identify the source-food based on its odor in the air. This behavior is expressed by the following system of equations:

$$x_i(t+1) = \begin{cases} Gbest + b \cdot (w \cdot x_1(t) - x_2(t)) & \text{if } \text{rand}(0, 1) < p \\ c \cdot x_i(t) & \text{Otherwise} \end{cases} \quad (1)$$

where,  $x_i(t)$  is the position of  $i$ th slime mould at iteration  $t$ ,  $Gbest$  is the global best solution obtained to date,  $x_1$  and  $x_2$  are two individuals randomly selected from the population of slime moulds,  $c$  is a parameter which decreases linearly from 1 to 0, and  $b$  is another parameter in the interval  $[-a, a]$ , so as:

$$a = \text{arctanh} \left( - \left( \frac{t}{T} \right) + 1 \right) \quad (2)$$

where,  $T$  is the maximum number of iterations. On the other hand,  $p$  and  $w$  in equation (1) are given as follows:

$$p = \tanh |S(i) - \text{BF}| \quad (3)$$

where,  $S(i)$  is the fitness of the  $i$ th search agent, and BF is best fitness found so far.

$$w(\text{SmeelIndex}(t)) = \begin{cases} 1 + \text{rand}(0, 1) \cdot \log \left( \frac{\text{BF} - S(i)}{\text{BF} - \text{WF}} + 1 \right), & \text{condition} \\ 1 - \text{rand}(0, 1) \cdot \log \left( \frac{\text{BF} - S(i)}{\text{BF} - \text{WF}} + 1 \right), & \text{others} \end{cases} \quad (4)$$

$$\text{SmeelIndex} = \text{sort}(S) \quad (5)$$

where, BF and WF represent the best and the worst fitness values, respectively,  $\text{SmeelIndex}$  stands for the sequence of fitness values, and *condition* indicates that  $S(i)$  ranks first half of the search agents.

## 2.2. Warp food mechanism

Here, and if the food density is low, slime mould moves towards another direction in the search space. This movement is given by equation (6).

$$x_i(t+1) = \begin{cases} \text{rand}(0,1) \cdot (ub - lb) + lb & \text{rand}(0,1) < z \\ Gbest + b \cdot (w \cdot x_1(t) - x_2(t)) & z \leq \text{rand}(0,1) < p \\ c \cdot x_i(t) & \text{rand}(0,1) \geq p \end{cases} \quad (6)$$

where,  $z \in [0,0.1]$  and it is used to control the exploration and exploitation of the search space during the optimization process.  $lb$  and  $ub$  are the upper and lower bounds of the search space, respectively.

## 3. THE PROPOSED MSMACLS ALGORITHM

As in almost all population-based SI-based metaheuristics, and during the exploitation phase, the search agents of SMA algorithm update their current positions with respect to the position of the best search agent ( $Gbest$ ) found until now. Therefore, increasing the strength of SMA algorithm to exploit the most promising regions in the search space can increase the possibility to improve the quality of the optimal solution in the next iterations, hence, improving the performance of the algorithm in general manner. Under this context, the use of a local search strategy seems to be more effective to improve the exploitation ability of Slime Mould Algorithm.

In this paper, and as mentioned before, a local search strategy based on the chaos theory is incorporated into the basic SMA algorithm to improve its performance. The proposed mutated Chaotic Local Search (mCLS) is used to refine the current global best solution  $Gbest$ . Here, it should be noted that the adopted CLS acts as a mutation operator. That is to say, it is not applied to all dimensions of the current  $Gbest$ , but it is applied only to some dimensions of  $Gbest$  based on a predefined mutation probability  $P_m$  ( $P_m = 50\%$  in our case), which makes the difference between our strategy and other existing CLS strategies in literature. The main reason behind this is because CLS is an iterative method, and applying it to all dimensions of  $Gbest$  can lead the search to go very far from  $Gbest$  after few iterations, hence, losing the main objective behind using CLS strategy (which is performing a search in the neighborhood of  $Gbest$ ). Accordingly, applying CLS just on some dimensions of  $Gbest$  lead to maintain the role for which CLS is adopted, hence, generating a new candidate solution in the neighborhood of  $Gbest$ . The implemented mCLS in the proposed mSMACLS algorithm can be expressed by:

$$x'_d = Gbest_d + r(ub - lb)(c_{\text{rand}} - 0.5) \quad (7)$$

where,  $x'$  is the new candidate solution and  $x'_d$  is its  $d$ th-dimension generated by CLS.  $Gbest_d$  is the  $d$ th-dimension of  $Gbest$  on which the mCLS will be applied,  $d$  is in  $[1, D]$ , where  $D$  is the size of the problem in consideration.  $r$  is the CLS radius initialized to 0.0001, and adjusted over iterations by  $r(t+1) = 0.988(t)$ .  $ub$  and  $lb$  are the upper and lower bounds of the search space, respectively.  $c_{\text{rand}}$  stands for a chaotic number selected randomly from the adopted chaotic sequence, and it is removed temporary from the chaotic sequence to avoid select it again during the local iterations. The chaotic sequence is generated by using one of the 10 chaotic maps listed in Table 2. They are also graphically illustrated for 100 iterations in Figure 1.

Another important perspective which should be mentioned is that the execution of mCLS stops when a maximum number of iterations  $t_{cls}$  is reached. However, the proposed mCLS can improve the current global best solution  $Gbest$  before reaching this maximum local iterations  $t_{cls}$ , which may leads in unwanted premature convergence. Hence, to avoid, as much as possible, this issue, mCLS procedure is forced to be stopped once a better solution is found even if the maximum local iterations is not reached. The general process of proposed mCLS is presented in Algorithm 1. Furthermore, Algorithm 2 outlines the pseudo-code of the proposed mSMACLS algorithm.

Accordingly, the algorithm start the optimization by generating randomly the initial population of candidate solutions. Then, and over iterations, each search agent is evaluated to update the global best solution  $Gbest$ .

TABLE 2. Details of the ten chaotic maps.

No	Map name	Map equation
1	Chebyshev map	$c_{k+1} = \cos(0.5 \cos^{-1} c_k)$
2	Circle map	$c_{k+1} = c_k + 0.5 - \frac{1.1}{\pi} \sin(2\pi c_k) \bmod(1)$
3	Gauss/mouse map	$c_{k+1} = 0$ if $c_k = 0$ , $c_{k+1} = \frac{1}{c_k} \bmod(1)$ else
4	Iterative map	$c_{k+1} = \text{abs}\left(\sin\left(\frac{p}{c_k}\right)\right)$ , $p \in (0, 1)$
5	Logistic map	$c_{k+1} = 4 \times c_k(1 - c_k)$
6	Piecewise map	$c_{k+1} = \left\{ \begin{array}{l} \frac{c_k}{0.4} \text{ if } 0 \leq c_k < 0.4, \frac{c_k - 0.4}{0.1} \text{ if } 0.4 \leq c_k < 0.5, \frac{0.6 - c_k}{0.1} \text{ if } 0.5 \leq \\ c_k < 0.6, \frac{1 - c_k}{0.4} \text{ if } 0.6 \leq c_k < 1 \end{array} \right\}$
7	Sine map	$c_{k+1} = \sin(\pi c_k)$
8	Singer map	$c_{k+1} = 1.073(7.86c_k - 23.31c_k^2 + 28.75c_k^3 - 13.302875c_k^4)$
9	Sinusoidal map	$c_{k+1} = 2.3c_k^2 \sin(\pi c_k)$
10	Tent map	$c_{k+1} = \frac{c_k}{0.4}$ if $0 < c_k \leq 0.4$ , $c_{k+1} = \frac{1 - c_k}{0.6}$ if $0.4 < c_k \leq 1$

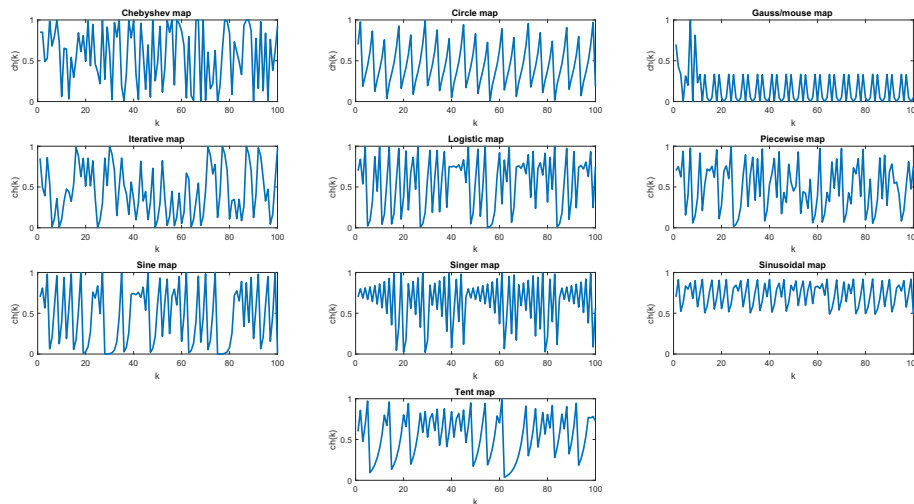


FIGURE 1. Visualization of the 10 adopted chaotic maps.

After that, the algorithm updates the position of each search agent based on equation (6) to simulate the behavior of slime moulds. The new population is then evaluated again to select the new  $G_{best}$  solution. At this stage, the proposed mutated Chaotic Local Search algorithm is invoked to update  $G_{best}$ . This process is executed until the maximum number of iterations is reached. Finally, the algorithm returns the optimal solution and its fitness function.

#### 4. EXPERIMENTAL RESULTS AND ANALYSIS

This section provides an extensive experiments to validate the proposed algorithm, and it is divided into the following main subsections:

- The first one exhibits the comparison of SMA and the ten implemented variants of mSMACLS to identify the most appropriate chaotic map.

---

**Algorithm 1.** Pseudo-code of mCLS strategy.

---

```

1: Generate the chaotic sequence
2: Identify the number of dimensions  $p$  on which the CLS will be applied
3: while  $t < t_{cls}$  do
4:   Set  $x' = CurrGbest$ 
5:   for ( $j = 1$  to  $dim$ ) do
6:     if  $\text{rand}(0, 1) < P_m$  then
7:       Select randomly a chaotic number  $c_{rand}$ 
8:       Generate new solution  $x'$  by applying CLS using equation (7)
9:       Remove temporary  $c_{rand}$  from the generated chaotic sequence
10:    end if
11:  end for
12:  Adjust the bounds of  $x'$  if necessary
13:  if ( $x'$  is better than  $CurrGbest$ ) then
14:    Update  $Gbest$ 
15:  end if
16: end while

```

---

**Algorithm 2.** Pseudo-code of mSMACLS algorithm.

---

```

1: Initialize the population  $POP$  with  $N$  search agents
2: while  $t < iter\_max$  do
3:   Evaluate each search agent
4:   Update the current  $Gbest$ 
5:   Update the best fitness BF
6:   Calculate  $w$  based on equation (4)
7:   for (each search agent) do
8:     Update  $p$ ,  $b$ , and  $c$  parameters
9:     Update the position of the current search agent based on equation (6)
10:  end for
11:  Evaluate the new population  $POP$ 
12:  Identify  $Gbest$ 
13:  Apply mCLS using Algorithm 1
14: end while
15: Return the final  $Gbest$ 

```

---

- The second outlines the comparison of mSMACLS and the first 6 selected algorithms on twenty-three classical test functions first, and then, on ten complex problems.
- The third subsection compares mSMACLS first with other four selected algorithms, and then, with three chaotic-based algorithms, on fifteen constrained optimization test problems taken from CEC2017 competition series.
- Finally, the fourth subsection, which outlines the performance of the proposed mSMACLS on three real-world problems.

All algorithms are implemented on MATLAB2016b, and they are executed on the same computer.

#### 4.1. Comparison between SMA and mSMACLS variants

Twenty-three classical benchmark functions are used to compare the basic SMA and mSMACLS algorithms. These functions include 7 unimodal (F1–F7), 6 multimodal (F8–F13), and 10 fixed-size dimension multimodal (F14–F23). Some information about these functions are listed in Table 3. The population size and the number of iterations in all algorithms are set to 30 and 100 respectively. Furthermore, and due to their stochastic behavior, the comparison is mainly done based on the averaged results obtained over 20 independent runs for each algorithm.

TABLE 3. Characteristics of the used benchmark functions.

Test function	Name	Type	Dimension	Range	Optimum
F1	Sphere	Unimodal	30	$[-100, 100]$	0
F2	Schwefel 2.22	Unimodal	30	$[-10, 10]$	0
F3	Schwefel 1.2	Unimodal	30	$[-100, 100]$	0
F4	Schwefel 2.21	Unimodal	30	$[-100, 100]$	0
F5	Rosenbrock	Unimodal	30	$[-30, 30]$	0
F6	Step	Unimodal	30	$[-100, 100]$	0
F7	Quartic	Unimodal	30	$[-1.28, 1.28]$	0
F8	Schwefel 2.26	Multimodal	30	$[-500, 500]$	-418.9829D
F9	Rastrigin	Multimodal	30	$[-5.12, 5.12]$	0
F10	Ackley	Multimodal	30	$[-32, 32]$	$8.8818e^{-16}$
F11	Griewank	Multimodal	30	$[-600, 600]$	0
F12	Penalized	Multimodal	30	$[-50, 50]$	0
F13	Penalized2	Multimodal	30	$[-50, 50]$	0
F14	Foxholes	Multimodal	2	$[-65.53, 65.53]$	0.998004
F15	Kowalik	Multimodal	4	$[-5, 5]$	0.0003075
F16	Six-hump camel back	Multimodal	2	$[-5, 5]$	-1.03163
F17	Branin	Multimodal	2	$[-5, 10] \times [0, 15]$	0.398
F18	Goldstein-Price	Multimodal	2	$[-5, 5]$	3
F19	Hartman 3	Multimodal	3	$[0, 1]$	-3.8628
F20	Hartman 6	Multimodal	6	$[0, 1]$	-3.32
F21	Langermann 5	Multimodal	4	$[0, 10]$	-10.1532
F22	Langermann 7	Multimodal	4	$[0, 10]$	-10.4029
F23	Langermann 10	Multimodal	4	$[0, 10]$	-10.5364

Table 4 displays the averaged results obtained by the basic SMA algorithm and the different mSMACLS variants. It should be noted that mSMACLS1 adopts Chebyshev map, mSMACLS2 adopts Circle map, mSMACLS3 uses Gauss/mouse map, and so on. The best results are marked in **boldface**. Hence, and as a first observation, all algorithms have the same performance in 5 test functions (F9, F10, F11, F16, and F17), and this is due by the fact that they reached the theoretical optimal values of these test functions. As a second observation, it is noticeable that the basic SMA is superior, or equivalent in the worst-case, to the other mSMACLS variants in F8 and F12 test functions. However, in the rest of test functions (*i.e.*, 16 test functions), there is always some mSMACLS variants which are better than the basic SMA algorithm. For example, but not limited to, mSMACLS4 has outperformed the basic SMA on 10 cases out of the rest 16 test functions, mSMACLS2 and mSMACLS8 outperformed it on 11 cases, and mSMACLS7, mSMACLS9 and mSMACLS5 were better on 12 cases. Furthermore, it seems that mSMACLS5 with Logistic map is the best performer among the ten chaotic maps, providing the best average values in 12 benchmark functions, followed by mSMACLS2 with Circle map with 10 best average values in 10 functions.

