Soft Computing
https://doi.org/10.1007/s00500-021-06140-2

OPTIMIZATION

Adaptive opposition slime mould algorithm

Manoj Kumar Naik[1] · Rutuparna Panda[2] · Ajith Abraham[3]

Accepted: 9 August 2021

Abstract
Recently, the slime mould algorithm (SMA) has become popular in function optimization, because it effectively uses exploration and exploitation to reach an optimal solution or near-optimal solution. However, the SMA uses two random search agents from the whole population to decide the future displacement and direction from the best search agents, which limits its exploitation and exploration. To solve this problem, we investigate an adaptive approach to decide whether opposition-based learning (OBL) will be used or not. Sometimes, the OBL is used to further increase the exploration. In addition, it maximizes the exploitation by replacing one random search agent with the best one in the position updating. The suggested technique is called an adaptive opposition slime mould algorithm (AOSMA). The qualitative and quantitative analysis of AOSMA is reported using 29 test functions that consisting of 23 classical test functions and 6 recently used composition functions from the IEEE CEC 2014 test suite. The results are compared with state-of-the-art optimization methods. Results presented in this paper show that AOSMA's performance is better than other optimization algorithms. The AOSMA is evaluated using Wilcoxon's rank-sum test. It also ranked one in Friedman's mean rank test. The proposed AOSMA algorithm would be useful for function optimization to solve real-world engineering problems.

Keywords Soft computing · Slime mould algorithm · Function optimization · Engineering applications

1 Introduction

In the real-world, the resources are limited, and the optimization of the available resources is much desirable. So, optimization is a means to obtain the potential solution for a specific problem. Broadly, based on the mathematical foundation, the optimization problems are classified into two categories: deterministic and stochastic. The deterministic optimization algorithm requires the knowledge of gradient information to obtain the optimal solution, some well-known methods are linear and nonlinear programming. These methods are good for linear search spaces, however, not effective for nonlinear search spaces (Faramarzi and Afshar 2014). Stochastic optimization does not require gradient information; however, they generate and use random variables to obtain the optimal solution. The metaheuristic optimization algorithm is implemented based on a stochastic operator with an advantage of simplicity, flexibility, gradient-free, and problem independent (Mirjalili et al. 2014). The metaheuristic algorithms are gradient-free and problem independent, the problem can be considered as a black box with identified input and output variables. The metaheuristic algorithm also does not require an initial guess to a solution space, which that received major attention in recent years (Naik et al. 2020).

Based on the number of the solution obtained by the metaheuristic algorithm, they are classified into two types (Talbi 2009): a single solution-based and population-based. The single solution-based optimization generates only one solution throughout the optimization stages. On the other hand, population-based optimization generates a set of solutions at every generation of the optimization stages and is mostly inspired by

✉ Rutuparna Panda
r_pparda@yahoo.co.in

Manoj Kumar Naik
naik.manoj.kumar@gmail.com

Ajith Abraham
ajith.abraham@ieee.org

[1] Faculty of Engineering and Technology, Siksha O Anusandhan, Bhubaneswar, Odisha 751030, India

[2] Department of Electronics and Telecommunication Engineering, Veer Surendra Sai University of Technology, Burla, Odisha 768018, India

[3] Machine Intelligence Research Labs, Scientific Network for Innovation and Research Excellence, Auburn, WA 98071-2259, USA

Published online: 23 August 2021

natural phenomena. The well-known example for single solution-based optimization is simulated annealing (SA) (Kirkpatrick et al. 1983), and the population-based optimization is a genetic algorithm (GA) (Holland 1975).

On the other way, based on the source of inspiration, the metaheuristic algorithms are mostly classified into four categories: evolution algorithms (EAs), swarm intelligence (SI), physics-based, and human-based. The EAs are inspired by biological evolution, such as crossover, mutation, and selection. The most popular EAs is a genetic algorithm (GA) (Holland 1975), which is based on the Darwinian theory of evolution. The GA used the crossover to generate the offspring from parents that help to explore the search space. After that, the next generation of parents is evolved by selection. The other popular EAs is differential evolution (DE) (Feoktistov 2006). The SI algorithms are inspired by the intelligent social behavior of organisms living in swarms, herds, schools, or flocks. The most widely used SI algorithms are particle swarm optimization (PSO) (Kennedy and Eberhart 1995), which simulate the bird flocking behaviors with each bird is considered as a candidate solution. Each bird updates its path based on the best one to reach optimal solutions. Some other popular SI algorithms are Cuckoo search (CS) (Yang 2014), ant colony optimization (ACO) (Dorigo and Stützle 2004), firefly algorithm (FA) (Yang 2014), and artificial bee colony (ABC) (Karaboga and Basturk 2008). Physics-based algorithms are inspired by physical laws in nature. The well-known examples of a physics-based algorithm are simulated annealing (SA) (Kirkpatrick et al. 1983) and gravitational search algorithm (GSA) (Rashedi et al. 2009). The SA uses the thermodynamic law of deformation of size due to heating and cooling, whereas GSA uses Newton's gravitational law to obtain the optimal position based on mass, force, and distance between them. The human-based algorithm is based on human interactions and behavior. Some most prominent algorithms in these categories are teaching–learning-based-optimization (TLBO) (Rao et al. 2012) which is based on impact of teachers on learners and tabu search (TS) (Glover 1989, 1990).

Apart from the above presented well-known algorithms, the researchers come up with many more metaheuristic algorithms inspired by the nature to combat the problem in function optimization and engineering problem (Naik et al. 2020). Some of the recent developments of population-based metaheuristic algorithm that are quite good on function optimization include a gray wolf optimizer (GWO) (Mirjalili et al. 2014), whale optimization algorithm (WOA), moth search algorithm (MSA) (Wang 2018), monarch butterfly optimization (MBO) (Wang et al. 2019), squirrel search algorithm (SSA) (Jain et al. 2019), Harris hawks optimization (HHO) (Heidari et al. 2019), sailfish optimizer (SFO) (Shadravan et al. 2019), equilibrium optimizer (EO) (Faramarzi et al. 2020), slime mould algorithm (SMA) (Li et al. 2020), manta ray foraging optimization (MRFO) (Zhao et al. 2020), hunger games search (HGS) (Yang et al. 2021), RUNge Kutta optimizer (RUN) (Ahmadianfar et al. 2021), and Colony Predation Algorithm (CPA) (Tu et al. 2021). The GWO mimics the hunting mechanism such as searching, encircling, and attacking the prey by gray wolves (*Canis lupus*) in a leadership hierarchy. The WOA is inspired by the social and hunting behavior like searching, encircling, and bubble-net attacking of humpback whales (*Megaptera novaeangliae*). The MSA is inspired by the phototaxis movement of a family of moths which belong to order Lepidoptera that follows Lévy flights. The MBO is inspired by the migration behavior of monarch butterflies between two lands (USA and Maxico). The SSA stimulates the foraging behavior of southern flying squirrel (*Glaucomys Volans*) with effective locomotion known as gliding. The HHO has inspired on Harris's hawk (*Parabuteo unicinctus*) cooperative behavior of chasing the prey based on surprise pounce or seven kills strategy. The SFO is a model of the behavior of sailfish (*Istiophorus platyerus*) on group hunting strategy and attacking pray sardine (*Sardinella aurita*). The EO is inspired by a physics principle, controlled mass balance model to estimate the equilibrium state. The SMA is inspired by how the plasmodial slime mould (*Physarum ploycephalum*) establishes an optimal path to reach the solution. The MRFO is inspired on intelligent foraging strategy of manta rays such as chain, cyclone, and somersault. The HGS is designed based on the fitness-wise search method according to the hunger-driven activities shown by animals. The RUN is a metaphor-free optimizer based on slope variation computed by Runge Kutta method widely used in the mathematics. The CPA is based on the community prediction of animal with the strategy such as dispersing prey, encircling prey, assisting the successful predictor, and looking for another target.

As the original optimization algorithms are designed based on basic nature-inspired principles, they possess flaws and constraints. Furthermore, as per the "no free lunch" (NFL) theorem (Wolpert and Macready 1997), one algorithm may not work for different classes of problems. So, there is always a chance of improvement in the performance of the algorithm. This provoked the researchers to enhance the original optimization algorithms with different initial design techniques (Wunnava et al. 2020b), remodeling/modifying the search patterns (Qian et al. 2020; Wunnava et al. 2020a, c; Guha et al. 2020) or hybridizing the optimization algorithms (Zhu et al. 2020). In this context, opposition-based learning (OBL) (Tizhoosh 2005) is a method used to enrich the exploration of any optimization algorithm, because it utilizes the information of opposite relationships among entities to select the best one. The OBL was successfully applied in various optimization algorithms, artificial neural networks, fuzzy systems, and reinforcement learning to improve performance (Mahdavi et al. 2018; Dhargupta et al. 2020). The SMA (Li et al. 2020) is quite an interesting swarm-based optimization algorithm proposed in

2020 with a good performance and convergence. The SMA has been further improved in performance by many researchers and applied in various fields. Some of the prominent works are: CNMSMA (Liu et al. 2021) used the chaotic map with Nelder-Mead simplex strategy to enhance the performance of SMA and used to estimate the parameters of photovoltaic cells; DASMA (Zhao et al. 2021) used the diffusion and association strategy in SMA to enhance the performance and applied in medical multilevel image thresholding to help doctors; WQSMA (Yu et al. 2021) is proposed to boost the original SMA by adding the concept of water-cycle from water-cycle algorithm (WCA) and quantum rotation gate mechanism.

