International Journal of Innovative
Computing, Information and Control
Volume **17**, Number **6**, December **2021**

ICIC International ©2021 ISSN 1349-4198
pp. 2151–2176

AN IMPROVED SLIME MOULD ALGORITHM BASED ON TENT CHAOTIC MAPPING AND NONLINEAR INERTIA WEIGHT

Yaning Xiao, Xue Sun*, Yapeng Zhang, Yanling Guo
Yangwei Wang and Jian Li

College of Mechanical and Electrical Engineering
Northeast Forestry University
No. 26, Hexing Road, Xiangfang District, Harbin 150040, P. R. China
*Corresponding author: xunsun@nefu.edu.cn
{ xiaoyaning; zhangyp812; wang.yangwei }@nefu.edu.cn; guo.yl@hotmail.com; lijian499@163.com

Received May 2021; revised September 2021

Abstract. *Slime mould algorithm (SMA) is a newly proposed swarm intelligence algorithm. Because of its strong robustness, simple structure, and few setting parameters, SMA has been worldwide concerned. However, the original SMA still has the problems of slow convergence rate, poor optimization precision and propensity to jam at the local optimum in several applications. To overcome these deficiencies, this paper proposes an improved slime mould algorithm (ISMA) to generate higher-quality optimal solutions. First, Tent chaotic mapping is adopted to replace the random distribution mode, which is considered from promoting the diversity of the initial population. Then, a novel pinhole imaging learning technique is developed to effectively explore the unknown domain and help the algorithm get rid of the local optimum. Furthermore, a nonlinear inertia weight that combines the inverse incomplete gamma function and beta distribution is embedded to modify the updated rules, so that the algorithm maintains a balance between the global exploration and local exploitation stage. Attributed to these improvements, the convergence rate and optimization precision of the original SMA are greatly enhanced. The validity of the proposed ISMA is evaluated on fifteen selected benchmark functions and two structural engineering design problems. The results demonstrate that ISMA shows superior performance and robustness compared against original SMA and most other advanced meta-heuristic algorithms.*

Keywords: Slime mould algorithm, Tent chaotic mapping, Pinhole imaging learning, Nonlinear inertia weight

1. **Introduction.** With the development of technology, increasingly complex constrained optimization problems have attracted more attention from many researchers in different disciplines and engineering fields around the world [1-3]. How to settle these problems is an indivisible research hotspot. However, traditional mathematical optimization methods are challenged by sub-optimal areas and the growing spatial dimension that cannot satisfy the demands of current problems [4]. As a result, during the past two decades, swarm intelligence (SI) algorithms have been rapidly developed to tackle complicated NP-hard problems [5].

SI algorithm is a branch of meta-heuristic algorithms, which searches for an optimal solution by simulating biological behaviour and physical phenomena observed in nature. Due to its simplicity, efficiency and gradient-free information, numerous SI algorithms have been available in recent years. Particle swarm optimization (PSO) [6] was first proposed in 1995 based on the behaviour of birds in foraging. Ant colony optimization (ACO)

DOI: 10.24507/ijicic.17.06.2151

[7] is designed to simulate the behaviour of ants looking for the optimal pathway between their colony and the food source. Grey wolf optimizer (GWO) [8] is a simulation of the hunting mechanism and social hierarchy in grey wolves. Whale optimization algorithm (WOA) [9] mimics a special behaviour of humpback whales referred to as social behaviour. Besides, Harris Hawks optimization (HHO) [10] is another SI algorithm, which is inspired by the cooperation behaviour of Harries' Hawks in hunting.

As a novel biologically-inspired algorithm, slime mould algorithm (SMA) was proposed by Li et al. [11] in 2020, which principally emulates the foraging behaviour and morphological variation of slime mould. It has the merits of few setting parameters, simple operation, and strong robustness. As a result, SMA has been successfully applied in chemistry, physics, as well as other different fields during recent years, and the related literature is as follows. Kumar et al. [12] used SMA to accurately estimate the parameters of solar photovoltaic (PV) cells, which significantly improved the performance of PV systems. Abdel-Basset et al. [13] proposed a neoteric classification model through integrating SMA and whale optimization algorithm for the effective image segmentation of chest X-ray images to determine whether a person is infected with the COVID-19 virus. Zubaidi et al. [14] combined SMA with the artificial neural network to predict urban water demand to help administrators effectively manage current water systems and arrange for expansions to meet increasing water demand. Chen and Liu [15] presented a prediction model based on SVR with hybrid SMA and K-means clustering method, which achieved the best prediction precision on six datasets.