However, the above discussion cannot give us a clear idea on which is the most suitable chaotic map. Therefore, and to clarify the results, we present in Table 5 the final rank of each algorithm based on its average rank over the 23 test functions. From the last line of this table, it is evident that mSMACLS1 with Chebyshev map is the worst algorithm among the ten mSMACLS variants, and even it is worst than the basic SMA algorithm. This reveals that Chebyshev map cannot be a good choice for our proposed mCLS strategy. On the other hand, it is clear that all the rest 9 implemented mSMACLS algorithms are better than the basic SMA. This reveals that the proposed mCLS strategy has contributed significantly in improving the performance of SMA algorithm. Of course, each chaotic map with its own effectiveness. Indeed, the results of Table 5 have reinforced the previous conclusion which suggested that mSMACLS5 with Logistic map dominates the other mSMACLS

TABLE 4. Statistical results of algorithms on 23 benchmark functions.

SMA	mSMACLS1 chebyshev map	mSMACLS2 Circle map	mSMACLS3 Gauss/mouse map	mSMACLS4 Iterative map	mSMACLS5 Logistic map	mSMACLS6 Piecewise map	mSMACLS7 Sine map	mSMACLS8 Singer map	mSMACLS9 Sintusoidal map	mSMACLS10 Tent map
F1	8.3136E-77	2.2989E-91	1.2795E-82	3.3906E-90	<b>4.9286E-98</b>	7.5548E-84	3.6995E-93	3.9947E-86	2.1127E-78	1.0129E-70
F2	3.7541E-50	1.3147E-42	3.8004E-46	2.0455E-47	<b>1.9837E-53</b>	8.4980E-45	1.8551E-51	2.0355E-52	9.2396E-52	5.6409E-49
F3	8.8061E-65	2.2490E-64	5.0296E-77	2.1571E-45	<b>2.3193E-86</b>	2.1117E-74	1.6158E-73	2.2963E-63	5.8225E-81	1.6244E-66
F4	6.8440E-38	5.2497E-37	1.8288E-43	2.8687E-30	<b>2.0928E-45</b>	1.5187E-40	1.1017E-37	3.7073E-45	3.4645E-43	7.0718E-35
F5	25.13892	26.2666	20.1013	23.2000	<b>17.3890</b>	25.1669	18.5606	23.0886	23.0837	19.5346
F6	1.1988	1.1905	<b>0.6048</b>	0.9068	1.0265	0.9543	0.8360	1.3382	0.9701	0.8509
F7	1.1000E-03	6.5417E-04	7.2987E-04	<b>5.8747E-04</b>	6.4054E-04	5.8845E-04	7.8642E-04	7.8330E-04	7.7915E-04	6.9000E-04
F8	<b>-1.2551E+04</b>	<b>-1.2195E+04</b>	<b>-1.2545E+04</b>	<b>-1.2536E+04</b>	<b>-1.2536E+04</b>	<b>-1.2538E+04</b>	<b>-1.2528E+04</b>	<b>-1.2196E+04</b>	<b>-1.2550E+04</b>	<b>-1.2529E+04</b>
F9	0	0	0	0	0	0	0	0	0	0
F10	<b>8.8818E-16</b>	<b>8.8818E-16</b>	<b>8.8818E-16</b>	<b>8.8818E-16</b>	<b>8.8818E-16</b>	<b>8.8818E-16</b>	<b>8.8818E-16</b>	<b>8.8818E-16</b>	<b>8.8818E-16</b>	<b>8.8818E-16</b>
F11	0	0	0	0	0	0	0	0	0	0
F12	0.0870	0.1059	0.0746	0.0623	0.0647	0.0760	0.0660	0.0576	0.0786	0.0871
F13	0.9213	0.7076	0.9927	0.7367	1.2892	0.7668	0.6477	0.5130	0.6316	<b>0.4167</b>
F14	1.0974	<b>0.9980</b>	1.1964	1.2958	1.9074	1.0974	1.0974	<b>0.9980</b>	<b>0.9980</b>	1.0974
F15	8.9622E-04	6.3000E-03	1.1000E-03	8.5900E-04	8.3664E-04	8.5461E-04	8.2712E-04	7.1460E-04	7.2874E-04	8.8294E-04
F16	<b>-1.0316</b>	<b>-1.0316</b>	<b>-1.0316</b>	<b>-1.0316</b>	<b>-1.0316</b>	<b>-1.0316</b>	<b>-1.0316</b>	<b>-1.0316</b>	<b>-1.0316</b>	<b>-1.0316</b>
F17	<b>0.3979</b>	<b>0.3979</b>	<b>0.3979</b>	<b>0.3979</b>	<b>0.3979</b>	<b>0.3979</b>	<b>0.3979</b>	<b>0.3979</b>	<b>0.3979</b>	<b>0.3979</b>
F18	<b>3.0000</b>	4.3500	<b>3.0000</b>	<b>3.0000</b>	<b>3.0000</b>	4.3500	<b>3.0000</b>	<b>3.0000</b>	<b>3.0000</b>	<b>3.0000</b>
F19	-3.8623	-3.8621	-3.8622	-3.8621	<b>-3.8626</b>	-3.8620	-3.8625	-3.8623	-3.8624	-3.8621
F20	-3.2792	-3.2379	-3.2543	-3.2551	-3.2669	-3.2455	-3.2632	-3.2300	-3.2796	-3.2533
F21	-10.1476	-9.3874	-10.1503	-10.1483	-9.8982	-10.1505	-9.8983	-9.6421	-9.8966	-9.8949
F22	-10.3944	-9.8727	<b>-10.4003</b>	-10.4000	-10.4000	-10.4000	-9.8734	-10.3997	-10.3983	-10.3997
F23	-10.5315	-8.8390	-9.9939	<b>-10.5344</b>	-10.5339	-10.5339	-10.5334	-10.5343	-9.9978	-10.5331

TABLE 5. Rank of algorithms.

	SMA	mSMACLS1	mSMACLS2	mSMACLS3	mSMACLS4	mSMACLS5	mSMACLS6	mSMACLS7	mSMACLS8	mSMACLS9	mSMACLS10
F1	10	3	9	7	4	1	6	2	5	8	11
F2	5	1	6	9	8	1	10	4	2	3	7
F3	7	8	10	3	11	1	4	5	9	2	6
F4	7	9	5	3	11	1	6	8	2	4	10
F5	9	1	5	4	8	1	10	2	7	6	3
F6	9	8	10	1	4	7	5	2	11	6	3
F7	11	5	3	7	1	4	2	10	9	8	6
F8	1	9	1	3	5	5	4	7	8	2	6
F9	1	1	1	1	1	1	1	1	1	1	1
F10	1	1	1	1	1	1	1	1	1	1	1
F11	1	1	1	1	1	1	1	1	1	1	1
F12	9	11	1	6	3	4	7	5	2	8	10
F13	9	5	6	10	7	11	8	4	2	3	1
F14	2	1	3	2	4	2	2	1	1	2	2
F15	9	11	1	10	7	5	6	4	2	3	8
F16	1	1	1	1	1	1	1	1	1	1	1
F17	1	1	1	1	1	1	1	1	1	1	1
F18	1	2	1	1	1	1	2	1	1	1	1
F19	4	6	5	5	6	1	7	2	4	3	6
F20	2	9	6	10	5	3	8	4	11	1	7
F21	5	11	3	3	4	7	2	6	10	8	9
F22	6	8	4	1	2	2	2	7	3	5	3
F23	8	11	7	10	1	4	3	5	2	9	6
Avg rank	5.17	6.26	3.87	4.34	4.22	<b>2.87</b>	4.30	3.65	4.17	3.78	4.78
Final rank	10	11	4	8	6	1	7	2	5	3	9

variants, which means that Logistic map has a positive impact on the exploitation strength of the algorithm without affecting the exploration ability. Based on these findings, Logistic is selected as the optimal map for the proposed mSMACLS algorithm, and it will be investigated in the next experiments.

#### 4.1.1. Convergence behavior analysis

Although the above results have demonstrated the ability of the proposed mutated CLS strategy in improving the performance of the standard SMA algorithm, but, and until now, we haven't any idea about the ability of the proposed mSMACLS algorithm to converge towards the global optimum. Therefore, to investigate this very important perspective, four well-known qualitative metrics namely search history, trajectory in the first dimension, average fitness of entire population, and convergence curve, are employed, and they are represented in Figure 2.

From the search history metric, it can be seen that the search agents, represented by black points, tend to scatter around the red point which represents the global optimum in the search space. This behavior is approximately identical in almost all test functions, indicating the ability of the proposed algorithm to get consistently closer to the optimal configuration during the optimization process. Furthermore, and by inspecting the trajectory metric of the leader search agent in its first variable, we can observe that there are sudden changes in the beginning steps of optimization process. After that, these sharp movements are gradually decreased in the next steps. These observations indicate that the proposed algorithm tends to explore the search space first and then progressively exploit it over the course of iterations, resulting in a good transition between exploration and exploitation. Broadly speaking, such behavior reveals that the proposed mSMACLS eventually converges to a specific point and moves within promising local areas in the search space.

Another convergence indicator can be extracted from the third qualitative metric which shows the average fitness of all individuals over the iterations. As it can be seen, the descending behavior of average fitness curves proves that there is a good and fast enhancement of the initial random population. Indeed, this high performance can be easily concluded by observing the fourth metric in which the proposed mSMACLS algorithm shows a good convergence speed on most benchmarking functions.

#### 4.1.2. Computational time analysis

The effect of the incorporated mCLS strategy on the computational time is also investigated in our experiments. Figure 3 outlines the average runtime (in Seconds) of the standard SMA and the proposed mSMACLS over the 20 independent runs.

From this figure, it is evident that the proposed algorithm consumes more run time than the standard SMA algorithm, and this is not surprising due to the incorporated mCLS strategy that results in supplementary computation cost. Furthermore, this figure allows us to observe that the runtime time gap between SMA and mSMACLS decreases as the dimension of the problem decrease, which is the case for example in F16, F17, F19, etc. which is also very logical because the proposed mCLS strategy acts as mutation operator, as described previously in Section 3, thus, it visits certain dimensions rather than visiting all dimensions. Therefore, and based on all the above findings and discussions, we can say that the proposed mSMACLS algorithm sacrifices slightly the runtime at the hope of getting further improvement in the final solution.

Most mSMACLS variants significantly outperform the original SMA, confirming the benefit of integrating chaotic local search into the optimization process. Among them, mSMACLS5 with the Logistic map consistently achieves the best rankings, showing its ability to enhance both the exploration and exploitation balance. This improvement is attributed to the ergodic and sensitive nature of chaotic maps, which introduce structured randomness into the local search, allowing the algorithm to escape local optima and intensify the search around promising regions. In contrast, mSMACLS1 with the Chebyshev map performs worse than the original SMA, highlighting the importance of selecting an effective chaotic map.

The proposed mSMACLS algorithm demonstrates strong and reliable convergence, with agents consistently approaching the global optimum and a smooth transition from exploration to exploitation. It achieves faster and more stable convergence than the standard SMA, as shown by fitness and convergence curves. While it

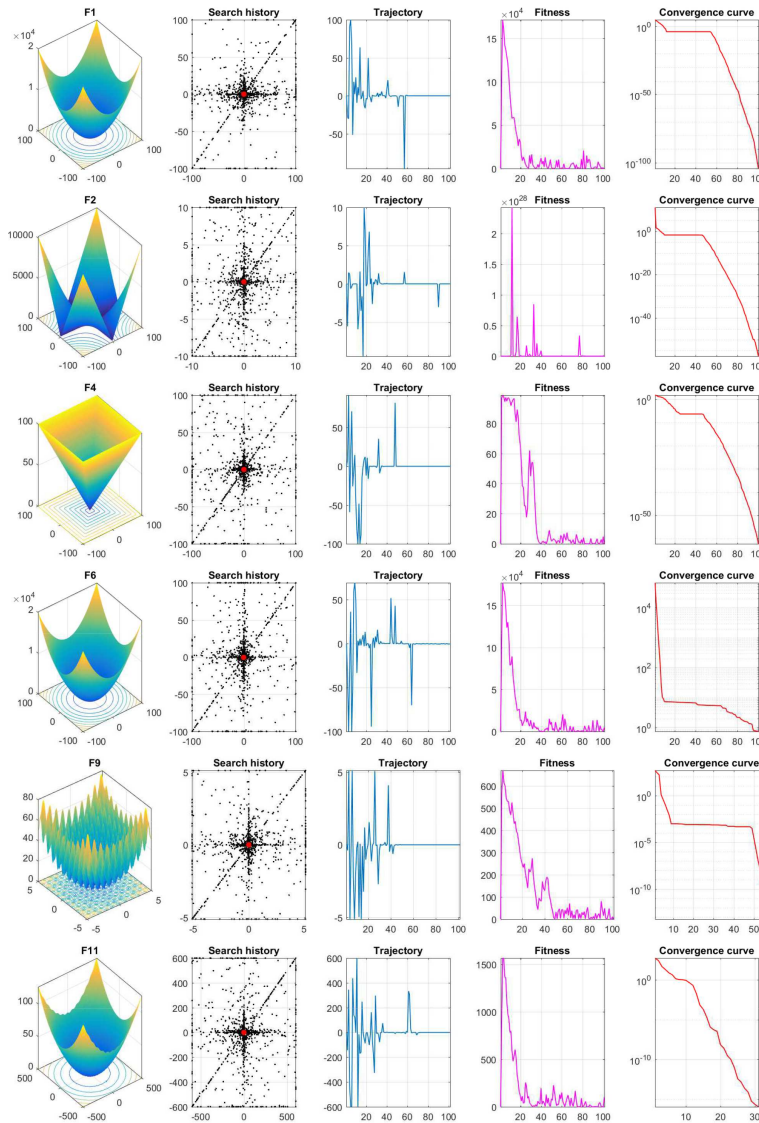


FIGURE 2. Search history, trajectory, average fitness, and convergence curve.

FIGURE 2. Search history, trajectory, average fitness, and convergence curve.

incurs a slight increase in runtime due to the mCLS strategy, this overhead is minimal in lower-dimensional problems. Overall, mSMACLS offers a worthwhile trade-off between computational cost and improved solution quality.