In the same way, the authors of this article are primarily inspired by the performance of SMA (Li et al. 2020) in function optimization. It is noteworthy to mention here that the SMA effectively uses the exploration and exploitation phases to reach an optimal solution or near-optimal solution. However, the SMA uses two random search agents from the whole population to decide the future displacement and direction from the best search agents. This feature of the SMA limits its exploitation and exploration capabilities. This has motivated us to propose a new algorithm for function optimization. In this work, we suggest an adaptive approach to decide whether to use opposition-based learning (OBL) or not. This idea is used to further enrich the exploration capability. In addition, it ensures the maximization of the exploitation phase by replacing one random search agent with the best one in the position updating. The proposed method is coined as an adaptive opposition slime mould algorithm (AOSMA). The qualitative and quantitative analysis of AOSMA is reported using 29 test functions, which are composed of 23 classical test functions (Yao et al. 1999; Naik and Panda 2016) and 6 recent composition functions from the IEEE CEC 2014 test suite (Liang et al. 2013). The results are compared with some recently developed (state-of-the-art) optimization algorithms such as SMA, MRFO, EO, SFO, HHO, SSA, and WOA, which shows AOSMA's superiority among other optimization algorithms. The AOSMA is validated using Wilcoxon's rank-sum test. Further, it is ranked one in Friedman's mean rank test. Exemplar solutions are presented to attract the readers.

The paper is organized as follows. Section 1 is committed to a brief introduction. The proposed work of the development of AOSMA is discussed in Sect. 2. The qualitative and quantitative result analysis of AOSMA with a comparison with the state-of-the-art algorithms are reported in Sect. 3. In Sect. 4, a concluding remark is drawn.

2 Proposed work

The development of the adaptive opposition slime mould algorithm (AOSMA) is based on remodeling the approaching behavior of slime mould as discussed in (Li et al. 2020) with an adaptive decision for opposition-based learning. The SMA is a stochastic optimizer based on the oscillation mode of plasmodial slime mould (*Physarum ploycephalum*). The slime mould uses the oscillation mode along with positive–negative feedback to establish the optimal path to connect food.

2.1 Mathematical formulation of AOSMA

Let us assume N slime moulds are present in the search space with upper boundary (UB) and lower boundary (LB). Then, i th slime mould position in d-dimensions can be expressed as $X_i = \left(x_i^1, x_i^2, \ldots, x_i^d\right), \forall i \in [1, N]$, and fitness (odor) of the ith slime is represented as $f(X_i), \forall i = [1, N]$. So, the position and fitness of N slime mould at the current time (iteration) t is expressed as:

$$X(t) = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^d \\ x_2^1 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \vdots & \vdots \\ x_N^1 & x_N^2 & \cdots & x_N^d \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix} \tag{1}$$

$$f(X) = [f(X_1), f(X_2), \ldots, f(X_N)] \tag{2}$$

The position of slime mould for the next iteration $(t + 1)$ in SMA (Li et al. 2020) is updated using Eq. (3).

$$X_i(t+1) = \begin{cases} X_{\mathrm{LB}}(t) + V_{\mathrm{b}}(W \cdot X_{\mathrm{A}}(t) - X_{\mathrm{B}}(t)) & r_1 \geq \delta \text{ and } r_2 < p_i \\ V_{\mathrm{c}} \cdot X_i(t) & r_1 \geq \delta \text{ and } r_2 \geq p_i \\ \mathrm{rand} \cdot (\mathrm{UB} - \mathrm{LB}) + \mathrm{LB} & r_1 < \delta \end{cases}, \quad \forall i \in [1, N] \tag{3}$$

The X_{LB} represent the local best individual for the current iteration, X_{A} and X_{B} are randomly pooled slime mould from current populations, W as the weight factor, and V_{b} and V_{c} as the random velocity. The r_1 and r_2 are random numbers in the range of $[0, 1]$. The δ is the chance of the slime mould that initializes to a random search location which is fixed at 0.03.

The p_i is the threshold value of i th slime mould that helps to choose the slime mould position using the best individual or itself for the next iteration, which is evaluated as:

$$p_i = \tan h|f(X_i) - f_{\mathrm{GB}}|, \forall i \in [1, N] \tag{4}$$

where $f(X_i)$ is the fitness value of ith slime mould X_i, and the global best fitness value f_{GB} is evaluated using Eq. (5) of the global best position X_{GB}.

$$f_{GB} = f(X_{GB}) \tag{5}$$

Then, the weight W for N slime mould in a current iteration t is determined as:

$$W(\mathrm{SortInd_f}(i)) = \begin{cases} 1 + \mathrm{rand} \cdot \log\left(\frac{f_{\mathrm{LB}} - f(X_i)}{f_{\mathrm{LB}} - f_{\mathrm{LW}}} + 1\right) & 1 \le i \le \frac{N}{2} \\ 1 - \mathrm{rand} \cdot \log\left(\frac{f_{\mathrm{LB}} - f(X_i)}{f_{\mathrm{LB}} - f_{\mathrm{LW}}} + 1\right) & \frac{N}{2} < i \le N \end{cases} \tag{6}$$

where rand is a random number in between $[0, 1]$, f_{LB} as the local best fitness and the local worst fitness value is f_{LW}. The f_{LB} and f_{LW} are determined from the fitness value f given in Eq. (2). For a minimization problem, sort the fitness value in ascending order as:

$$[\mathrm{Sort_f}, \mathrm{SortInd_f}] = \mathrm{sort}(f) \tag{7}$$

The local best fitness f_{LB} and corresponding local best individual X_{LB} are extracted as:

$$f_{\mathrm{LB}} = f(\mathrm{Sort_f}(1)) \tag{8}$$

$$X_{\mathrm{LB}} = X(\mathrm{SortInd_f}(1)) \tag{9}$$

The local worst fitness f_{LW} is extracted as:

$$f_{\mathrm{LW}} = f(\mathrm{Sort_f}(N)) \tag{10}$$

The V_{b} and V_{c} are the random velocity chosen from the continuous uniform distribution in the interval $[-b, b]$ and $[-c, c]$. The b and c for the iteration t are chosen as:

$$b = \arctan\ h\left(-\left(\frac{t}{T}\right) + 1\right) \tag{11}$$

and

$$c = 1 - \frac{t}{T} \tag{12}$$

where the maximum iteration is T.

The SMA has shown promising exploration and exploitation capability to solve the function optimization and engineering design problem as discussed in (Li et al. 2020). However, there is a scope of improvements in the search process of the optimal food path for slime mould as described by Eq. (3). The next generation position updating rule of slime mould in SMA mainly depends on three cases based on δ and p_i, which are:

Case 1 When $r_1 \ge \delta$ and $r_2 < p_i$, the slime mould search trajectory guided by the local best individual X_{LB} and two randomly pooled slime X_{A} and X_{B} from search space of N slime mould with velocity V_{b}. This step helps in balancing exploration and exploitation.

Case 2 When $r_1 \ge \delta$ and $r_2 \ge p_i$, the slime mould trajectory is guided by the position of itself with a velocity V_c. This step helps in exploitation.

Case 3 When $r_1 < \delta$, the slime mould reinitializes again in the search space, this helps in exploration.

Case 1 shows that X_{A} and X_{B} are two randomly pooled slime mould, the chances of solutions we get are not guided properly to explore and exploit. This limitation can be overcome by replacing one randomly pooled slime mould X_{A} with the local best individual X_{LB}. So, the ith $(i = 1, 2, \ldots, N)$ slime mould position updating rule of Eq. (3) is remodeled as Eq. (13).

$$Xn_i(t) = X_{\mathrm{LB}}(t) + V_{\mathrm{b}}(W \cdot X_{\mathrm{LB}}(t) - X_{\mathrm{B}}(t)), \quad \text{if } r_1 \ge \delta \text{ and } r_2 < p_i \tag{13a}$$

$$Xn_i(t) = V_{\mathrm{c}} \cdot X_i(t), \quad \text{if } r_1 \ge \delta \text{ and } r_2 \ge p_i \tag{13b}$$

$$Xn_i(t) = \mathrm{rand} \cdot (\mathrm{UB} - \mathrm{LB}) + \mathrm{LB}, \quad \text{if } r_1 < \delta \tag{13c}$$

As per *Case 2*, the slime mould exploits a nearby place, so it may be following a path of lower fitness value than before. To overcome this limitation, an adaptive decision mechanism can be a better choice. Based on *Case 3*, the SMA has a provision for dedicated exploration, however as δ is a small value, the exploration is limited. To overcome this limitation, we need to supplement extra exploration to SMA, which help it to overcome the local minima. As a combined effort to overcome the limitations of Case 2 and Case 3, we use an adaptive decision strategy for whether it is needed to explore furthermore using OBL (Tizhoosh 2005).

2.1.1 Opposition-based learning

The OBL used an estimate Xo_i in the search space which is the exact opposite of the position Xn_i for each slime mould $(i = 1, 2, \ldots, N)$ and compare it to update the position of the next iterations. This step helps to avoid the chances of being trapped in the local minima with improved convergence. So, the Xo_i for ith slime mould in j th dimension is estimated as:

$$Xo_i^j(t) = \min(Xn_i(t)) + \max(Xn_i(t)) - Xn_i^j(t) \tag{14}$$

where $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, d$.

Let us represent Xs_i is the ith slime mould position, which is selected for minimization problem as:

$$Xs_i(t) = \begin{cases} Xo_i(t) & \text{if } f(Xo_i(t)) < f(Xn_i(t)) \\ Xn_i(t) & \text{if } f(Xo_i(t)) \ge f(Xn_i(t)) \end{cases} \tag{15}$$

2.1.2 Adaptive decision strategy

When the slime mould is following a decedent nutrient path, an adaptive decision is taken based on the current fitness value $f(Xn_i(t))$ and old fitness value $f(X_i(t))$. The adaptive decision helps to supplement extra exploration when needed via OBL. Finally, the position for the next iteration is updated using the adaptive decision strategy of AOSMA, which is modeled as:

$$X_i(t+1) = \begin{cases} Xn_i(t) & \text{if } f(Xn_i(t)) \leq f(X_i(t)) \\ Xs_i(t) & \text{if } f(Xn_i(t)) > f(X_i(t)) \end{cases}, \forall i \in [1, N] \tag{16}$$

Interestingly, the proposed AOSMA enhances the efficiency of SMA with the help of an adaptive decision strategy to whether OBL is needed during the search trajectory. The pseudocode is presented in Sect. 2.2. A flowchart is shown in Fig. 1.