#### 4.2. mSMACLS vs. First 6 selected metaheuristics

The current subsection outlines the comparison of mSMACLS against six recent and well-reputed algorithms, including Artificial Hummingbird Algorithm (AHA) [65], Grey Wolf Optimizer (GWO) [6], Harris Hawks Optimizer (HHO) [66], Manta Ray Foraging Optimizer (MRFO) [8], Teaching–Learning–Based Optimization (TLBO)

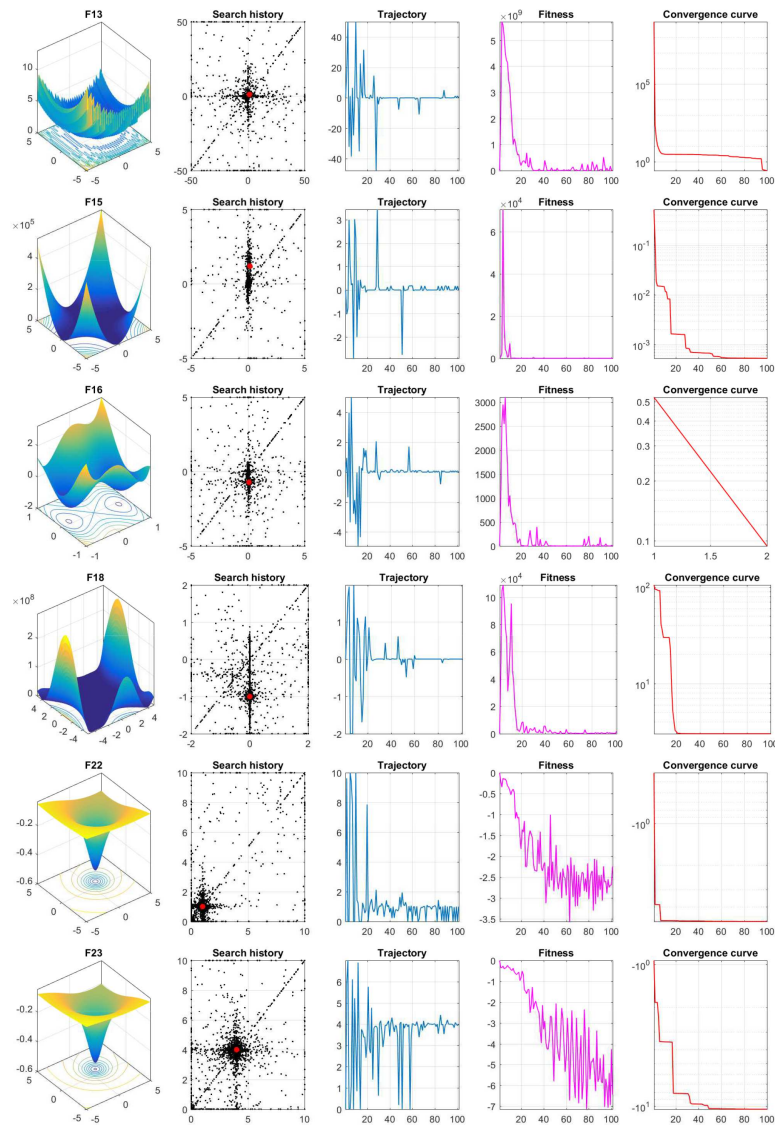


FIGURE 2. Continued.

[67], and Whale Optimization Algorithm (WOA) [7]. For fair comparison, the common parameters such as the population size, the number of iterations, and the number of independent runs, are set to be 30, 100, and 20, respectively, for all algorithms. On the other hand, it is important to emphasize that the other parameters of each algorithm have been kept as they were adopted by their original authors (under the hypothesis that these parameters are the best choice), and they are detailed in Table 6. Finally, it should be noted that the first comparison is performed on the above classical 23 benchmark functions, whereas the second comparison is performed on other 10 modern functions.

#### 4.2.1. Classical benchmark suites

The results are presented in Table 7 where *Mean* and *Std* stand for the average and the standard deviation, respectively. Based on the theoretical perspective, it may be seen that the proposed mSMACLS ranked third by

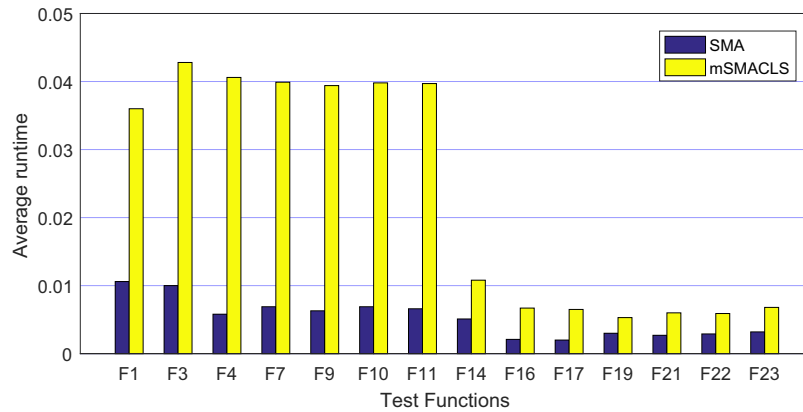


FIGURE 3. Runtime of SMA and mSMACLS algorithm.

TABLE 6. Parameter settings of the compared algorithms.

Algorithm	Parameter setting
AHA	Migration foraging coefficient: $M = 2n$ .
GWO	The coefficients: $a = 2 - \frac{2 \times t}{T}$ , $A = 2 \cdot a \cdot \text{rand}(0, 1) - a$ , $C = 2 \cdot \text{rand}(0, 1)$ .
HHO	Factor to show the decreasing energy of rabbit: $E = 2 \cdot E_0 \cdot (1 - \frac{t}{T})$ .
MRFO	The somersault factor: $S = 2$ .
TLBO	Teaching Factor: $\text{TF} = \text{rand}i([1 \quad 2])$
WOA	The coefficients: $a = 2 - t \times (\frac{2}{T})$ , The shape of the logarithmic spiral: $b = 1$ .
mSMACLS	The factors: $b = 1 - \frac{t}{T}$ , and $z = 0.03$

achieving the theoretical optimum value in 6 cases (F9–11, F16–18), after the AHA algorithm which has attained the optimum value in 7 cases (F6, F9, F11, F16–19) and the MRFO algorithm which has achieved the theoretical optimum in 9 cases (F6, F9–F11, F14, F16–F19). However, and based on the average point of view, the proposed mSMACLS has obtained the best *Mean* value in 15 test functions (F1–4, F7–11, F16–18, F21–23), while MRFO ranked second with best *Mean* value in 10 cases (F6, F9–11, F14–F19). Moreover, it is clear that mSMACLS straightly outperforms GWO, HHO, TLBO, and WOA algorithms in terms of theoretical and average results.

Based on the standard deviation, mSMACLS has obtained the best *Std* metric in 12 cases (F1–4, F7–11, F21–23), followed by MRFO algorithm with best *Std* in 9 cases, meaning that mSMACLS provides a good stability and robustness over the 20 independent executions compared to its competitors. This conclusion can be confirmed by the Box-and-whisker diagrams of some test functions presented in Figure 4. For each algorithm, the top and bottom edges of each box represent the minimum and maximum values. The top and bottom blue lines borders define the first and the third quartiles, respectively. The red line represents the median, whereas the red “+” indicates the outliers points which are outside the box. Knowing that IRQ metric stands for the inter-quartile range, or simply, the distance between the first and the third quartiles. The shorter the IRQ is, the better the stability of the obtained solutions is. Hence, it may be observed that mSMACLS has the shortest IRQ metric in almost all considered test functions. Also, it is clear that the mSMACLS generates less outliers points, which means that the proposed algorithm has a stronger stability and robustness compared to the selected state-of-the-art algorithms during the repeated executions.

TABLE 7. Experimental results of different algorithms on classical test suites.

Function	Metric	AHA	GWO	HHO	MRFO	TLBO	WOA	mSMACLS
F1	Mean	7.0911E-23	0.0178	2.0457E-22	1.1539E-76	1.6256E-14	1.2596E-11	<b>1.2885E-91</b>
	Std	2.2421E-22	0.0126	4.9369E-22	2.6336E-76	1.1027E-14	2.5224E-11	<b>4.6510E-91</b>
F2	Mean	5.3503E-14	0.0259	1.4109E-12	2.6252E-40	7.6180E-08	3.9572E-09	<b>2.0223E-52</b>
	Std	9.3046E-14	0.0087	3.4914E-12	4.9742E-40	3.7662E-08	6.6074E-09	<b>6.3782E-52</b>
F3	Mean	2.9840E-22	333.4569	5.1270E-16	3.6214E-72	1.6414	1.0592E+05	<b>8.2593E-75</b>
	Std	6.0152E-22	214.4506	1.9933E-15	1.4018E-71	1.2216	2.5010E+04	<b>3.6621E-74</b>
F4	Mean	1.3397E-11	1.4553	5.8165E-13	3.8358E-39	2.6139E-06	69.6378	<b>3.6785E-49</b>
	Std	3.2333E-11	0.4222	1.9989E-12	8.5284E-39	1.3200E-06	25.2138	<b>1.4473E-48</b>
F5	Mean	28.6962	30.7466	<b>0.3563</b>	26.6881	27.7190	28.7698	19.2578
	Std	0.1692	3.1548	0.4233	0.4567	0.3409	<b>0.0605</b>	13.1896
F6	Mean	<b>0</b>	2.8382	0.0066	<b>0</b>	0.3776	1.9700	0.8301
	Std	<b>0</b>	0.6059	0.0097	<b>0</b>	0.1913	0.4665	0.5431
F7	Mean	0.0015	0.0191	9.1070E-04	8.7536E-04	0.0067	0.0309	<b>5.9468E-04</b>
	Std	8.4103E-04	0.0082	9.1040E-04	7.0091E-04	0.0027	0.0324	<b>5.4260E-04</b>
F8	Mean	-7.3451E+03	-5.8989E+03	-8.6028E+03	-6.9697E+03	-4.8594E+03	-9.3071E+03	<b>-1.2562E+04</b>
	Std	533.4850	976.8578	2.3581E+03	867.2794	811.1306	6.1110E+03	<b>5.7791</b>
F9	Mean	<b>0</b>	30.4698	<b>0</b>	<b>0</b>	63.6426	14.0962	<b>0</b>
	Std	<b>0</b>	13.9383	<b>0</b>	<b>0</b>	29.1798	50.9306	<b>0</b>
F10	Mean	8.8427E-13	0.0241	2.9612E-13	<b>8.8818E-16</b>	3.1877E-08	5.4630E-07	<b>8.8818E-16</b>
	Std	2.9401E-12	0.0099	6.7808E-13	<b>0</b>	1.5899E-08	7.9375E-07	<b>0</b>
F11	Mean	<b>0</b>	0.0610	<b>0</b>	<b>0</b>	7.2970E-12	3.4525E-11	<b>0</b>
	Std	<b>0</b>	0.0606	<b>0</b>	<b>0</b>	3.1862E-11	6.2019E-11	<b>0</b>
F12	Mean	0.1556	0.4159	<b>6.7725E-04</b>	0.0041	0.0159	0.1311	0.0560
	Std	0.0391	0.2089	<b>9.1074E-04</b>	0.0038	0.0202	0.0695	0.0446
F13	Mean	2.6080	2.0783	<b>0.0062</b>	2.5897	0.6032	1.2794	0.7854
	Std	0.4880	0.5687	<b>0.0064</b>	0.8648	0.2900	0.4247	0.9010
F14	Mean	1.9704	4.3757	1.9891	<b>0.9980</b>	<b>0.9980</b>	4.0925	1.2956
	Std	2.2175	4.2509	1.2417	<b>8.9251E-09</b>	2.4318E-08	3.7276	0.7269
F15	Mean	4.2020E-04	0.0051	6.9386E-04	<b>4.0335E-04</b>	6.0394E-04	7.5228E-04	7.5412E-04
	Std	1.3962E-04	0.0081	0.0013	<b>1.0716E-04</b>	1.6509E-04	4.7198E-04	2.9986E-04
F16	Mean	<b>-1.0316</b>	<b>-1.0316</b>	<b>-1.0316</b>	<b>-1.0316</b>	<b>-1.0316</b>	<b>-1.0316</b>	<b>-1.0316</b>
	Std	1.0366E-07	5.1778E-07	4.3768E-06	<b>1.0188E-16</b>	1.3478E-16	1.5764E-06	5.1356E-10
F17	Mean	<b>0.3979</b>	0.3980	0.3982	<b>0.3979</b>	<b>0.3979</b>	0.3987	<b>0.3979</b>
	Std	1.9494E-05	3.1583E-04	7.7387E-04	<b>6.0657E-13</b>	7.6535E-13	0.0016	1.5994E-08
F18	Mean	<b>3.0000</b>	3.0005	3.0003	<b>3.0000</b>	<b>3.0000</b>	8.4279	<b>3.0000</b>
	Std	1.0882E-07	8.8324E-04	4.4156E-04	4.6521E-15	<b>1.4552E-15</b>	11.1366	8.6339E-10
F19	Mean	<b>-3.8628</b>	-3.8611	-3.8553	<b>-3.8628</b>	<b>-3.8628</b>	-3.8237	-3.8626
	Std	2.4840E-07	0.0023	0.0116	<b>1.8536E-15</b>	2.0402E-15	0.0423	5.6702E-04
F20	Mean	<b>-3.3002</b>	-3.2479	-3.2232	-3.2744	-3.2957	-3.1676	-3.2768
	Std	0.0434	0.0775	0.0909	0.0598	<b>0.0456</b>	0.1349	0.0632
F21	Mean	-7.5539	-6.8623	-5.8156	-6.8363	-9.4881	-6.8256	<b>-10.1472</b>
	Std	2.0587	3.4320	1.8639	2.4903	1.7675	2.8996	<b>0.0170</b>
F22	Mean	-8.2812	-9.7713	-6.8913	-8.8084	-9.0017	-6.4882	<b>-10.4001</b>
	Std	2.1741	1.8492	2.5256	2.4990	2.5786	3.2141	<b>0.0037</b>
F23	Mean	-9.0561	-8.9005	-4.8551	-8.8433	-10.1956	-6.1605	<b>-10.5335</b>
	Std	1.9028	3.2693	0.8330	2.5146	1.4972	2.6316	<b>0.0031</b>

To further identify the significant difference between the algorithms, the Wilcoxon rank-sum with a significant level equal to 0.05 is adopted as a non-parametric statistical test. This test provides as a result the so-called “ $p$ -value”, and it leads to three possible situations:

- Case 1 {=}:  $p$ -value (mSMACLS, algorithm  $A$ )  $> 0.05$ , implies that there is no difference between the compared pairwise algorithms.
- Case 2 {+}:  $p$ -value (mSMACLS, algorithm  $A$ )  $< 0.05$ , implies that mSMACLS is significantly better than algorithm  $A$  if its average is better.
- Case 3 {-}:  $p$ -value (mSMACLS, algorithm  $A$ )  $< 0.05$ , implies that mSMACLS is significantly worse than algorithm  $A$  if its average is worst.

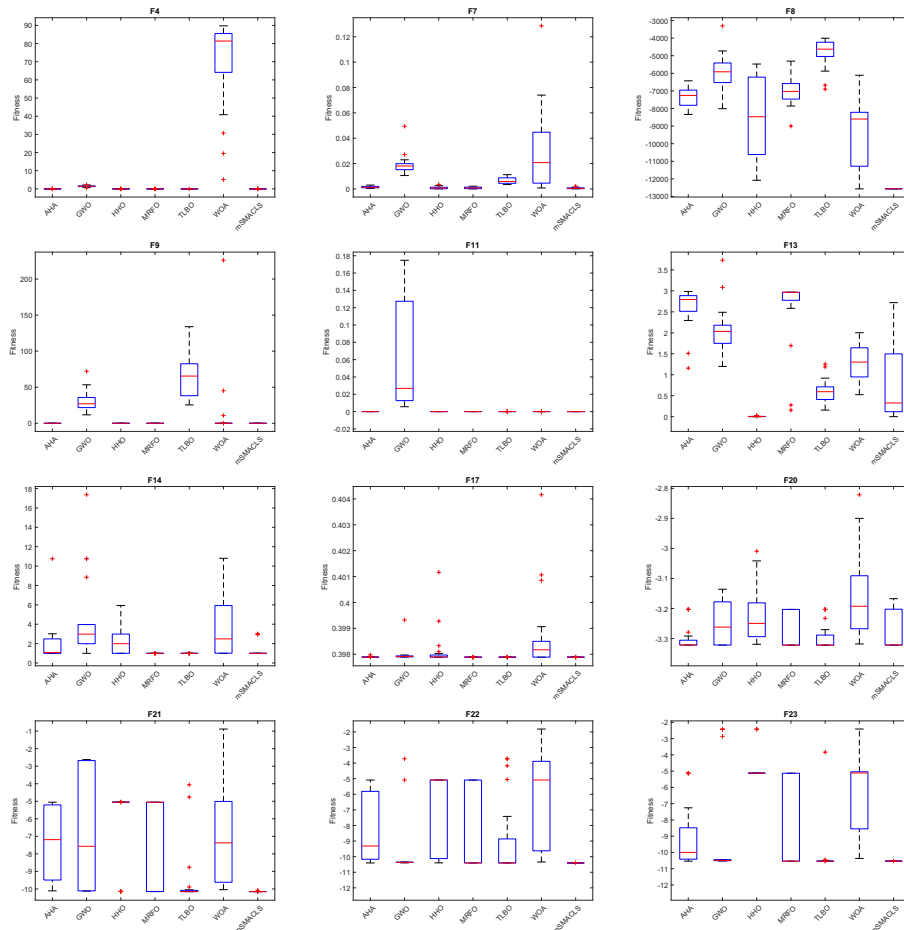


FIGURE 4. Box-and-whisker diagrams on classical test suites.

The results of Wilcoxon test are reported in Table 8, and by seeing the last row which summarizes the obtained  $p$ -values, it is evident that the proposed algorithm performs significantly better than the selected methods. Indeed, and for all the 138 cases (6 algorithms  $\times$  23 functions), the proposed mSMACLS was significantly better (+) in 88 cases, similar (=) in 34 cases, and significantly worst (–) in 16 cases. In addition, the results have confirmed that the MRFO is the best competitor to the proposed algorithm.

Furthermore, Figure 5 presents the convergence curves of algorithms along with the number of iterations. It is worth mentioning that these curves are plotted in semi-logarithmic form. Accordingly, it can be stated that the proposed mSMACLS converges faster with better accuracy compared to the selected six algorithms, especially compared to AHA, GWO, TLBO, and WOA. This can be clearly observed in almost all test functions. Additionally, in certain cases such as F10–12 test functions, mSMACLS’s convergence speed was slightly lower than those of HHO and MRFO algorithms, however, it reached eventually a good convergence rate in the final stage of optimization process.

TABLE 8.  $p$ -values of Wilcoxon test on classical test suites.

Function	Wilcoxon	AHA	GWO	HHO	MRFO	TLBO	WOA
F1	$p$ -value	6.7956E-08	6.7956E-08	6.7956E-08	6.7956E-08	6.7956E-08	6.7956E-08
	Sign	+	+	+	+	+	+
F2	$p$ -value	6.7956E-08	6.7956E-08	6.7956E-08	6.7956E-08	6.7956E-08	6.7956E-08
	Sign	+	+	+	+	+	+
F3	$p$ -value	6.7956E-08	6.7956E-08	6.7956E-08	1.1044E-05	6.7956E-08	6.7956E-08
	Sign	+	+	+	+	+	+
F4	$p$ -value	6.7956E-08	6.7956E-08	6.7956E-08	6.7956E-08	6.7956E-08	6.7956E-08
	Sign	+	+	+	+	+	+
F5	$p$ -value	9.7479E-06	1.0472E-06	5.0907E-04	0.1075	0.1198	6.7956E-08
	Sign	+	+	-	=	=	+
F6	$p$ -value	8.0065E-09	7.8980E-08	7.8980E-08	8.0065E-09	1.1452E-02	1.2008E-06
	Sign	-	+	-	-	-	+
F7	$p$ -value	3.0479E-04	6.7956E-08	0.4093	0.3234	6.7956E-08	1.9177E-07
	Sign	+	+	=	=	+	+
F8	$p$ -value	6.7956E-08	6.7956E-08	6.7956E-08	6.7956E-08	6.7956E-08	7.5773E-06
	Sign	+	+	+	+	+	+
F9	$p$ -value	1.0000	8.0065E-09	1.0000	1.0000	8.0065E-09	2.9867E-08
	Sign	=	+	=	=	+	+
F10	$p$ -value	2.9764E-08	8.0065E-09	1.0398E-07	1.0000	8.0065E-09	8.0065E-09
	Sign	+	+	+	=	+	+
F11	$p$ -value	1.0000	8.0065E-09	1.0000	1.0000	8.0065E-09	8.0065E-09
	Sign	=	+	=	=	+	+
F12	$p$ -value	1.3760E-06	7.8980E-08	9.1266E-07	2.9248E-05	4.3201E-03	6.8622E-04
	Sign	+	+	-	-	-	+
F13	$p$ -value	1.2008E-06	9.2779E-05	5.1657E-06	3.4994E-06	0.3648	1.6668E-02
	Sign	+	+	-	+	=	+
F14	$p$ -value	2.9248E-05	2.0615E-06	2.0407E-05	0.5883	0.7048	4.5400E-06
	Sign	+	+	+	=	=	+
F15	$p$ -value	1.2940E-04	0.2976	3.7499E-04	5.8995E-05	0.1555	0.4569
	Sign	-	=	-	-	=	=
F16	$p$ -value	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	Sign	=	=	=	=	=	=
F17	$p$ -value	1.0000	6.7956E-08	1.9956E-04	1.0000	1.0000	2.2177E-07
	Sign	=	+	+	=	=	+
F18	$p$ -value	1.0000	6.7956E-08	2.9248E-05	1.0000	1.0000	6.7956E-08
	Sign	=	+	+	=	=	+
F19	$p$ -value	7.8980E-08	1.2505E-05	1.7936E-04	1.9543E-08	4.1424E-08	7.8980E-08
	Sign	-	+	+	-	-	+
F20	$p$ -value	0.0638	0.0531	4.3201E-03	9.7864E-03	3.3287E-03	3.7499E-04
	Sign	=	=	+	+	-	+
F21	$p$ -value	9.1727E-08	2.5629E-07	1.6570E-07	6.0213E-03	0.9460	6.7956E-08
	Sign	+	+	+	+	=	+
F22	$p$ -value	3.0691E-06	1.9177E-07	1.2346E-07	4.6791E-02	0.08103	6.7956E-08
	Sign	+	+	+	+	=	+
F23	$p$ -value	3.9873E-06	9.1727E-08	6.7956E-08	0.1264	3.0566E-03	6.7956E-08
	Sign	+	+	+	=	+	+
Final score of +/-/-		14/6/3	20/3/0	14/4/5	9/10/4	10/9/4	21/2/0

#### 4.2.2. CEC2019 benchmark suites

This part presents an extra comparison of the proposed mSMACLS and 6 selected algorithms on more complex benchmark functions. These 10 functions, listed with their main characteristics in Table 9, were introduced in the Special Competition on 100-Digit Challenge and Numerical Optimization Scenarios (CEC2019) [68].

Table 10 shows the obtained results and the rank of all algorithms on the CEC2019 test suites. From this table, it can be seen that the proposed mSMACLS has obtained the best statistical *Mean* values in five cases (CEC04, CEC06-08, CEC10), while TLBO and MRFO have obtained the best *Mean* values in four and three

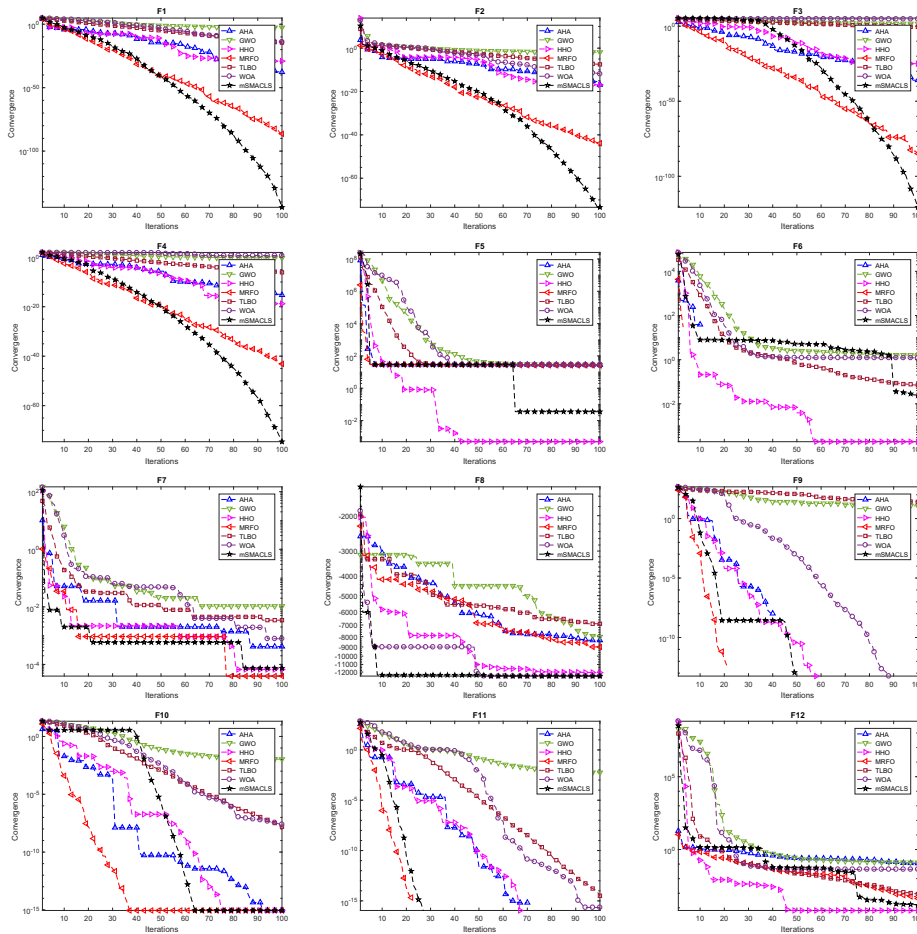


FIGURE 5. Convergence curves on classical test suites.

cases, respectively. However, the overall rank of all algorithms suggest that AHA was the second best performer after the winner mSMACLS. These findings prove the high performance of the proposed algorithm compared to the selected methods. However, and despite these encouraging results, the *Std* values listed in Table 10 suggested that the stability of mSMACLS is not the same compared to its stability on the first benchmark functions set, but it is still acceptable.

Indeed, Figure 6 on CEC2019 test suites shows that the proposed algorithm is still very competitive to the other methods in terms of stability.

We used Wilcoxon test again to verify the performance of algorithms on these CEC2019 test functions. The obtained *p*-values are listed in Table 11, and based on last row of this table, it can be seen that the proposed algorithm is significantly superior to the other algorithms.

Figure 7 presents the convergence curves of the algorithms on CEC2019 test functions. It can be seen that the convergence behavior is approximately similar to the convergence behavior of the algorithms on classical benchmark functions. That is to say, the proposed mSMACLS converges faster with better accuracy in certain cases (as in CEC04 and CEC06-07 functions), and it was slower on other functions (as CEC08 and CEC09), but with a good convergence rate in the final iterations. The exception is on CEC10 test function where the HHO was the best performer in terms of accuracy and convergence speed.

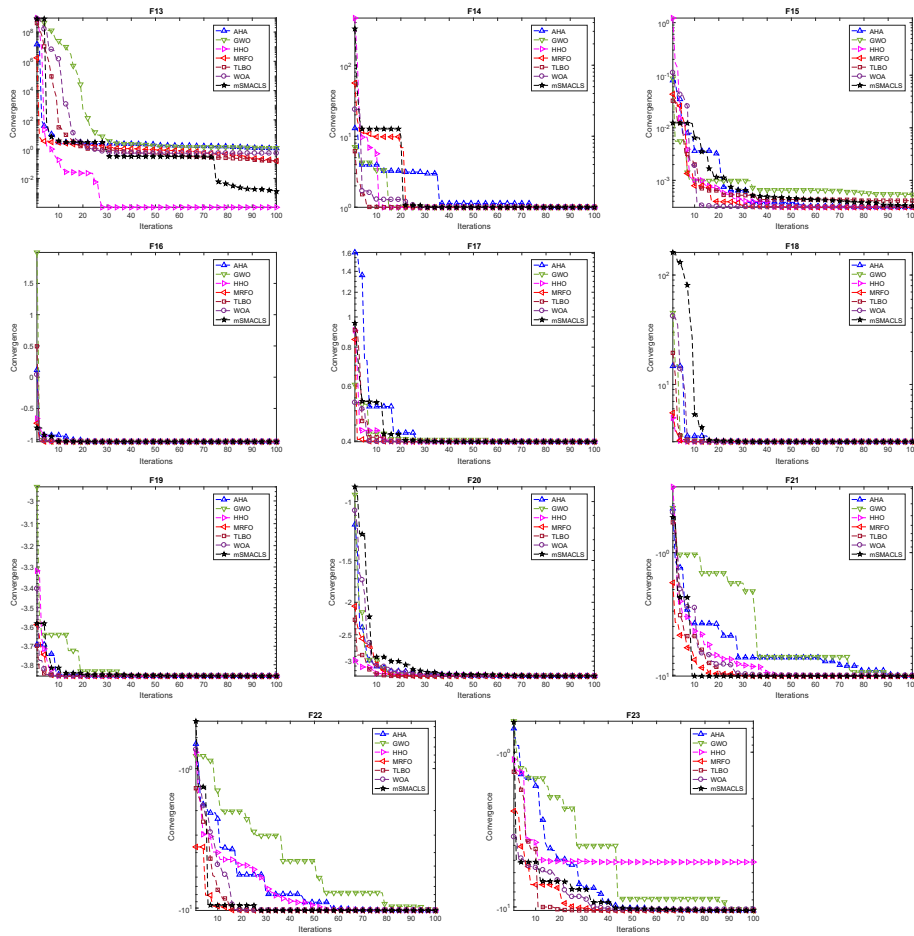


FIGURE 5. Continued.