2.2 Pseudocode of AOSMA

In the beginning, identity the number of slimes mould N to be employed, objective function f of dimension d, search space boundary UB and LB, and the maximum number of iterations allowed as T. The pseudocode of the suggested AOSMA is as follows:

Begin
Inputs: N, d, T, δ and select an objective function f with search boundary range $[LB, UB]$.
Outputs: X_{GB} and f_{GB}
Initialization: Randomly initialize the slime mould $X_i = \left(x_i^1, x_i^2, \cdots, x_i^d\right), \forall i \in [1, N]$ within the search boundary UB and LB for initial iteration $t = 1$.
while $(t \leq T)$

→ Calculate the fitness values $f(X)$ of N slime mould using Eq. (2).

→ Sort the fitness value using Eq. (7).

→ Update the local best fitness f_{LB} using Eq. (8) and corresponding local best individual X_{LB} using Eq. (9).

→ Update the local worst fitness f_{LW} using Eq. (10).

→ Update the global best fitness f_{GB} and corresponding global best individual X_{GB}.

→ Update the weight W using Eq. (6).

→ Update the b using Eq. (11) and c using Eq. (12).

for (each slime mould $i = 1{:}N$)

- Generate random numbers r_1 and r_2.
- Generate the threshold value p_i using Eq. (4).
- Evaluate new slime mould position Xn_i using Eq. (13).
- Evaluate fitness value of new slime mould $f(Xn_i)$.

if $\left(f(Xn_i) > f(X_i)\right)$ // *Adaptive decision strategy*

- Estimate Xo_i using Eq. (14). // *Opposition based learning*
- Select Xs_i using Eq. (15).

end

- Update the next iteration slime mould X_i using Eq. (16).

end

→ Next iteration $t = t + 1$

end
Return: Global best solution space X_{GB}.

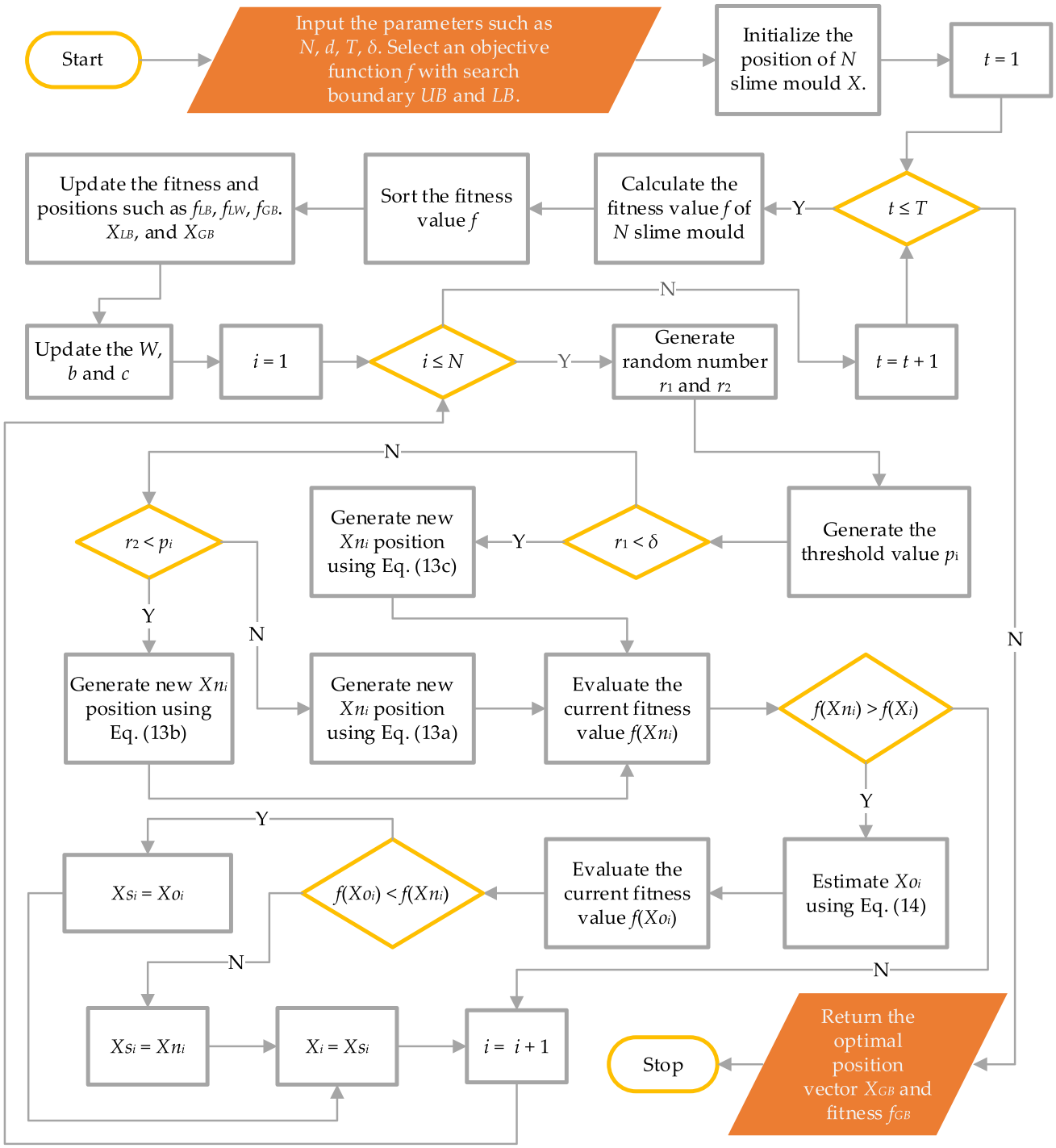

Fig. 1 Flowchart of AOSMA

3 Results and discussion

This section presents the performance evaluation of the proposed AOSMA on a set of 29 test functions that include 23 classical test functions (Yao et al. 1999; Wunnava et al. 2020b) and 6 composition test functions from the CEC-2014 test suite (Liang et al. 2013). In this study, the search history, trajectory, and average fitness history, convergence curve, and boxplots are used as a qualitative metric, however, the average fitness value ('Ave') and standard deviation ('Std') are used as a quantitative metric for comparisons and validations. The proposed AOSMA is also validated using a statistical method such as Friedman's mean rank and Wilcoxon rank-sum test.

3.1 Test functions, compared algorithms, and experimental setup

Here, 29 test functions considered for the performance evaluation comprise four core groups of benchmark landscapes: unimodal $(f_1 - f_7)$, multimodal $(f_8 - f_{13})$, multimodal with fixed dimensions $(f_{14} - f_{23})$, and composition (CEC14 − F23 to CEC14 − F28). The unimodal test functions have a unique global optimal solution, which helps to understand the exploitation ability of the optimization algorithm. The multimodal test functions have more than one optimal solution with many local minima, so these test functions help to understand the exploration's ability to obtain a globally optimal solution without trapping in the local minima of the optimization algorithm. The composition test functions having many locally optimal solutions cover hybrid composite shifted, rotated, and extended multimodal test functions. They mimic the elevated complexity in the search domain. These composition test functions help to understand the tradeoff ability between exploitation and exploration to reach a globally optimal solution by avoiding the local optima of the optimization algorithm.

The performance of AOSMA on test functions is compared with some recently developed optimization algorithms such as the SMA (Li et al. 2020), MRFO (Zhao et al. 2020), EO (Faramarzi et al. 2020), SFO (Shadravan et al. 2019), HHO (Heidari et al. 2019), SSA (Jain et al. 2019), and WOA (Mirjalili and Lewis 2016). The experimental parameter settings are shown in Table 1, which are taken as reported in the original work. To maintain consistency, the maximum function evaluation and population size are taken as 15,000 and 30 for all optimization algorithms The results are compared with the help of the average fitness value ('Ave') and standard deviation among the fitness value ('Std') using 51 independent runs of the optimization algorithm.

Table 1 Parameter settings

Algorithm	Parameters
AOSMA / SMA	$\delta = 0.03$
MRFO	$S = 2$
EO	$a_1 = 2$, $a_2 = 1$ and $GP = 0.5$
SFO	$A = 4$ and $\epsilon = 0.001$
HHO	$\beta = 1.5$
SSA	$P_{dp} = 0.1$, $d_g = 0.8$ and $G_c = 1.9$
WOA	$a = [0, 2]$, $b = 1$, and $l = [-1, 1]$

3.2 Qualitative analysis of AOSMA

The qualitative analysis of AOSMA is presented in Fig. 2 which comprises search history, trajectory, and optimization history metrics. This study shows how AOSMA searches for the optimal solutions. The first column of Fig. 2 shows a three-dimensional representation with a contour line of randomly selected unimodal (f_1 and f_7), multimodal (f_{10}), multimodal with fixed dimensions (f_{14}), and composition (CEC14 − F27 and CEC14 − F28) test functions.