TABLE 9. Characteristics of CEC2019 benchmark suites.

Name	Function	Dimension	Range	Optimum
CEC01	Storn's Chebyshev Polynomial Fitting Problem	9	$[-8192, 8192]$	1
CEC02	Inverse Hilbert Matrix Problem	16	$[-16384, 16384]$	1
CEC03	Lennard-Jones Minimum Energy Cluser	18	$[-4, 4]$	1
CEC04	Rastrigin's Function	10	$[-100, 100]$	1
CEC05	Griewangk's Function	10	$[-100, 100]$	1
CEC06	Weierstrass Function	10	$[-100, 100]$	1
CEC07	Modified Schwefel's Function	10	$[-100, 100]$	1
CEC08	Expanded Schaffer's F6 Function	10	$[-100, 100]$	1
CEC09	Happy Cat Function	10	$[-100, 100]$	1
CEC10	Ackley Function	10	$[-100, 100]$	1

TABLE 10. Comparison and ranks of algorithms on CEC2019 benchmarks.

Function	Metric	AHA	GWO	HHO	MRFO	TLBO	WOA	SMA
CEC01	Mean	4.7681E+04	2.3941E+09	7.3015E+04	<b>4.2198E+04</b>	1.6297E+09	2.0471E+11	6.0435E+04
	Std	3.6390E+03	1.8947E+09	1.8863E+04	<b>2.6401E+03</b>	1.0258E+09	2.5391E+11	2.8301E+04
	Rank	2	5	4	<b>1</b>	6	7	3
CEC02	Mean	17.3642	17.3479	17.3833	<b>17.3429</b>	<b>17.3429</b>	17.5725	17.3462
	Std	0.0183	0.0016	0.0169	8.3385E-07	<b>1.3728E-08</b>	0.2075	0.0016
	Rank	4	3	5	<b>1</b>	<b>1</b>	6	2
CEC03	Mean	<b>12.7024</b>	12.7025	<b>12.7024</b>	<b>12.7024</b>	<b>12.7024</b>	<b>12.7024</b>	12.7027
	Std	1.5900E-06	4.5273E-04	1.7511E-05	<b>1.2961E-12</b>	3.6610E-07	3.1181E-05	3.3615E-04
	Rank	<b>1</b>	2	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	3
CEC04	Mean	280.7217	252.1711	2.9623E+03	60.8599	78.6651	1.9000E+03	<b>51.2453</b>
	Std	264.4406	517.1113	1.9513E+03	<b>21.0407</b>	32.4937	730.8877	33.3710
	Rank	5	4	7	<b>2</b>	3	6	<b>1</b>
CEC05	Mean	1.5926	1.6414	2.6630	1.2717	<b>1.2610</b>	2.3261	1.3890
	Std	0.1824	0.2198	0.7500	0.2317	0.1383	0.3857	<b>0.1372</b>
	Rank	4	5	6	2	<b>1</b>	7	<b>3</b>
CEC06	Mean	9.0614	12.2195	10.4983	11.3723	11.3240	10.3046	<b>8.8254</b>
	Std	0.8908	0.9118	1.2003	1.0077	<b>0.7903</b>	1.3868	1.2051
	Rank	2	7	4	6	5	3	<b>1</b>
CEC07	Mean	319.5472	763.9039	414.2357	694.2357	825.8908	961.8305	<b>290.8125</b>
	Std	<b>108.2871</b>	347.0000	122.0902	248.8753	191.4255	234.8166	250.3932
	Rank	2	5	3	4	6	7	<b>1</b>
CEC08	Mean	5.4984	5.7812	5.9565	5.9329	5.9348	6.3034	<b>5.3364</b>
	Std	0.5327	0.9746	0.6036	0.6227	<b>0.4634</b>	0.6409	0.5934
	Rank	2	3	6	4	5	7	<b>1</b>
CEC09	Mean	4.1733	5.0250	122.7516	2.9960	<b>2.8584</b>	175.3242	3.1628
	Std	0.8446	1.2639	121.2131	0.3032	<b>0.2490</b>	109.2464	0.6615
	Rank	4	5	6	2	1	7	3
CEC10	Mean	20.2188	20.5960	20.3299	20.5654	20.5358	20.4706	<b>20.0888</b>
	Std	0.0898	0.1150	0.3357	<b>0.0760</b>	0.1073	0.1557	0.0977
	Rank	2	7	3	6	5	4	<b>1</b>
Avg rank		2.8	4.6	4.5	2.9	3.4	5.5	<b>1.9</b>
Overall		2	6	5	3	4	7	<b>1</b>

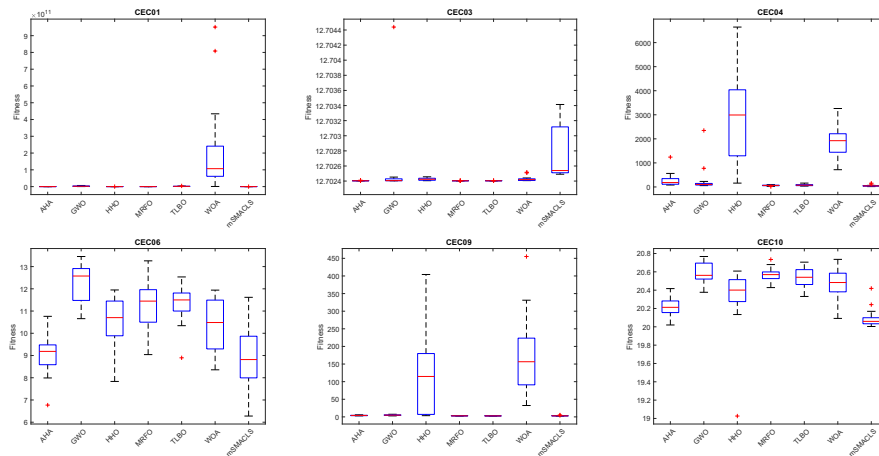


FIGURE 6. Box-and-whisker diagrams on CEC2019 test suites.

TABLE 11.  $p$ -values of Wilcoxon test on CEC2019 test suites.

Function	Wilcoxon	AHA	GWO	HHO	MRFO	TLBO	WOA
CEC01	$p$ -value	4.9863E-02	6.7956E-08	2.5606E-03	2.3556E-06	6.7956E-08	6.7956E-08
	Sign	-	+	+	-	+	+
CEC02	$p$ -value	9.8815E-05	5.2120E-03	1.2009E-05	1.2009E-05	1.2009E-05	1.2009E-05
	Sign	+	+	+	-	-	+
CEC03	$p$ -value	6.7956E-08	1.2008E-06	6.7956E-08	6.7956E-08	6.7956E-08	4.5389E-07
	Sign	-	-	-	-	-	-
CEC04	$p$ -value	5.2268E-07	8.5974E-06	6.7956E-08	2.5639E-03	2.1392E-03	6.7956E-08
	Sign	+	+	+	+	+	+
CEC05	$p$ -value	6.2199E-04	5.6290E-04	6.7956E-08	9.0453E-03	7.7118E-03	6.7956E-08
	Sign	+	+	+	-	-	+
CEC06	$p$ -value	0.5608	1.6570E-07	3.7499E-04	2.3556E-06	2.3556E-06	2.3412E-03
	Sign	=	+	+	+	+	+
CEC07	$p$ -value	0.5075	5.8959E-05	2.9440E-02	4.6804E-05	1.5756E-06	2.9597E-07
	Sign	=	+	+	+	+	+
CEC08	$p$ -value	0.3909	0.0903	1.0063E-02	2.9427E-02	1.6499E-02	1.2225E-03
	Sign	=	=	+	+	+	+
CEC09	$p$ -value	7.4064E-05	4.5400E-06	1.9177E-07	0.8392	0.0810	6.7956E-08
	Sign	+	+	+	=	=	+
CEC10	$p$ -value	7.4064E-05	7.8980E-08	7.5773E-06	6.7956E-08	1.0645E-07	3.9388E-07
	Sign	+	+	+	+	+	+
Final score of +/=/-		5/3/2	8/1/1	9/0/1	5/1/4	6/1/3	9/0/1

In summary, the results of the first experiments have demonstrated that the introduced mutated Chaotic Local Search strategy has significantly improved the performance of the standard Slime Mould Algorithm. In fact, 9 out of the 10 tested CLS maps showed a considerable performance increase over the classical SMA. More specifically, the Logistic map based CLS presented the best results among the 10 experimented CLS variants. These results suggested that exploiting the Logistic map in the mSMALCS gives a better exploration/exploitation balance, leading to concurrent results for global optimization.

Furthermore, the SMA-mutated Chaotic Local Search demonstrates a good convergence. It was consistent even if it was not the fastest one, it reached satisfying results. CLS's stop mechanism proved its efficiency in avoiding premature convergence and stagnation in a local optimum.

Also, the proposed algorithm has proven to be very competitive and even superior to other well-regarded algorithms. Indeed, mSMACLS exhibited high stability and boosted efficiency on classical benchmarks, as well as modern complex ones. Regarding runtime, mSMACLS consumes very little additional time compared to the classical SMA, but it is still very fast.

### 4.3. mSMACLS on constrained CEC2017 test functions

This subsection presents an additional analysis of the performance of mSMACLS on 15 benchmark functions taken from CEC2017 competition. The CEC2017 benchmark suite for constrained single-objective optimization consists of a diverse set of test functions designed to evaluate the effectiveness of algorithms in handling constraints. These functions typically operate in 10, 30, 50, or 100 dimensions, with 30 being the most commonly used. Each function includes a varying number of equality and/or inequality constraints, ranging from none to over 30, and the feasible regions can be small or disconnected, making the search space particularly challenging. The benchmark covers a wide spectrum of function types, including separable, non-separable, multimodal, and rotated functions, and some are derived from real-world engineering problems. This diversity makes the CEC2017 suite a robust and demanding benchmark for evaluating the convergence, robustness, and constraint-

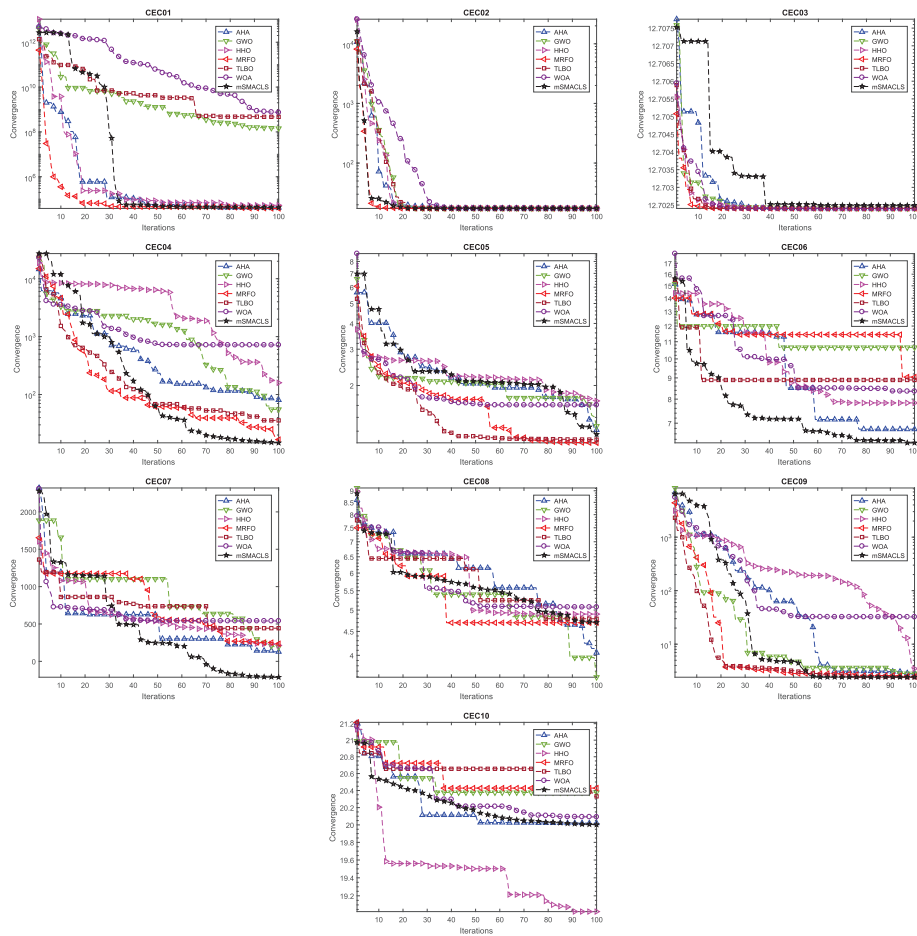


FIGURE 7. Convergence curves on CEC2019 test suites.

handling capabilities of optimization algorithms [69]. However, it is important to state that 30 is the number of decision variables (*i.e.*, dimension) of each used test function in this study.

#### 4.3.1. mSMACLS vs. Second four selected algorithms

This section outlines the comparison of mSMACLS with four well-reputed algorithms namely JADE [70], SHADE [71], CLPSO [72], and RIME [19]. All algorithms were executed 20 independent times, along with 2500 iterations, and a population size of 30 search agent.

Table 12 reports the mean and standard deviation (Std) of results obtained for 15 benchmark functions (F1, F3–F6, F11–F15, F21–F25) where lower values indicate better performance. A comparative evaluation across the benchmark suite reveals that SHADE achieves the best results in 8 functions, while JADE outperforms others in 6 functions. mSMACLS ranks first in only 1 function (F25). This indicates that SHADE and JADE remain dominant in terms of solution quality on this benchmark set.

However, a closer inspection shows that on several functions F5, F6, F11, F21, F23, and F24, the differences between all algorithms are marginal, with mSMACLS producing results nearly indistinguishable from the best-performing methods. Even in F25, where mSMACLS achieves the best average performance, the gap remains relatively small.

TABLE 12. Statistical results of algorithms on CEC2017.

Function	Metric	JADE	SHADE	CLPSO	RIME	mSMACLS
F1	Mean	1.6825E+02	<b>1.0000E+02</b>	1.7234E+04	8.3355E+04	4.2512E+03
	Std	3.0522E+02	1.8064E-13	8.5523E+03	3.5800E+04	4.6828E+03
F3	Mean	8.6781E+03	<b>3.0000E+02</b>	7.1237E+04	9.7322E+02	3.2304E+02
	Std	1.3781E+04	5.6191E-06	1.3126E+04	4.6025E+02	2.2978E+01
F4	Mean	<b>4.1550E+02</b>	4.5394E+02	5.1670E+02	5.0031E+02	4.9615E+02
	Std	2.5227E+01	2.3730E+03	1.2926E+01	2.3586E+01	1.1225E+01
F5	Mean	5.4623E+02	<b>5.3251E+02</b>	5.8295E+02	5.8799E+02	6.1482E+02
	Std	0.8703E+01	0.9608E+01	1.2145E+01	2.6218E+01	3.5183E+01
F6	Mean	<b>6.0000E+02</b>	6.0002E+02	6.0001E+02	6.0258E+02	6.1307E+02
	Std	1.5865E-13	0.0030E-01	0.0004E-01	0.2126E+01	0.7931E+01
F11	Mean	<b>1.1811E+03</b>	1.2241E+03	1.2837E+03	1.2661E+03	1.2590E+03
	Std	3.4487E+01	5.3310E+01	2.6186E+01	7.2221E+01	3.5135E+01
F12	Mean	<b>9.6140E+03</b>	1.7600E+04	2.4232E+06	4.1489E+06	2.0390E+06
	Std	9.1860E+03	1.5827E+04	1.1462E+06	2.8112E+06	1.4563E+06
F13	Mean	2.6414E+03	<b>2.3570E+03</b>	4.4391E+04	2.8424E+04	3.0341E+04
	Std	3.6754E+03	8.5057E+01	3.2460E+04	3.7283E+04	2.4012E+04
F14	Mean	1.3386E+04	<b>1.5795E+03</b>	8.4775E+04	3.4907E+04	6.2298E+04
	Std	1.9779E+04	7.0016E+01	8.0470E+04	2.9145E+04	3.8637E+04
F15	Mean	2.5314E+03	<b>1.7999E+03</b>	3.2101E+03	1.2670E+04	2.7536E+04
	Std	2.8727E+03	1.7870E+02	1.3224E+03	1.2325E+03	1.5065E+04
F21	Mean	2.3435E+03	<b>2.3342E+03</b>	2.3864E+03	2.3977E+03	2.3960E+03
	Std	0.8784E+01	0.8938E+01	1.3641E+01	2.5845E+01	2.7265E+01
F22	Mean	<b>2.3002E+03</b>	3.0979E+03	2.9279E+03	4.4880E+03	5.6354E+03
	Std	0.0756E+01	1.2425E+03	1.0994E+03	1.7630E+03	1.6103E+03
F23	Mean	<b>2.6891E+03</b>	2.6914E+03	2.7385E+03	2.7583E+03	2.7572E+03
	Std	1.0196E+01	1.3067E+01	1.0101E+01	3.0231E+01	2.3578E+01
F24	Mean	2.8618E+03	<b>2.8604E+03</b>	2.9632E+03	2.9270E+03	2.9431E+03
	Std	1.0999E+01	0.9075E+01	1.7252E+01	2.8719E+01	3.2370E+01
F25	Mean	2.8908E+03	2.8877E+03	2.8950E+03	2.8965E+03	<b>2.8875E+03</b>
	Std	1.2408E+01	0.00641E+01	0.5099E+01	1.8266E+01	0.1900E+01

On F3 and F4, mSMACLS performs competitively, producing results very close to the best ones achieved by SHADE and JADE, respectively. Similarly, for F1, F12, F13, F14, and F15, although mSMACLS does not lead, its performance remains within an acceptable range when compared to the top algorithms.

In terms of stability (as measured by standard deviation), SHADE and JADE demonstrate the most consistent performance across runs. mSMACLS shows a level of robustness comparable to CLPSO, and significantly more stable than RIME, which shows high variability across most functions.

Figure 8 shows the convergence rate of the 5 compared algorithms, CLPSO exhibits the slowest convergence, consistently falling behind all other algorithms. This indicates a lower optimization speed and suggests that it may require significantly more iterations to reach competitive solution quality.

JADE and SHADE demonstrate very similar and superior convergence curves, consistently outperforming others in terms of speed and stability. Their fast descent toward optimal regions highlights their strong exploration-exploitation balance and robust search mechanisms.

The mSMACLS algorithm presents an intermediate convergence profile, with curves that always lie between those of JADE and CLPSO. This suggests that while mSMACLS does not match the rapid convergence of JADE and SHADE, it clearly improves upon the slower progress of CLPSO, offering a good compromise between solution quality and convergence speed.

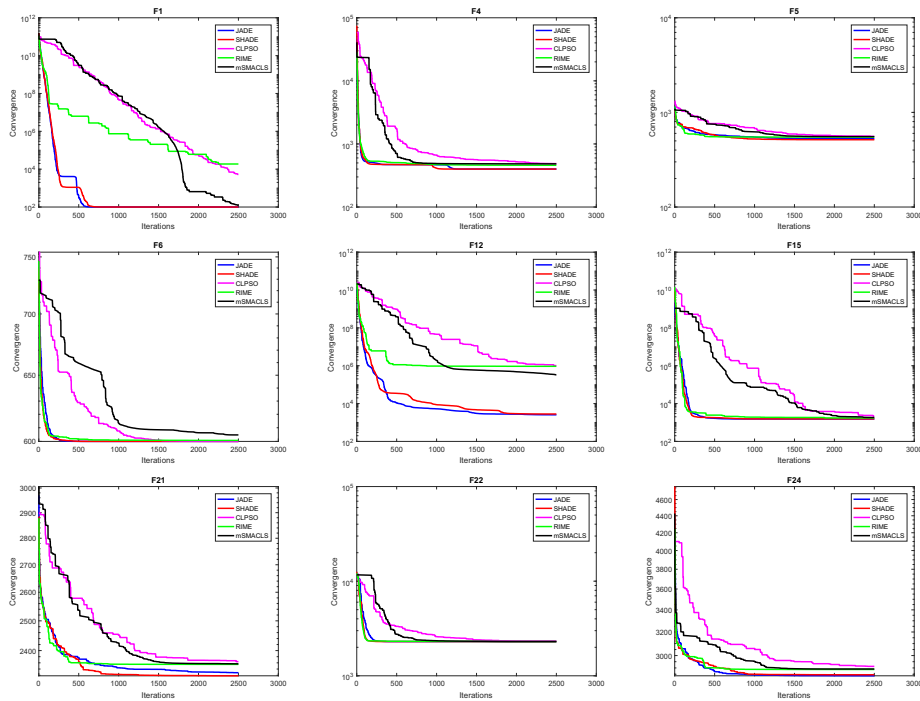


FIGURE 8. Convergence curves on CEC2017 test suites.

#### 4.3.2. *mSMACLS* vs. Chaotic-based algorithms

The current experiments investigate the performance of our algorithm against three chaotic-based improved metaheuristics such as the Chaotic Kbest Gravitational Search Algorithm (CKGSA) [73], Chaotic Hybrid Sine Cosine and Harris Hawk optimizers (CSCAHHO) [74], and the Chaotic Hybrid Butterfly with Particle Swarm Optimizers (HPSOBOA) [75].

The results are presented in Table 13, *mSMACLS* clearly outperforms the other chaotic-based algorithms in the majority of functions, achieving the best mean results on 14 out of 15 benchmark functions. While CKGSA consistently delivers the weakest performance, both in terms of accuracy and reliability, HPSOBOA surpasses *mSMACLS* on only one function (F22). Among the remaining algorithms, CSCAHHO, although trailing behind *mSMACLS*, stands out as the closest competitor, showing relatively better performance compared to CKGSA and HPSOBOA across most functions.

The experimental results also reveal that *mSMACLS* demonstrates superior stability, achieving the lowest standard deviation on 11 out of 15 functions, which highlights its consistency and robustness across diverse optimization landscapes. These findings confirm *mSMACLS* as the most reliable and effective algorithm among the chaotic-based methods evaluated.

Figure 9 presents the convergence rate of the four algorithms. The convergence curves highlight that *mSMACLS* consistently outperforms all other algorithms, showing a steady and continuous improvement across all benchmark functions. Importantly, *mSMACLS*'s curves do not exhibit early stagnation on most functions, which suggests that the algorithm maintains a high exploration capacity and could potentially achieve even better results with more iterations.

In contrast, CKGSA and HPSOBOA exhibit a significant lack of convergence, with their curves stagnating early or showing minimal improvement, an indication that both algorithms are likely trapped in local optima and unable to escape. CSCAHHO performs slightly better, achieving a more effective convergence than CKGSA

TABLE 13. The results of chaotic-based algorithms on CEC2017.

Function	Metric	CKGSA	CSCAHHO	HPSOBOA	mSMACLS
F1	Mean	1.2336E+11	1.3459E+09	7.8694E+10	<b>4.2512E+03</b>
	Std	2.2811E+11	3.8911E+08	2.0313E+09	4.6828E+03
F3	Mean	9.8819E+09	6.6397E+04	9.4243E+04	<b>3.2304E+02</b>
	Std	3.2702E+10	7.6418E+03	8.2449E+01	2.2978E+01
F4	Mean	5.6307E+04	9.8219E+02	2.6821E+04	<b>4.9615E+02</b>
	Std	1.4564E+04	1.4777E+02	4.6459E+03	1.1225E+01
F5	Mean	1.2046E+03	7.9648E+02	9.7807E+02	<b>6.1482E+02</b>
	Std	4.8640E+01	3.4532E+01	0.1772E+01	3.5183E+01
F6	Mean	7.4337E+02	6.7407E+02	7.0941E+02	<b>6.1307E+02</b>
	Std	1.5310E+01	0.8363E+01	0.2842E+01	0.7931E+01
F11	Mean	3.5773E+05	2.9821E+03	1.0872E+04	<b>1.2590E+03</b>
	Std	8.8076E+05	6.5044E+02	5.9439E+01	3.5135E+01
F12	Mean	3.0297E+10	4.4024E+08	2.3665E+10	<b>2.0390E+06</b>
	Std	8.5418E+09	3.3899E+08	3.1263E+09	1.4563E+06
F13	Mean	3.5360E+10	5.1099E+07	3.4655E+10	<b>3.0341E+04</b>
	Std	1.3470E+10	4.1314E+07	5.0037E+09	2.4012E+04
F14	Mean	1.0393E+08	1.1146E+06	8.9990E+07	<b>6.2298E+04</b>
	Std	1.0389E+08	8.6257E+05	5.7245E+07	3.8637E+04
F15	Mean	8.8371E+09	1.0191E+06	1.9277E+09	<b>2.7536E+04</b>
	Std	3.8086E+09	9.2666E+05	1.9831E+08	1.5065E+04
F21	Mean	2.9568E+03	2.6047E+03	2.9070E+03	<b>2.3960E+03</b>
	Std	7.0016E+01	4.4360E+01	0.9687E+01	2.7265E+01
F22	Mean	1.2543E+04	5.2135E+03	<b>1.0613E+04</b>	5.6354E+04
	Std	4.6795E+02	2.5340E+03	4.8121E+01	1.6103E+03
F23	Mean	4.1302E+03	3.2100E+03	3.7453E+03	<b>2.7572E+03</b>
	Std	3.6883E+02	1.0552E+02	8.0696E+01	2.3578E+01
F24	Mean	4.5949E+03	3.3298E+03	4.4737E+03	<b>2.9431E+03</b>
	Std	3.7869E+02	7.5318E+01	2.3388E+02	3.2370E+01
F25	Mean	1.9048E+04	3.0845E+03	7.3777E+03	<b>2.8875E+03</b>
	Std	4.6077E+03	4.4460E+01	1.0228E+03	0.1900E+01

and HPSOBOA, but still stagnates at an early stage when compared to mSMACLS, limiting its ability to reach high-quality solutions in the later iterations.

Overall, these results emphasize mSMACLS's superior convergence dynamics, which reflect a well-maintained balance between exploration and exploitation throughout the optimization process.

#### 4.4. mSMACLS for engineering design problems

As we know, any novel or improved optimization algorithm should be tested in real-world problems. For this reason, three constrained engineering design problems including Pressure vessel, Speed reducer, and I-beam vertical deflection problems, are used to verify the applicability of mSMACLS in real-world problems. The mathematical description of these problems, their graphical representation, and the obtained results are given in the following subsections.

##### 4.4.1. Pressure vessel problem

This problem is a minimization problem in which the objective is to minimize the cost of a cylinder-shaped pressure vessel by manipulating four design variables, the thickness of the shell ( $T_s$ ), the head thickness ( $T_h$ ),

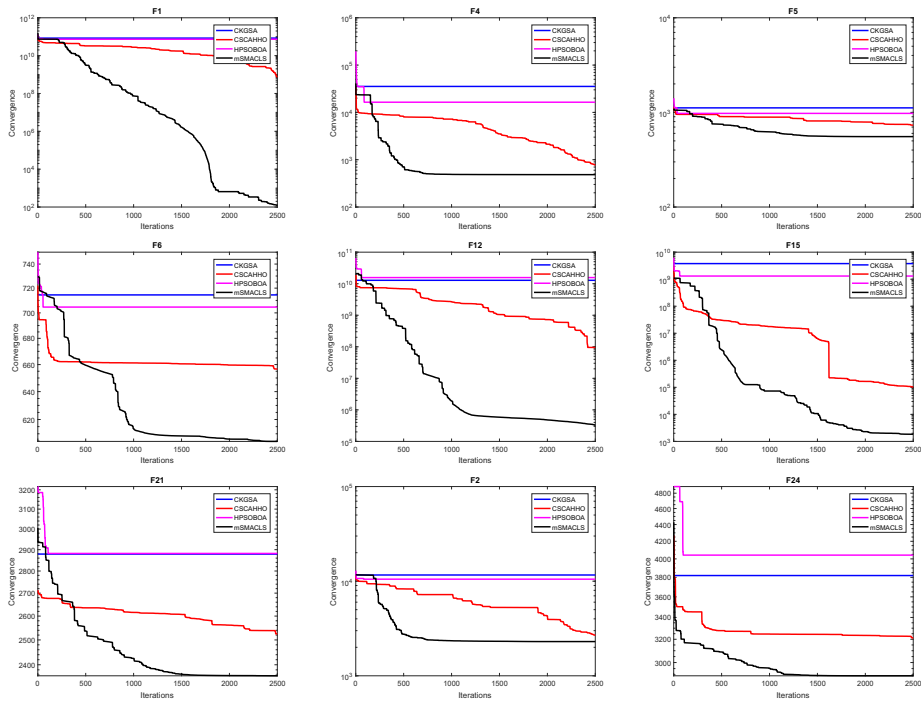


FIGURE 9. Convergence curves of chaotic-based algorithms on CEC2017 test suites.