The search history qualitative metric is shown in the second column of Fig. 2, which comprise the first two dimensions (x^1 and x^2) of N slime mould for the first iteration to the last iteration. The positions are shown on contour lines of the search space for a better thought. The positions of slime mould are aggregate nearer to the optimal solution space, which reflects that the AOSMA has performed effective exploitation. However, some of the positions of slime mould are also scattered around the search space, which reflects that the AOSMA also performed exploration. From the search history metric, it is observed that the AOSMA has well balanced the exploitation and exploration, that is needed for solving a real-world optimization problem.

The trajectory qualitative metric is shown in the third column of Fig. 2, which comprises the positions of the first slime $X = \{x^1, x^2, \ldots, x^d\}$ in the d-dimensional space within the range $[\mathrm{UB}, \mathrm{LB}]$ of the search boundary. From Fig. 2, it is evident that the positions of slime mould are initialized in a random location that covers the whole search boundary. However, as the generation (iteration) increases, the positions try to converge to the optimal solution value, which describes the exploitation efficiency. Sometimes, the trajectory abruptly changes due to the exploration. Finally, in a later stage (nearer to the maximum iteration), the trajectory gets flattened, which stabilizes the searching and converges to global/local optimum solutions. The AOSMA has a well-tuned performance on exploitation and exploration.

The optimization history (convergence curve) metric is shown in the fourth column of Fig. 2, which describes the fitness value of the best slime mould during the iterations. A decreasing trend in optimization history reveals the effectiveness of the AOSMA.

3.3 AOSMA's comparative performance on test functions

In this section, the results of AOSMA are compared with various optimization algorithms presented in Table 2 for 29 test functions such as unimodal $(f_1 - f_7)$, multimodal

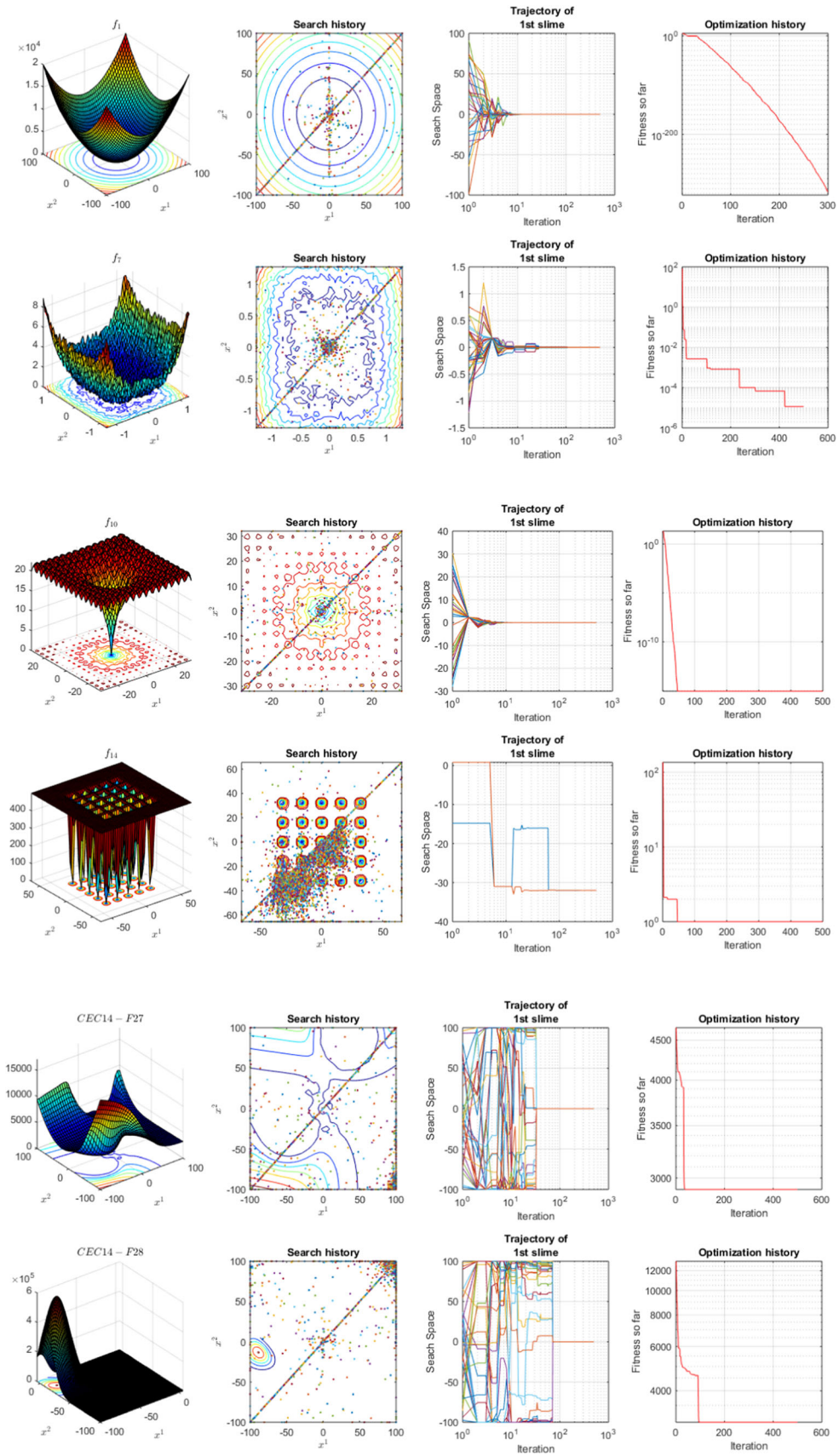
f_1
Search history
Trajectory of 1st slime
Optimization history
Seach Space
Iteration
Fitness so far
f_7
f_{10}
f_{14}
$CEC14-F27$
$CEC14-F28$

Table 2 Results of the optimization algorithm and Friedman mean rank

	Function	Matric	AOSMA	SMA	MRFO	EO	SFO	HHO	SSA	WOA
Unimodal test function (scalable)	f_1	Ave	**0**	1.06E-302	2.13E-246	3.58E-40	6.44E-16	5.31E-59	1.32E-13	1.95E-18
		Std	0	0	0	1.65E-39	1.76E-15	2.56E-58	7.32E-13	3.20E-18
	f_2	Ave	**0**	2.51E-140	2.93E-126	5.04E-24	1.44E-07	1.83E-31	4.34E-07	3.51E-11
		Std	0	1.40E-139	6.40E-126	7.76E-24	1.31E-07	4.29E-31	2.16E-06	3.01E-11
	f_3	Ave	**0**	**0**	2.74E-223	3.38E-09	5.53E-13	7.78E-49	4.82E + 03	4.30E-05
		Std	0	0	0	8.56E-09	1.18E-12	4.33E-48	1.87E + 04	1.31E-04
	f_4	Ave	**0**	6.14E-159	3.32E-122	1.84E-10	8.12E-09	1.05E-31	1.12E-06	8.61E-05
		Std	0	3.14E-158	1.71E-121	3.63E-10	9.99E-09	2.62E-31	6.24E-06	9.17E-05
	f_5	Ave	6.13E + 00	6.38E + 00	2.52E + 01	2.53E + 01	5.93E-02	1.79E-02	**1.19E-06**	2.85E + 01
		Std	10.09457	11.55532	0.384533026	0.221978	0.101776217	0.036307	6.46E-06	10.61531
	f_6	Ave	8.30E-05	0.006317	8.80E-03	9.41E-06	3.002044077	0.000152	**9.07E-11**	0.813679
		Std	5.96E-05	0.003635	4.55E-02	7.25E-06	1.458728427	0.000273	5.05E-10	0.405307
	f_7	Ave	**8.13E-05**	0.000196	2.31E-04	0.001312	0.000179024	0.000212	0.00054182	0.005945
		Std	7.41E-05	0.000189	1.92E-04	0.000683	0.00016296	2.86E-04	0.00163784	0.003034
Multimodal test function (scalable)	f_8	Ave	**− 12,569.48**	− 12,568.95	− 2.38E + 291	− 8873.12	− 371,812.74	− 12,568.89	− 2.41E + 61	− 7985.58
		Std	4.19E-03	4.69E-01	Inf	7.22E + 02	1.73E + 06	1.08E + 00	1.05E + 62	5.83E + 02
	f_9	Ave	**0**	**0**	**0**	**0**	7.52E-14	**0**	2.60E + 01	1.87E + 01
		Std	0	0	0	0	1.68E-13	0	6.88E + 01	1.26E + 01
	f_{10}	Ave	**8.88E-16**	**8.88E-16**	**8.88E-16**	9.37E-15	1.13E-08	**8.88E-16**	1.29E + 00	6.44E-10
		Std	0	0	0	2.85E-15	1.18E-08	0	5.00E + 00	4.16E-10
	f_{11}	Ave	**0**	**0**	**0**	2.39E-04	1.33E-16	**0**	1.13E-03	2.87E-17
		Std	0	0	0	1.33E-03	3.65E-16	0	6.29E-03	5.71E-17
	f_{12}	Ave	6.18E-04	4.37E-03	6.45E-04	4.47E-07	6.08E-01	9.36E-06	**5.02E-21**	3.42E-02
		Std	2.22E-03	7.72E-03	1.94E-03	3.01E-07	2.19E-01	1.69E-05	2.67E-20	1.80E-02
	f_{13}	Ave	2.28E-03	4.67E-03	2.43E + 00	3.77E-02	8.13E-03	8.57E-05	**3.46E-08**	8.01E-01
		Std	5.32E-03	5.30E-03	1.02E + 00	6.17E-02	1.64E-02	8.74E-05	1.93E-07	3.03E-01

Table 2 (continued)