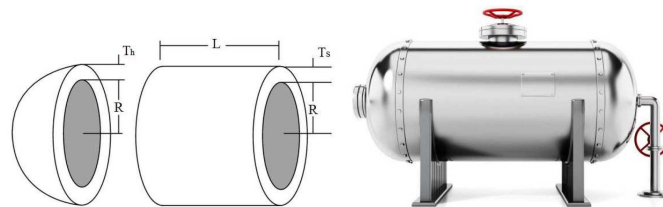


FIGURE 10. Pressure vessel design problem.

the inner radius ( $R$ ), and the length of cylindrical unit ( $L$ ). Its mathematical model is as follows (Fig. 10):

$$f(T_s, T_h, R, L) = 0.6224T_sRL + 1.7781T_hR^2 + 3.1661T_s^2L + 19.84T_s^2T_h$$

Subject to:

$$\begin{aligned} g_1(T_s, R) &= -T_s + 0.0193R \leq 0 \\ g_2(T_h, R) &= -T_h + 0.0095R \leq 0 \\ g_3(R, L) &= -\pi R^2L - \frac{4}{3}\pi R^3 + 1296 \times 10^3 \leq 0 \\ g_4(L) &= L - 240 \leq 0 \end{aligned}$$

where,  $0 \leq T_s, T_h \leq 99$ , and  $0 \leq R, L \leq 200$ .

The obtained results, including the optimal design variables and the best fitness function for pressure vessel design problem, are reported in Table 14. It can be seen that mSMACLS and TLBO algorithms get approxi-

TABLE 14. Comparison of results on pressure vessel design.

Algorithm	Design variables				Optimum cost
	$T_s$	$T_h$	$R$	$L$	
mSMACLS	<b>0.778178103711504</b>	<b>0.384667742401682</b>	<b>40.320060163183456</b>	<b>199.994170489339</b>	<b>5885.402471561636</b>
AHA	0.826553136874047	0.408171986438046	42.622357264475085	173.8503919473286	6084.258799830543
GWO	0.783900146865949	0.390933753981144	40.551640784373674	196.8896590169518	5916.019231628435
HHO	0.868432830501544	0.448923192942521	44.988278455077000	143.8401418869510	6129.909877181706
MRFO	0.804192655203123	0.399887985390306	41.640621777394530	182.8757083307062	5953.213006162547
TLBO	0.778182076150101	0.384657192775173	40.320202030579400	199.9932573227911	5885.404461211394
WOA	1.402708642177636	0.624453175826076	65.225508268659450	10.0000	7901.748320105698

mately similar results. However, mSMACLS yields the optimal cylindrical pressure vessel design, with a cost of 5885.402471 at the design variables  $T_s = 0.778178$ ,  $T_h = 0.384667$ ,  $R = 40.320060$ , and  $L = 199.994170$ . Overall, the obtained findings demonstrate that the proposed mSMACLS is the preferred tool for pressure vessel design problem compared to the other selected algorithms.

#### 4.4.2. Speed reducer problem

The aim of this problem is to minimize the weight of the speed reducer by handling seven variables including the face width ( $b$ ), module of teeth ( $m$ ), the number of teeth in the pinion ( $z$ ), length of the first shaft between bearings ( $l_1$ ), length of the second shaft between bearings ( $l_2$ ), the diameter of first shafts ( $d_1$ ), and the diameter of second shafts ( $d_2$ ). The mathematical formulation of this problem is given as (Fig. 11):

$$f(b, m, z, l_1, l_2, d_1, d_2) = 0.7854bm^2(3.3333z^2 + 14.9334z - 43.0934) - 1.508b(d_1^2 + d_2^2) + 7.4777(d_1^3 + d_2^3) + 0.7854(l_1d_1^2 + l_2d_2^2)$$

Subject to:

$$\begin{aligned} g_1(b, m, z) &= \frac{27}{bm^2z} - 1 \leq 0, & g_2(b, m, z) &= \frac{397.5}{bm^2z^2} - 1 \leq 0 \\ g_3(m, z, l_1, d_1) &= \frac{1.93l_1^3}{md_1^4z} - 1 \leq 0, & g_4(m, z, l_2, d_2) &= \frac{1.93l_2^3}{md_2^4z} - 1 \leq 0 \\ g_5(m, z, l_1, d_1) &= \frac{\sqrt{\left(\frac{745l_1}{mz}\right)^2 + 16.9 \times 10^6}}{110d_1^3} - 1 \leq 0 \\ g_6(m, z, l_2, d_2) &= \frac{\sqrt{\left(\frac{745l_2}{mz}\right)^2 + 157.5 \times 10^6}}{85d_2^3} - 1 \leq 0 \\ g_7(m, z) &= \frac{mz}{40} - 1 \leq 0, & g_8(b, m) &= \frac{5m}{b} - 1 \leq 0, & g_9(b, m) &= \frac{b}{12m} - 1 \leq 0 \\ g_{10}(l_1, d_1) &= \frac{1.5d_1 + 1.9}{l_1} - 1 \leq 0, & g_{11}(l_2, d_2) &= \frac{1.1d_2 + 1.9}{l_2} - 1 \leq 0 \end{aligned}$$

where,  $2.6 \leq b \leq 3.6$ ,  $0.7 \leq m \leq 0.8$ ,  $17 \leq z \leq 28$ ,  $7.3 \leq l_1, l_2 \leq 8.3$ ,  $2.9 \leq d_1 \leq 3.9$ , and  $5 \leq d_2 \leq 5.5$ .

The results of algorithms on speed reducer design problem are listed in Table 15. This table reveals that TLBO algorithm has obtained the optimal weight of 2994.4711, while the proposed mSMACLS algorithm has ranked the second with a weight of speed reducer design of 2994.5703. Hence, these results demonstrate that mSMACLS can be a good alternative to solve speed reducer design problem.

#### 4.4.3. I-beam vertical deflection problem

I-beam design is a minimization problem in which the goal is to reduce the vertical deflection of the I-beam via the manipulation of four design variables such as beam thickness ( $b$ ), flange width ( $h$ ), web thickness ( $t_w$ ),

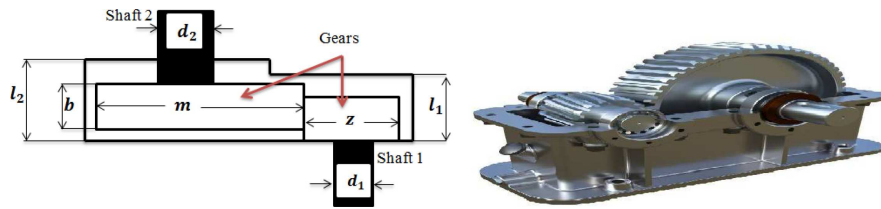


FIGURE 11. Speed reducer design problem.

TABLE 15. Comparison of results on speed reducer design.

Algorithm	Design variables							Optimum weight
	$b$	$m$	$z$	$l_1$	$l_2$	$d_1$	$d_2$	
mSMACLS	3.5000	0.7000	17.0000	7.3000	7.7169	3.3502	5.2867	2994.5703
AHA	3.5025	0.7001	17.0073	7.3646	7.7970	3.3534	5.2883	3001.6412
GWO	3.5124	0.7000	17.0000	7.3000	8.0405	3.3511	5.2985	3014.3547
HHO	3.5765	0.7145	21.6296	8.1629	7.7150	3.5683	5.2861	3133.7276
MRFO	3.5006	0.7000	17.0019	7.3947	7.7729	3.3545	5.2870	2998.5086
TLBO	<b>3.5000</b>	<b>0.7000</b>	<b>17.0000</b>	<b>7.3000</b>	<b>7.7153</b>	<b>3.3502</b>	<b>5.2866</b>	<b>2994.4711</b>
WOA	3.5340	0.7019	26.0971	8.2141	7.7368	3.5412	5.2859	3099.5772

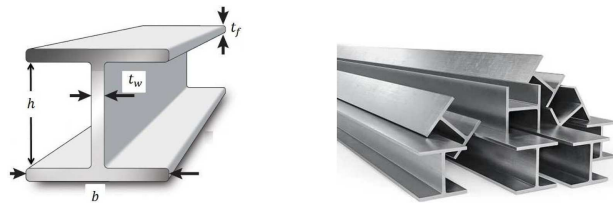


FIGURE 12. I-beam vertical deflection problem.

and flange thickness ( $t_f$ ). Its mathematical model is as follows (Fig. 12):

$$f(b, h, t_w, t_f) = \frac{5000}{\frac{t_w(b-2t_f)^3}{12} + \frac{ht_f^3}{6} + 2ht_f \frac{(b-2t_f)^2}{4}}$$

Subject to:

$$g(b, h, t_w, t_f) = 2ht_w + t_w(b - 2t_f) \leq 0$$

where,  $10 \leq b \leq 50$ ,  $10 \leq h \leq 80$ ,  $0.9 \leq t_w, t_f \leq 5$ .

Table 16 presents the results of the compared algorithms on I-beam vertical deflection design problem. It can be observed that mSMACLS has achieved the optimum vertical deflection of 0.01713017 at the design variables  $b = 50.0000$ ,  $h = 80.0000$ ,  $t_w = 1.475600$ , and  $t_f = 3.346828$ . Moreover, TLBO algorithm has proven to be the nearest competitor to mSMACLS compared to the remaining selected algorithms. Accordingly, the proposed mSMACLS is the preferred algorithm for for reducing the vertical deflection of I-beam design problem.

## 5. CONCLUSION AND FUTURE WORK

In this article, a new improved Slime Mould Algorithm is proposed to solve global optimization tasks. The proposed mSMACLS incorporates a novel mutated Chaotic Local Search to maintain a good equilibrium between

TABLE 16. Comparison of results on I-beam vertical deflection design.

Algorithm	Design variables				Optimum cost
	$b$	$h$	$t_w$	$t_f$	
mSMACLS	<b>50.0000</b>	<b>80.0000</b>	<b>1.475600865206976</b>	<b>3.346828933659509</b>	<b>0.017130177226180</b>
AHA	49.850190306614635	79.723084958362620	1.299162757571804	3.204087384127902	0.017963501147718
GWO	50.0000	80.0000	1.464194008265208	3.324720661279445	0.017228171692087
HHO	49.908673800842920	78.104851187869680	1.504487221655084	3.357443971792722	0.017566702540271
MRFO	49.961419909197700	78.139992129549750	1.500614884190163	3.363277978813137	0.017493027284533
TLBO	50.0000	80.0000	1.475540913529209	3.346723329199981	0.017130642194437
WOA	47.154322937883390	58.107550137590100	1.920754845258790	3.591827638452064	0.025190071359208

exploration and exploitation. Ten different chaotic maps have been investigated. The first experiments show that the proposed mCLS strategy can significantly improve the basic SMA algorithm, especially with Logistic map. The proposed mSMACLS is then compared to well-known algorithms on classical and more complex test functions, and the simulation results demonstrated that mSMACLS is significantly superior or at least very competitive to the selected state-of-the-art metaheuristics. Finally, the algorithm is applied to three classical engineering design problems, and the results demonstrate that the proposed mSMACLS can be a potential tool for handling real-world applications.

However, and as any metaheuristic, we have observed that the proposed mSMACLS suffers from some limitations such as the high execution time especially in high-dimensional problems, and the inconstancy in some complex problems, meaning that mSMACLS can be further improved. Under this context, the use of a dynamic CLS strategy or the fine-tune of CLS radius parameter can be an interesting idea. On the other hand, we plan to apply mSMACLS to other real-world challenges, as optimal power flow problem. Also, a multi-objective version of this algorithm may be considered in the future.

#### FUNDING

The authors consciously ensure that this manuscript received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

#### CONFLICT OF INTEREST

The authors declare that they have no competing interests.

#### DATA AVAILABILITY STATEMENT

The source code of this article is made publicly available by the authors at <https://www.mathworks.com/matlabcentral/fileexchange/182303-slime-mould-algorithm-based-mutated-chaotic-local-search>.

#### REFERENCES

- [1] X.-S. Yang, A.H. Gandomi, S. Talatahari and A.H. Alavi, Metaheuristics in Water, Geotechnical and Transport Engineering. Newnes (2012).
- [2] K. Hussain, M.N.M. Salleh, S. Cheng and Y. Shi, On the exploration and exploitation in popular swarm-based metaheuristic algorithms. *Neural Comput. App.* **31** (2019) 7665–7683.
- [3] X. Zhao, F. Yang, Y. Han and Y. Cui, An opposition-based chaotic salp swarm algorithm for global optimization. *IEEE Access* **8** (2020) 36485–36501.
- [4] X.-S. Yang, Nature-inspired optimization algorithms: challenges and open problems. *J. Comput. Sci.* **46** (2020) 101104.
- [5] X.-S. Yang and M. Karamanoglu, Swarm intelligence and bio-inspired computation: an overview, in *Swarm Intelligence and Bio-Inspired Computation*. Elsevier (2013) 3–23.
- [6] S. Mirjalili, S.M. Mirjalili and A. Lewis, Grey wolf optimizer. *Adv. Eng. Softw.* **69** (2014) 46–61.
- [7] S. Mirjalili and A. Lewis, The whale optimization algorithm. *Adv. Eng. Softw.* **95** (2016) 51–67.