	Function	Matric	AOSMA	SMA	MRFO	EO	SFO	HHO	SSA	WOA
Multimodal test function (fixed)	f_{14}	Ave	**0.9980**	**0.9980**	1.3810	**0.9980**	8.2415	1.3495	1.1029	**0.9980**
		Std	9.25E-14	2.00E-12	1.07E + 00	1.67E-16	4.47E + 00	9.38E-01	3.08E-01	1.27E-12
	f_{15}	Ave	4.82E-04	5.22E-04	8.00E-04	3.58E-03	3.52E-04	**4.27E-04**	1.67E-03	5.52E-04
		Std	2.08E-04	2.19E-04	4.44E-04	7.48E-03	4.73E-05	3.38E-04	6.20E-13	3.43E-04
	f_{16}	Ave	**− 1.0316**	**− 1.0316**	**− 1.0316**	**− 1.0316**	− 1.0314	**− 1.0316**	− 0.9777	**− 1.0316**
		Std	1.13E-10	1.09E-09	5.65E-16	6.13E-16	1.09E-03	1.61E-09	5.30E-02	1.69E-10
	f_{17}	Ave	**0.3979**	**0.3979**	**0.3979**	**0.3979**	0.4183	**0.3979**	**0.4309**	**0.3979**
		Std	1.44E-08	5.99E-08	0	0	7.40E-02	4.87E-05	2.88E-02	9.33E-09
	f_{18}	Ave	**3**	**3**	**3**	**3**	9.1154	**3**	9.2466	**3**
		Std	6.70E-09	4.55E-11	2.29E-15	1.45E-15	1.67E + 01	3.06E-07	7.18E + 00	1.13E-08
	f_{19}	Ave	− 3.8626	**− 3.8628**	**− 3.8628**	**− 3.8628**	− 3.8187	− 3.8584	− 3.8260	− 3.8585
		Std	5.91E-04	2.25E-07	2.60E-15	2.50E-15	5.50E-02	7.23E-03	2.59E-02	3.99E-03
	f_{20}	Ave	**− 3.2501**	− 3.2489	− 3.2798	− 3.2369	− 3.0258	− 3.0501	− 2.8131	− 2.9024
		Std	6.28E-02	5.91E-02	5.78E-02	9.81E-02	1.76E-01	1.51E-01	1.94E-01	4.48E-01
	f_{21}	Ave	**− 10.1532**	− 10.1529	− 7.8589	− 8.5964	− 4.9979	− 5.3586	− 8.4410	− 5.0552
		Std	2.39E-05	2.46E-04	2.80E + 00	2.51E + 00	8.62E-02	1.19E + 00	2.56E + 00	5.55E-06
	f_{22}	Ave	**− 10.4029**	− 10.4026	− 7.2417	− 8.3653	− 5.2304	− 5.0843	− 9.3781	− 5.0877
		Std	3.44E-05	2.18E-04	3.22E + 00	3.05E + 00	1.26E + 00	3.77E-03	2.13E + 00	1.09E-05
	f_{23}	Ave	**− 10.5364**	− 10.5360	− 7.7072	− 9.2773	− 5.0398	− 5.4431	− 8.6311	− 5.1285
		Std	2.69E-05	3.36E-04	3.02E + 00	2.65E + 00	1.09E-01	1.23E + 00	2.61E + 00	7.00E-06
Composition test function (CEC-2014)	CEC14 − F23	Ave	**2500**	**2500**	**2500**	2615.3290	2500.000001	**2500**	**2500**	**2500**
		Std	0	0	0	1.82E-01	8.88E-07	3.01E-12	1.39E-05	1.97E-08
	CEC14 − F24	Ave	**2600**	**2600**	**2600**	2600.0213	2600.0005	2600.0007	2600.0026	2600.2932
		Std	0	0	0	8.29E-03	2.75E-04	1.51E-03	1.03E-02	1.18E-01
	CEC14 − F25	Ave	**2700**	**2700**	**2700**	2701.3479	**2700**	**2700**	2705.4081	**2700**
		Std	0	0	0	4.19E + 00	1.09E-08	0	3.01E + 01	3.31E-10
	CEC14 − F26	Ave	**2707.14831**	2700.7350	2700.6366	2726.0904	2773.4668	2775.7707	2720.4649	2700.8131
		Std	2.48E + 01	1.16E-01	1.50E-01	4.43E + 01	4.22E + 01	4.18E + 01	1.43E + 01	1.67E-01
	CEC14 − F27	Ave	**2900**	**2900**	2923.258092	3290.7969	**2900**	**2900**	3369.1884	**2900**
		Std	0	0	129.4955738	7.62E + 01	1.26E-07	8.90E-12	5.90E + 02	3.05E-09
	CEC14 − F28	Ave	**3000**	**3000**	**3000**	3823.9213	**3000**	**3000**	4782.6991	**3000**
		Std	0	0	0	2.25E + 02	3.25E-07	1.57E-11	2.31E + 03	7.69E-09
Friedman's mean rank			**2.4310**	3.0517	3.5172	4.7414	5.9655	4.4310	5.8966	5.9655
Rank			**1**	2	3	5	7	4	6	8

Bold values represent better results

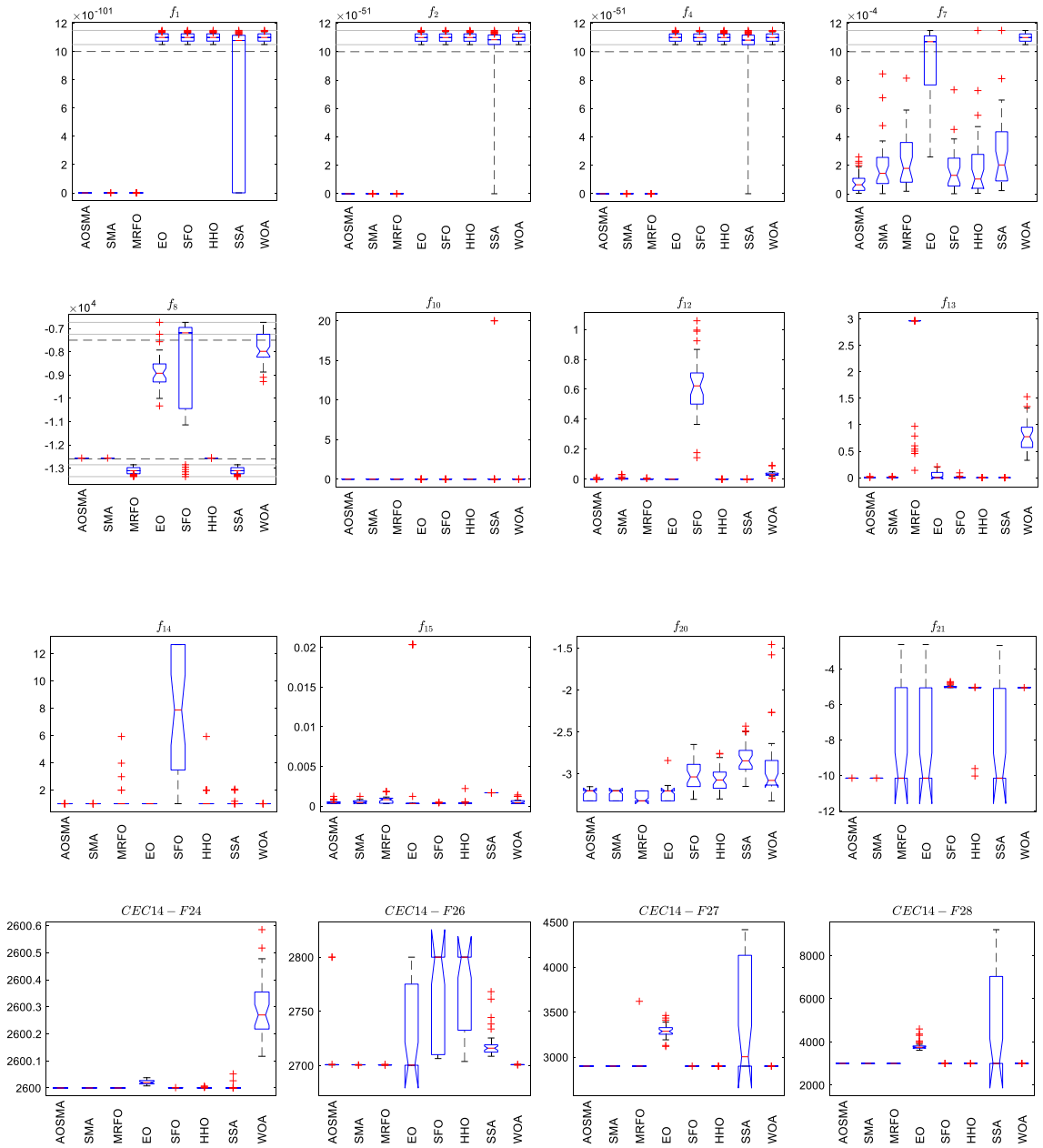

Fig. 3 Boxplot of several test benchmark functions

$(f_8 - f_{23})$, and composition test functions (CEC14 − F23 to CEC14 − F28). The results presented in Table 2 are evaluated for 31 independent runs to obtain the average value 'Ave' and standard deviation 'Std' for 30-dimensional test cases (except multimodal test functions with fixed dimensions $f_{14} - f_{23}$) with 15,000 maximum function evaluation.

The unimodal test functions $f_1 - f_7$ are designed to test the exploitation ability of the optimization algorithm. From the results presented in Table 2, AOSMA obtains the optimal minima for the test functions $f_1 - f_4$. The AOSMA has shown superior results for test functions f_7 compared to another optimization algorithm. For the test function f_6, AOSMA shows a significant improvement over its predecessors SMA and has shown superiority over SMA, SFO, HHO, and WOA. However, for the test function f_5, AOSMA results are like the SMA. However, the results are better than MRFO, EO, and WOA.

The multimodal test functions $(f_8 - f_{23})$ have a higher number of local optima, so these test functions are used to analyse the optimizer to verify how efficiently they explore the search space to reach the global minima. From the results presented in Table 2, it is noted that the AOSMA has shown superiority, among other optimization algorithms to obtain the optimal values. The AOSMA

Table 3 p-values with 5% significance for the test functions using Wilcoxon rank-sum test (p-values greater than 0.05 are shown in boldface)

Test functions	SMA	MRFO	EO	SFO	HHO	SSA	WOA
f_1	**5.00E − 01**	9.31E − 10	9.31E − 10	9.31E − 10	9.31E − 10	9.31E − 10	9.31E − 10
f_2	9.31E − 10	9.31E − 10	9.31E − 10	9.31E − 10	9.31E − 10	9.31E − 10	9.31E − 10
f_3	**1.00E + 00**	9.31E − 10	9.31E − 10	9.31E − 10	9.31E − 10	1.86E − 09	9.31E − 10
f_4	1.86E − 09	9.31E − 10	9.31E − 10	9.31E − 10	9.31E − 10	9.31E − 10	9.31E − 10
f_5	2.94E − 02	3.40E − 05	1.92E − 04	1.07E − 02	8.78E − 04	9.31E − 10	2.98E − 08
f_6	9.31E − 10	**1.50E − 01**	2.98E − 08	9.31E − 10	**7.20E − 01**	9.31E − 10	9.31E − 10
f_7	2.94E − 02	1.92E − 04	9.31E − 10	2.94E − 02	**7.08E − 02**	3.33E − 03	9.31E − 10
f_8	2.98E − 08	9.31E − 10	9.31E − 10	8.78E − 04	4.65E − 06	9.31E − 10	9.31E − 10
f_9	**1.00E + 00**	**1.00E + 00**	**1.00E + 00**	4.88E − 04	**1.00E + 00**	3.91E − 03	9.31E − 10
f_{10}	**1.00E + 00**	**1.00E + 00**	9.31E − 10	9.31E − 10	**1.00E + 00**	1.95E − 03	9.31E − 10
f_{11}	**1.00E + 00**	**1.00E + 00**	**1.00E + 00**	1.56E − 02	**1.00E + 00**	1.56E − 02	1.56E − 02
f_{12}	1.92E − 04	**1.00E + 00**	9.31E − 10	9.31E − 10	3.40E − 05	9.31E − 10	9.31E − 10
f_{13}	1.92E − 04	9.31E − 10	**2.81E − 01**	8.78E − 04	8.78E − 04	9.31E − 10	9.31E − 10
f_{14}	3.40E − 05	1.92E − 04	9.31E − 10	9.31E − 10	9.31E − 10	8.78E − 04	1.92E − 04
f_{15}	**2.81E − 01**	**1.50E − 01**	2.94E − 02	**1.50E − 01**	**2.81E − 01**	9.31E − 10	**1.00E + 00**
f_{16}	**7.08E − 02**	9.31E − 10	9.31E − 10	9.31E − 10	**1.00E + 00**	9.31E − 10	1.07E − 02
f_{17}	**1.00E + 00**	9.31E − 10	9.31E − 10	9.31E − 10	**1.50E − 01**	9.31E − 10	3.33E − 03
f_{18}	2.98E − 08	9.31E − 10	9.31E − 10	9.31E − 10	3.33E − 03	9.31E − 10	2.94E − 02
f_{19}	1.92E − 04	9.31E − 10	9.31E − 10	4.63E − 07	4.65E − 06	9.31E − 10	**7.20E − 01**
f_{20}	**2.81E − 01**	3.40E − 05	**7.08E − 02**	4.63E − 07	4.63E − 07	9.31E − 10	4.63E − 07
f_{21}	8.78E − 04	**4.73E − 01**	4.73E − 01	9.31E − 10	9.31E − 10	**7.08E − 02**	9.31E − 10
f_{22}	2.98E − 08	**1.00E + 00**	**1.50E − 01**	9.31E − 10	9.31E − 10	2.98E − 08	9.31E − 10
f_{23}	9.31E − 10	**1.00E + 00**	8.78E − 04	9.31E − 10	9.31E − 10	2.98E − 08	9.31E − 10
CEC14 − F23	**1.00E + 00**	**1.00E + 00**	9.31E − 10	9.31E − 10	3.91E − 03	3.13E − 02	9.31E − 10
CEC14 − F24	**1.00E + 00**	**1.00E + 00**	9.31E − 10	9.31E − 10	7.45E − 09	3.13E − 02	9.31E − 10
CEC14 − F25	**1.00E + 00**	**1.00E + 00**	9.31E − 10	9.31E − 10	**1.00E + 00**	4.88E − 04	9.31E − 10
CEC14 − F26	**7.20E − 01**	3.33E − 03	1.07E − 02	2.98E − 08	5.77E − 08	4.63E − 07	**4.73E − 01**
CEC14 − F27	**1.00E + 00**	**1.00E + 00**	9.31E − 10	9.31E − 10	2.44E − 04	3.81E − 06	9.31E − 10
CEC14 − F28	**1.00E + 00**	**1.00E + 00**	9.31E − 10	9.31E − 10	9.77E − 04	3.05E − 05	9.31E − 10
+	14 (48.3%)	15 (51.7%)	24 (82.8%)	28 (96.6%)	20 (69.0%)	28 (96.6%)	26 (89.7%)
≈	10 (34.5%)	11 (38.0%)	02 (06.9%)	0	05 (17.2%)	0	0
−	05 (17.2%)	03 (10.3%)	03 (10.3%)	01 (03.4%)	04 (13.8%)	01 (03.4%)	03 (10.3%)

outperformed other optimization algorithms on test functions f_8 and $f_{21} - f_{23}$ to obtain the global minimum. The AOSMA obtains the global minima for test functions f_9 and f_{11} and consistent optimal solution for test function f_{10} (as SMA, MRFO, EO, and HHO). However, for the test functions f_{12} and f_{13}, AOSMA has shown significant improvement over SMA, although the results are not the best among other optimizers. Moreover, the results of multimodal test functions with fixed dimensions $(f_{14} - f_{23})$ have shown similar results. However, AOSMA has shown superiority, among other optimization algorithms to achieve global optimal results in $f_{14}, f_{16} - f_{19}$, and $f_{21} - f_{23}$.

The compositions test functions from the IEEE CEC 2014 test suite are used to test the optimization algorithm that simulates to real search domain with many local minima. A comparative result of AOSMA with other optimization algorithm results is presented in Table 2 of six composition functions (CEC14 − F23 to CEC14 − F28). The AOSMA has shown similar results as MRFO, SMA, and WOA for all reported composition functions. However, AOSMA has shown some improvement over SFO, HHO, and SSA for composition functions like CEC14 − F26.

The result presented in Table 2 are obtained based on 31 independent runs, so for better understanding, a boxplot of four test functions from each unimodal, multimodal, multimodal with fixed dimensions and composition categories are presented in Fig. 3. From Fig. 3, one can visualize that

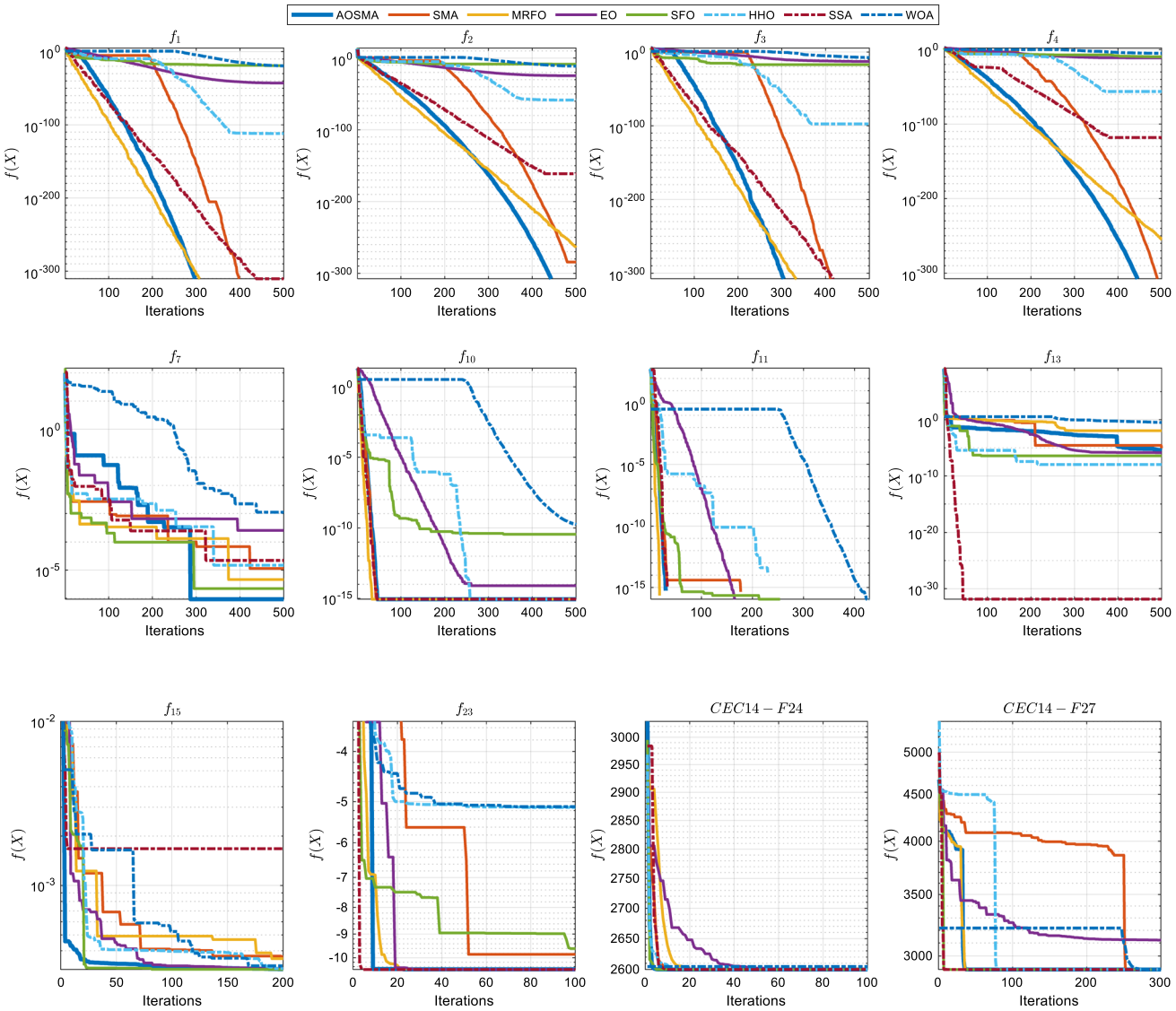

Fig. 4 Convergence curve

the AOSMA has shown superior consistency among various optimization algorithms.