- [8] W. Zhao, Z. Zhang and L. Wang, Manta ray foraging optimization: an effective bio-inspired optimizer for engineering applications. *Eng. App. Artif. Intell.* **87** (2020) 103300.
- [9] L. Abualigah, D. Yousri, M. Abd Elaziz, A.A. Ewees, M.A. Al-Qaness and A.H. Gandomi, Aquila optimizer: a novel meta-heuristic optimization algorithm. *Comput. Ind. Eng.* **157** (2021) 107250.
- [10] L. Wang, Q. Cao, Z. Zhang, S. Mirjalili and W. Zhao, Artificial rabbits optimization: a new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. *Eng. App. Artif. Intell.* **114** (2022) 105082.
- [11] M. Dehghani, Z. Montazeri, E. Trojovská and P. Trojovský, Coati optimization algorithm: a new bio-inspired metaheuristic algorithm for solving optimization problems. *Knowl.-Based Syst.* **259** (2023) 110011.
- [12] A. Seyyedabbasi and F. Kiani, Sand cat swarm optimization: a nature-inspired algorithm to solve global optimization problems. *Eng. Comput.* **39** (2023) 2627–2651.
- [13] E.-S.M. El-kenawy, N. Khodadadi, S. Mirjalili, A.A. Abdelhamid, M.M. Eid and A. Ibrahim, Greylag goose optimization: nature-inspired optimization algorithm. *Expert Syst. App.* **238** (2024) 122147.
- [14] S. Mirjalili, Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. *Knowl.-Based Syst.* **89** (2015) 228–249.
- [15] H. Zamani, M.H. Nadimi-Shahraki, S. Mirjalili, F. Soleimanian Gharehchopogh and D. Oliva, A critical review of moth-flame optimization algorithm and its variants: structural reviewing, performance evaluation, and statistical analysis. *Arch. Comput. Methods Eng.* **31** (2024) 2177–2225.
- [16] J. Lian, G. Hui, L. Ma, T. Zhu, X. Wu, A.A. Heidari, Y. Chen and H. Chen, Parrot optimizer: algorithm and applications to medical problems. *Comput. Biol. Med.* **172** (2024) 108064.
- [17] C. Yuan, D. Zhao, A.A. Heidari, L. Liu, Y. Chen, Z. Wu and H. Chen, Artemisinin optimization based on malaria therapy: algorithm and applications to medical image segmentation. *Displays* **84** (2024) 102740.
- [18] C. Yuan, D. Zhao, A.A. Heidari, L. Liu, Y. Chen and H. Chen, Polar lights optimizer: algorithm and applications in image segmentation and feature selection. *Neurocomputing* **607** (2024) 128427.
- [19] H. Su, D. Zhao, A.A. Heidari, L. Liu, X. Zhang, M. Mafarja and H. Chen, RIME: a physics-based optimization. *Neurocomputing* **532** (2023) 183–214.
- [20] D. Zouache, A. Got and H. Drias, An external archive guided Harris Hawks optimization using strengthened dominance relation for multi-objective optimization problems. *Artif. Intell. Rev.* **56** (2023) 2607–2638.
- [21] A. Got, D. Zouache and A. Moussaoui, MOMRFO: multi-objective manta ray foraging optimizer for handling engineering design problems. *Knowledge-Based Systems* **237** (2022) 107880.
- [22] L. Allou, D. Zouache, K. Amroun and A. Got, A novel epsilon-dominance Harris Hawks optimizer for multi-objective optimization in engineering design problems. *Neural Comput. App.* **34** (2022) 17007–17036.
- [23] F. Berrah, M. Chebila, F. Innal and A. Got, Cost effective analysis of the design of safety instrumented systems using manta-ray foraging optimization algorithm. *Int. J. Saf. Secur. Eng.* **13** (2023) 975–986.
- [24] D. Zouache, A. Got, D. Alarabiat, L. Abualigah and E.-G. Talbi, A novel multi-objective wrapper-based feature selection method using quantum-inspired and swarm intelligence techniques. *Multimedia Tools App.* **83** (2024) 22811–22835.
- [25] A. Got, D. Zouache, A. Moussaoui, L. Abualigah and A. Alsayat, Improved manta ray foraging optimizer-based SVM for feature selection problems: a medical case study. *J. Bionic Eng.* **21** (2024) 409–425.
- [26] A. Got, N.A. Houacine, D. Zouache and H. Drias, An optimized svm with feature selection using swarm intelligence technique, in 2024 International Conference of the African Federation of Operational Research Societies (AFROS). IEEE (2024) 1–5.
- [27] P. Visu, T.S. Praba, N. Sivakumar, R. Srinivasan and T. Sethukarasi, Bio-inspired dual cluster heads optimized routing algorithm for wireless sensor networks. *J. Ambient Intell. Human. Comput.* **12** (2021) 3753–3761.
- [28] J. Ma, Z. Bi, T.O. Ting, S. Hao and W. Hao, Comparative performance on photovoltaic model parameter identification via bio-inspired algorithms. *Sol. Energy* **132** (2016) 606–616.
- [29] M. Hemici, D. Zouache, B. Brahmi, A. Got and H. Drias, A decomposition-based multiobjective evolutionary algorithm using simulated annealing for the ambulance dispatching and relocation problem during COVID-19. *Appl. Soft Comput.* **141** (2023) 110282.
- [30] C. Khelfa and I. Khennak, A survey on recent optimization strategies in ambulance dispatching and relocation problems, in Artificial Intelligence Doctoral Symposium, edited by H. Drias, F. Yalaoui and A. Hadjali. Springer Nature Singapore, Singapore (2023) 192–203.
- [31] L.S. Bendimerad, N.A. Houacine and H. Drias, Swarm intelligent approaches for ambulance dispatching and emergency calls covering: application to COVID-19 spread in Saudi Arabia, in Proceedings of the 13th International Conference on Soft Computing and Pattern Recognition (SoCPaR 2021), edited by A. Abraham, A. Engelbrecht,

- F. Scotti, N. Gandhi, P. Manghirmalani Mishra, G. Fortino, V. Sakalauskas and S. Pllana. Springer International Publishing, Cham (2022) 617–626.
- [32] X. Jin, T. He and Y. Lin, Multi-objective model selection algorithm for online sequential ultimate learning machine. *EURASIP J. Wireless Commun. Networking* **2019** (2019) 1–7.
- [33] H. Yu, K. Yuan, W. Li, N. Zhao, W. Chen, C. Huang, H. Chen and M. Wang, Improved butterfly optimizer-configured extreme learning machine for fault diagnosis. *Complexity* **2021** (2021) 6315010.
- [34] X. Liu, H. Huang and J. Xiang, A personalized diagnosis method to detect faults in gears using numerical simulation and extreme learning machine. *Knowl.-Based Syst.* **195** (2020) 105653.
- [35] S. Sharma, N. Khodadadi, A.K. Saha, F.S. Gharehchopogh and S. Mirjalili, Non-dominated sorting advanced butterfly optimization algorithm for multi-objective problems. *J. Bionic Eng.* **20** (2023) 819–843.
- [36] F.S. Gharehchopogh, B. Abdollahzadeh and B. Arasteh, An improved farmland fertility algorithm with hyper-heuristic approach for solving travelling salesman problem. *CMES-Comput. Model. Eng. Sci.* **135** (2023) 1981.
- [37] M. Ayar, A. Isazadeh, F. S. Gharehchopogh, and M. Seyedi, Nsica: Multi-objective imperialist competitive algorithm for feature selection in arrhythmia diagnosis, *Computers in Biology and Medicine*, **161** (2023) 107025.
- [38] F.A. Özbay, E. Özbay and F.S. Gharehchopogh, An improved artificial rabbits optimization algorithm with chaotic local search and opposition-based learning for engineering problems and its applications in breast cancer problem. *CMES-Comput. Model. Eng. Sci.* **141** (2024) 1067.
- [39] R. Caponetto, L. Fortuna, S. Fazzino and M.G. Xibilia, Chaotic sequences to improve the performance of evolutionary algorithms. *IEEE Trans. Evol. Comput.* **7** (2003) 289–304.
- [40] D. Yang, G. Li and G. Cheng, On the efficiency of chaos optimization algorithms for global optimization. *Chaos Solitons Fractals* **34** (2007) 1366–1375.
- [41] M.S. Tavazoei and M. Haeri, Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms. *Appl. Math. Comput.* **187** (2007) 1076–1085.
- [42] G. Kaur and S. Arora, Chaotic whale optimization algorithm. *J. Comput. Design Eng.* **5** (2018) 275–284.
- [43] M. Kohli and S. Arora, Chaotic grey wolf optimization algorithm for constrained optimization problems. *J. Comput. Design Eng.* **5** (2018) 458–472.
- [44] S. Arora and P. Anand, Chaotic grasshopper optimization algorithm for global optimization. *Neural Comput. App.* **31** (2019) 4385–4405.
- [45] Y. Wang, X. Zhang, D.-J. Yu, Y.-J. Bai, J.-P. Du and Z.-T. Tian, Tent chaotic map and population classification evolution strategy-based dragonfly algorithm for global optimization. *Math. Prob. Eng.* **2022** (2022) 2508414.
- [46] Y. Xu, H. Chen, A.A. Heidari, J. Luo, Q. Zhang, X. Zhao and C. Li, An efficient chaotic mutative moth-flame-inspired optimizer for global optimization tasks. *Expert Syst. App.* **129** (2019) 135–155.
- [47] S. Saha and V. Mukherjee, A novel quasi-oppositional chaotic antlion optimizer for global optimization. *Appl. Intell.* **48** (2018) 2628–2660.
- [48] G.-G. Wang, L. Guo, A.H. Gandomi, G.-S. Hao and H. Wang, Chaotic krill herd algorithm. *Inf. Sci.* **274** (2014) 17–34.
- [49] G.I. Sayed, G. Khoriba and M.H. Haggag, A novel chaotic salp swarm algorithm for global optimization and feature selection. *Appl. Intell.* **48** (2018) 3462–3481.
- [50] F. Daqaq, R. Ellaia, M. Ouassaid, H.M. Zawbaa and S. Kamel, Enhanced chaotic manta ray foraging algorithm for function optimization and optimal wind farm layout problem. *IEEE Access* **10** (2022) 78345–78369.
- [51] L.-Y. Chuang, C.-J. Hsiao and C.-H. Yang, Chaotic particle swarm optimization for data clustering. *Expert Syst. App.* **38** (2011) 14555–14563.
- [52] A. Irajli, J. Karimi, S. Keawsawasvong and M.L. Nehdi, Minimum safety factor evaluation of slopes using hybrid chaotic sand cat and pattern search approach. *Sustainability* **14** (2022) 8097.
- [53] W.-F. Gao, S.-Y. Liu and L.-L. Huang, Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique. *Commun. Nonlinear Sci. Numer. Simul.* **17** (2012) 4316–4327.
- [54] L. Wang, L. Zhang, W. Zhao and X. Liu, Parameter identification of a governing system in a pumped storage unit based on an improved artificial hummingbird algorithm. *Energies* **15** (2022) 6966.
- [55] J. Basha, N. Bacanin, N. Vukobrat, M. Zivkovic, K. Venkatachalam, S. Hubálovský and P. Trojovský, Chaotic Harris Hawks optimization with quasi-reflection-based learning: an application to enhance CNN design. *Sensors* **21** (2021) 6654.
- [56] Y. Wang, H. Liu, G. Ding and L. Tu, Adaptive chimp optimization algorithm with chaotic map for global numerical optimization problems. *J. Supercomput.* **79** (2023) 6507–6537.

- [57] Z. Elgamal, A.Q.M. Sabri, M. Tubishat, D. Tbaishat, S.N. Makhadmeh and O.A. Alomari, Improved reptile search optimization algorithm using chaotic map and simulated annealing for feature selection in medical field. *IEEE Access* **10** (2022) 51428–51446.
- [58] Y. Zhang and Y. Mo, Chaotic adaptive sailfish optimizer with genetic characteristics for global optimization. *J. Supercomput.* **78** (2022) 10950–10996.
- [59] O. Chengtian, L. Yujia and Z. Donglin, An adaptive chaotic sparrow search optimization algorithm, in 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE). IEEE (2021) 76–82.
- [60] S. Gao, Y. Yu, Y. Wang, J. Wang, J. Cheng and M. Zhou, Chaotic local search-based differential evolution algorithms for optimization. *IEEE Trans. Syst. Man Cybern.: Syst.* **51** (2019) 3954–3967.
- [61] S. Gupta and K. Deep, An opposition-based chaotic grey wolf optimizer for global optimisation tasks. *J. Exper. Theor. Artif. Intell.* **31** (2019) 751–779.
- [62] D. Jia, G. Zheng and M.K. Khan, An effective memetic differential evolution algorithm based on chaotic local search. *Inf. Sci.* **181** (2011) 3175–3187.
- [63] A. Chhabra, A.G. Hussien and F.A. Hashim, Improved bald eagle search algorithm for global optimization and feature selection. *Alexandria Eng. J.* **68** (2023) 141–180.
- [64] S. Li, H. Chen, M. Wang, A.A. Heidari and S. Mirjalili, Slime mould algorithm: a new method for stochastic optimization. *Future Generation Comput. Syst.* **111** (2020) 300–323.
- [65] W. Zhao, L. Wang and S. Mirjalili, Artificial hummingbird algorithm: a new bio-inspired optimizer with its engineering applications. *Comput. Methods Appl. Mech. Eng.* **388** (2022) 114194.
- [66] A.A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja and H. Chen, Harris Hawks optimization: algorithm and applications. *Future Generation Comput. Syst.* **97** (2019) 849–872.
- [67] R.V. Rao, V.J. Savsani and D. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Comput.-Aided Design* **43** (2011) 303–315.
- [68] K. Price, N. Awad, M. Ali and P. Suganthan, Problem definitions and evaluation criteria for the 100-digit challenge special session and competition on single objective numerical optimization. Technical report, Nanyang Technological University Singapore (2018).
- [69] G. Wu, R. Mallipeddi and P.N. Suganthan, Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization. Technical Report, National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore (2017).
- [70] J. Zhang and A.C. Sanderson, JADE: adaptive differential evolution with optional external archive. *IEEE Trans. Evol. Comput.* **13** (2009) 945–958.
- [71] R. Tanabe and A. Fukunaga, Success-history based parameter adaptation for differential evolution, in 2013 IEEE Congress on Evolutionary Computation. IEEE (2013) 71–78.
- [72] J.J. Liang, A.K. Qin, P.N. Suganthan and S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evol. Comput.* **10** (2006) 281–295.
- [73] H. Mittal, R. Pal, A. Kulhari and M. Saraswat, Chaotic kbest gravitational search algorithm (CKGSA), in 2016 Ninth International Conference on Contemporary Computing (IC3). IEEE (2016) 1–6.
- [74] Y.-J. Zhang, Y.-X. Yan, J. Zhao and Z.-M. Gao, CSCAHHO: chaotic hybridization algorithm of the sine cosine with Harris Hawk optimization algorithms for solving global optimization problems. *PLoS One* **17** (2022) e0263387.
- [75] M. Zhang, D. Long, T. Qin and J. Yang, A chaotic hybrid butterfly optimization algorithm with particle swarm optimization for high-dimensional optimization problems. *Symmetry* **12** (2020) 1800.

**Please help to maintain this journal in open access!**



This journal is currently published in open access under the Subscribe to Open model (S2O). We are thankful to our subscribers and supporters for making it possible to publish this journal in open access in the current year, free of charge for authors and readers.

Check with your library that it subscribes to the journal, or consider making a personal donation to the S2O programme by contacting [subscribers@edpsciences.org](mailto:subscribers@edpsciences.org).

More information, including a list of supporters and financial transparency reports, is available at <https://edpsciences.org/en/subscribe-to-open-s2o>.