Further, a statistical analysis of Friedman's mean rank test on the average value 'Ave' data presented in Table 2 is conducted, where the AOSMA ranked first among other optimization algorithms, as reported at the bottom of Table 2. In addition, a Wilcoxon signed-rank test is conducted to obtain p-value at $\alpha = 0.05$ and presented in Table 3. The Wilcoxon signed-rank test is used to find a substantial difference to obtain the fitness value by various methods based on p-value. A comparison of the p-value of AOSMA with another optimization algorithm is done using significantly better $(+)$, significantly equal $(\approx)$, and significantly poorer $(-)$. From Table 3, it is seen that the AOSMA has shown significantly better results. These are summarized as 48.3% over SMA, 51.7% over MRFO, 82.8% over EO, 96.6% over SFO, 69.0% over HHO, 96.6% over SSA, and 89.7% over WOA. From these data, it can be observed that the AOSMA has evolved as a better optimization algorithm.

3.4 AOSMA's comparative convergence analysis

This section presents a comparative convergence curve of AOSMA with other optimization algorithms for iterations 1 through 500. For the analysis, we have considered 5 from unimodal ($f_1 - f_4$ and f_7), 5 from multimodal ($f_{10}, f_{11}, f_{13}, f_{15}$ and f_{23}) and 2 from composition (CEC14 – F24 and CEC14 – F27) test functions. Figure 4 shows a comparative analysis of the convergence curve. Based on Fig. 4, we come up with some critical analysis as given below:

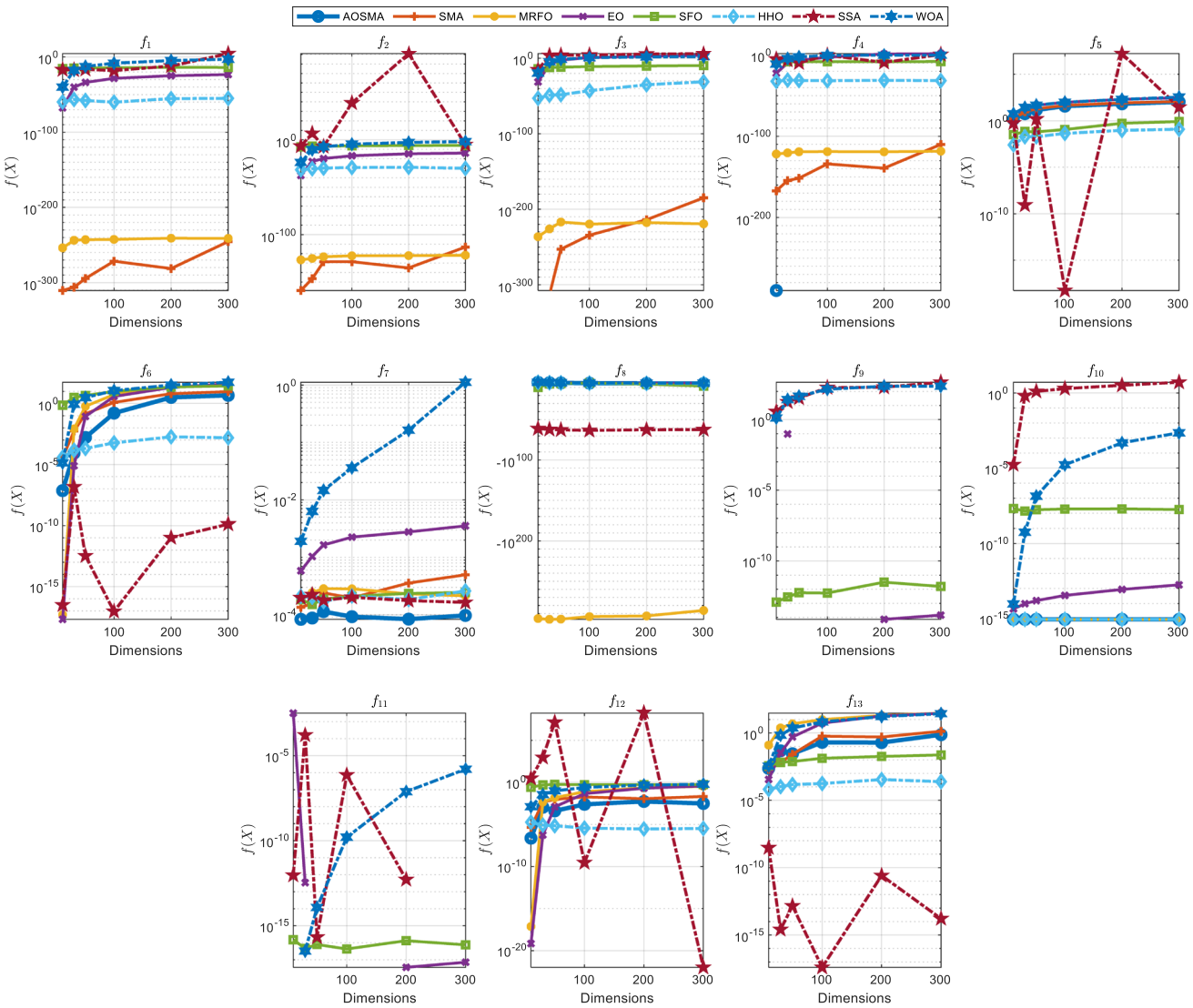

Fig. 5 Scalability analysis

- $f_1 - f_4$: At the beginning of the search process, the AOSMA has shown lagging behind MRFO, however, during the latter stage it shows superiority to obtain the optimal results.
- f_5: The AOSMA has shown a superiority convergence over another optimization algorithm.
- f_{10} and f_{11}: The AOSMA convergence is like the SMA, MRFO, and HHO to reach a globally optimal solution.
- f_{13}: The AOSMA convergence has shown little improvement over SMA, however, it still lags behind SSA, HHO, SFO, and EO.
- f_{15} and f_{23}: All the optimization algorithms are efficient in reaching the optimal solution, however, AOSMA has shown a better convergence.
- CEC14 – F24 and CEC14 – F27: The AOSMA convergence is the same as MRFO.

Based on these analyses, the AOSMA has shown an improved convergence due to the enhancement of the exploitation and exploration abilities.

3.5 AOSMA's comparative scalability analysis

In this section, a scalability analysis is performed to understand the impact of dimensions on the AOSMA performance. Its performances are compared with other optimization algorithms. For the experiment, the unimodal/multimodal test functions $f_1 - f_{13}$ are chosen with scalable dimensions $d = \{10, 30, 50, 100, 200, 300\}$. A comparative result based on the average fitness value 'Ave' on scalable dimensions for 31 independent runs with 15,000 maximum function evaluation is reported in Fig. 5. From Fig. 5, we can find in most of the cases, such as $f_1 - f_4, f_7$ and $f_9 - f_{11}$, that the AOSMA optimal results are

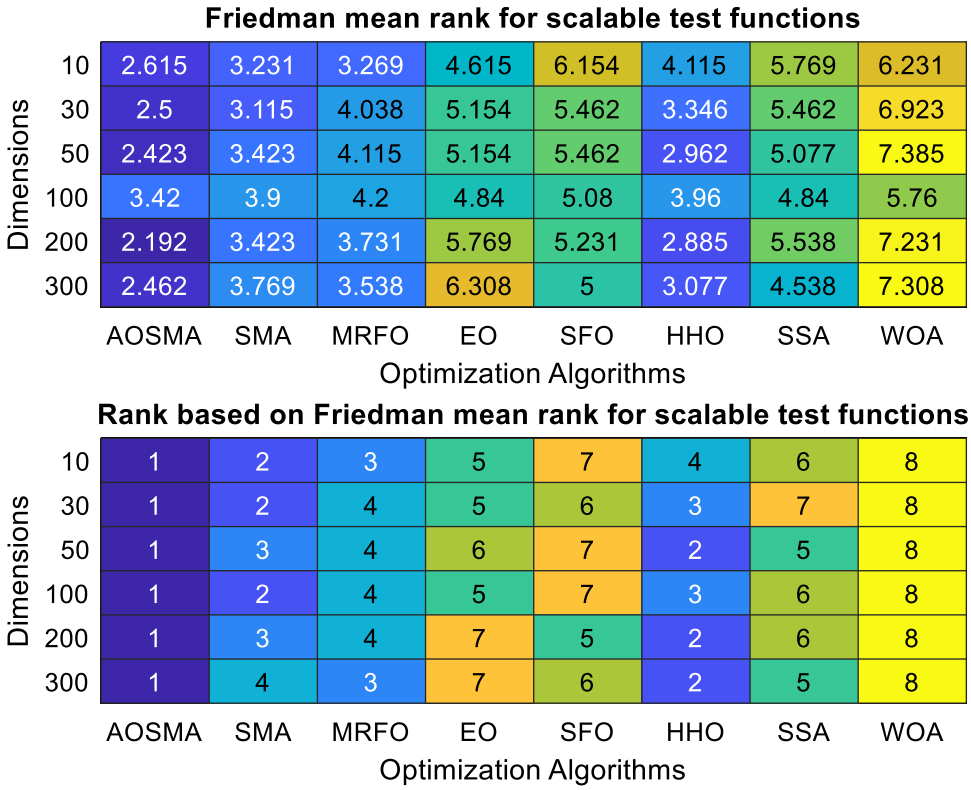

Fig. 6 Friedman mean rank score and ranking based on scalability results of test functions $f_1 - f_{13}$

independent of dimensionality changes. For the test function f_8, the scalable results are like SMA and HHO (we have not considered the SSA, MRFO, and SFO in this test case, as these algorithms produce a large deviation on results for the optimal value). However, for other test functions, the AOSMA has shown marginally degraded results on increasing dimensionality. For a better understanding of the overall performance of AOSMA on scalable test functions, a Friedman means rank test is conducted, based on the average fitness value obtained and reported in Fig. 6. Based on the results shown in Fig. 6, the AOSMA ranked one irrespective of dimensions.

4 Conclusion

This work presented an adaptive opposition slime mould algorithm (AOSMA) for function optimization. Nevertheless, the proposed algorithm exhibits better exploration and exploitation, because it adaptively decides whether to use the OBL or not. Need to mention here that the use of the position information from the opposition search space greatly enshrines the performances of the AOSMA. From Fig. 4, it is seen that the suggested AOSMA has shown better convergence than other state-of-the-art methods, because of its enhanced exploration and exploitation capabilities. Figure 5 shows the scalability feature of the proposed AOSMA. From the statistical analysis, it is observed that the performances are better than the recent methods. Moreover, the suggested AOSMA is more consistent over other optimizers, which is implicit in Fig. 3. From Table 3, it is observed that the AOSMA has shown explicitly better results. In summary, this algorithm uses only one random search agent, as opposed to the SMA, reducing 50 per cent chances of the misguidance of the exploration phase in certain instances. Furthermore, this algorithm inherently includes an adaptive mechanism to decide the use of the opposition-based learning on demand to delimit the exploration phase. To figure out, this is the reason behind the improved performances achieved. Finally, it is believed that the suggested algorithm would be useful for function optimization to solve real-world engineering problems.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Ahmadianfar I, Heidari AA, Gandomi AH et al (2021) RUN beyond the metaphor: an efficient optimization algorithm based on Runge Kutta method. Expert Syst Appl 181:115079. https://doi.org/10.1016/j.eswa.2021.115079

Dhargupta S, Ghosh M, Mirjalili S, Sarkar R (2020) Selective opposition based grey wolf optimization. Expert Syst Appl 151:113389. https://doi.org/10.1016/j.eswa.2020.113389

Dorigo M, Stützle T (2004) Ant colony optimization. MIT Press, Cambridge

Faramarzi A, Afshar M (2014) A novel hybrid cellular automata-linear programming approach for the optimal sizing of planar truss structures. Civ Eng Environ Syst 31:209–228. https://doi.org/10.1080/10286608.2013.820280

Faramarzi A, Heidarinejad M, Stephens B, Mirjalili S (2020) Equilibrium optimizer: a novel optimization algorithm. Knowl-Based Syst 191:105190. https://doi.org/10.1016/j.knosys.2019.105190

Feoktistov V (2006) Differential evolution. Springer, New York

Glover F (1989) Tabu search—part I. ORSA J Comput 1:190–206. https://doi.org/10.1287/ijoc.1.3.190

Glover F (1990) Tabu search—part II. ORSA J Comput 2:4–32. https://doi.org/10.1287/ijoc.2.1.4

Guha R, Ghosh M, Mutsuddi S et al (2020) Embedded chaotic whale survival algorithm for filter–wrapper feature selection. Soft Comput. https://doi.org/10.1007/s00500-020-05183-1

Heidari AA, Mirjalili S, Faris H et al (2019) Harris hawks optimization: algorithm and applications. Future Gener Comput Syst 97:849–872. https://doi.org/10.1016/j.future.2019.02.028

Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Michigan

Jain M, Singh V, Rani A (2019) A novel nature-inspired algorithm for optimization: Squirrel search algorithm. Swarm Evol Comput 44:148–175. https://doi.org/10.1016/j.swevo.2018.02.013

Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 8:687–697. https://doi.org/10.1016/j.asoc.2007.05.007

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Neural networks, 1995. Proceedings., IEEE International Conference on, vol 4. pp 1942–1948

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science (80-) 220:671. https://doi.org/10.1126/science.220.4598.671

Li S, Chen H, Wang M et al (2020) Slime mould algorithm: a new method for stochastic optimization. Future Gener Comput Syst. https://doi.org/10.1016/j.future.2020.03.055

Liang JJ, Qu BY, Suganthan PN (2013) Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization

Liu Y, Heidari AA, Ye X et al (2021) Boosting slime mould algorithm for parameter identification of photovoltaic models. Energy 234:121164. https://doi.org/10.1016/j.energy.2021.121164

Mahdavi S, Rahnamayan S, Deb K (2018) Opposition based learning: a literature review. Swarm Evol Comput 39:1–23. https://doi.org/10.1016/j.swevo.2017.09.010

Mirjalili S, Lewis A (2016) The Whale Optimization Algorithm. Adv Eng Softw 95:51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008

Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007

Naik MK, Panda R (2016) A novel adaptive cuckoo search algorithm for intrinsic discriminant analysis based face recognition. Appl Soft Comput 38:661–675. https://doi.org/10.1016/j.asoc.2015.10.039

Naik MK, Wunnava A, Jena B, Panda R (2020) 1. Nature-inspired optimization algorithm and benchmark functions: a literature survey. In: Bisht DCS, Ram M (eds) Computational intelligence, 3rd edn. De Gruyter, Berlin, Boston, pp 1–26

Qian S, Wu H, Xu G (2020) An improved particle swarm optimization with clone selection principle for dynamic economic emission dispatch. Soft Comput 24:15249–15271. https://doi.org/10.1007/s00500-020-04861-4

Rao RV, Savsani VJ, Vakharia DP (2012) Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems. Inf Sci (ny) 183:1–15. https://doi.org/10.1016/j.ins.2011.08.006

Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci (ny) 179:2232–2248. https://doi.org/10.1016/j.ins.2009.03.004

Shadravan S, Naji HR, Bardsiri VK (2019) The sailfish optimizer: a novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. Eng Appl Artif Intell 80:20–34. https://doi.org/10.1016/j.engappai.2019.01.001

Talbi E-G (2009) Metaheuristics. John Wiley & Sons, Inc., Hoboken, NJ, USA

Tizhoosh HR (2005) Opposition-based learning: a new scheme for machine intelligence. In: International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06). pp 695–701

Tu J, Chen H, Wang M, Gandomi AH (2021) The colony predation algorithm. J Bionic Eng 18:674–710. https://doi.org/10.1007/s42235-021-0050-y

Wang G-G (2018) Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Memetic Comput 10:151–164. https://doi.org/10.1007/s12293-016-0212-3

Wang G-G, Deb S, Cui Z (2019) Monarch Butterfly Optimization. Neural Comput Appl 31:1995–2014. https://doi.org/10.1007/s00521-015-1923-y

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1:67–82. https://doi.org/10.1109/4235.585893

Wunnava A, Kumar Naik M, Panda R et al (2020a) A differential evolutionary adaptive Harris hawks optimization for two dimensional practical Masi entropy-based multilevel image thresholding. J King Saud Univ - Comput Inf Sci. https://doi.org/10.1016/j.jksuci.2020.05.001

Wunnava A, Naik MK, Panda R et al (2020b) An adaptive Harris hawks optimization technique for two dimensional grey gradient based multilevel image thresholding. Appl Soft Comput 95:106526. https://doi.org/10.1016/j.asoc.2020.106526

Wunnava A, Naik MK, Panda R et al (2020c) A novel interdependence based multilevel thresholding technique using adaptive equilibrium optimizer. Eng Appl Artif Intell 94:103836. https://doi.org/10.1016/j.engappai.2020.103836

Yang Y, Chen H, Heidari AA, Gandomi AH (2021) Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts. Expert Syst Appl 177:114864. https://doi.org/10.1016/j.eswa.2021.114864

Yang X-S (2014) Cuckoo search and firefly algorithm: overview and analysis. In: Yang X-S (ed) Cuckoo search and firefly algorithm. Springer, New York, pp 1–26

Yao X, Yong L, Guangming L (1999) Evolutionary programming made faster. Evol Comput IEEE Trans 3:82–102. https://doi.org/10.1109/4235.771163

Yu C, Heidari AA, Xue X et al (2021) Boosting quantum rotation gate embedded slime mould algorithm. Expert Syst Appl 181:115082. https://doi.org/10.1016/j.eswa.2021.115082

Zhao W, Zhang Z, Wang L (2020) Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. Eng Appl Artif Intell 87:103300. https://doi.org/10.1016/j.engappai.2019.103300

Zhao S, Wang P, Heidari AA et al (2021) Multilevel threshold image segmentation with diffusion association slime mould algorithm and Renyi's entropy for chronic obstructive pulmonary disease. Comput Biol Med 134:104427

Zhu F, Chen D, Zou F (2020) A novel hybrid dynamic fireworks algorithm with particle swarm optimization. Soft Comput. https://doi.org/10.1007/s00500-020-05308-6