As with other SI algorithms, there are also two important phases in the search process of SMA: global exploration and local exploitation [16-18]. In the exploration stage, the search agents explore the overall target area as far as possible to rapidly reach the promising area. While in the exploitation stage, the feasible area defined in the exploration stage is investigated to achieve the global best solution. Generally, it is crucial for an algorithm to balance the relationship between exploration and exploitation so as to better evade local minima while accelerating convergence. Even though SMA is a novel bio-inspired algorithm that has already shown competitive performance on global optimization problems, it still suffers from poor optimization precision, slow convergence rate, and the propensity to get trapped in a local optimum in some applications. These drawbacks are primarily associated with the poor quality of the initial population, lack of a proper balance between the exploration and exploitation phases, and low likelihood of large spatial leaps in the iteration process [19]. Therefore, many researchers have suggested a lot of improvements based on the original SMA from different aspects. In [20], Sun et al. proposed a modified version of SMA, named BTβSMA for numerical optimization. The adaptive β-hill climbing mechanism was integrated into BTβSMA to prevent the algorithm from falling into a local optimum, while Brownian motion was applied to initializing the population of slime mould to enhance the global exploration capability. Although the advantages of BTβSMA in terms of solution accuracy have been demonstrated, it neglects the complex runtime of the algorithm and cannot balance well the relationship between the exploration and exploitation stages, which leads to poor performance in multi-peaked functions. To avoid the basic SMA trapping into the local optimum, Nguyen et al. [21] adopted the reverse learning strategy to select the better individual to participate in subsequent iterations of the computation and then introduced the weight coefficient to update the slime mould position. In [22], Houssein et al. designed a hybrid algorithm that combines SMA with adaptive guided differential evolution algorithm (SMA-AGDE) to avoid premature convergence to a local minimum and enhance the population's diversity. Furthermore, [23] proposed a modified SMA (called ImSMA) through hiring the Lévy flight strategy to explore the search area effectively, which was considered from boosting the convergence

speed and improving the global search ability of the algorithm in the later phase. However, all the above improved strategies are mainly aimed at helping the algorithm get rid of the local optimum, and it still lacks a balanced approach between search abilities and also has room for further advancements in the solution accuracy and convergence speed.

The no free lunch (NFL) theorem indicates that there is no universal algorithm applicable to all optimization problems [24]. So, it is essential to modify existing optimization algorithms or design a new one to obtain better results in present or prospective optimization problems [25]. With the purpose of further enhancing the convergence speed and accuracy while minimizing the probability of trapping into the local optimal solution, an improved slime mould algorithm (ISMA) is designed to resolve global optimization problems in this study. First, Tent chaotic mapping is specifically dedicated to initializing the population to enable the slime mould individual to be uniformly distributed in the solution space. Because the chaotic mapping is unpredictable, regular and ergodic, it contributes to improving the population diversity and solution accuracy of the original SMA. In addition, a brilliant optimization mechanism known as pinhole imaging learning is proposed to boost the global exploration ability and help the algorithm jump out from the local optimal solution. Lastly, a nonlinear inertia weight is put forward to redefine the position updated rules of the algorithm for facilitating a balance between the exploration and exploitation capabilities while expediting convergence. Fifteen benchmark functions, including unimodal, multimodal, and fix-dimension multimodal functions are selected to comprehensively investigate the effectiveness and feasibility of the proposed ISMA. The comparison results indicate ISMA has superior performance on search precision and convergence rate in contrast to the original SMA and other state-of-the-art algorithms. In addition, ISMA is also applied to tackling the design of three-bar truss and the design of pressure vessel problems. The experimental results reveal that ISMA provides the best design among all competitors.

The remainder of this paper is arranged as follows. Section 2 briefly describes the original SMA. The proposed ISMA is explained in detail in Section 3. Section 4 provides the benchmark function tests and performance analysis of the proposed algorithm. In Section 5, ISMA is implemented to solve two practical application problems. Finally, the conclusions of this paper are given in Section 6.

2. **Slime Mould Algorithm.**

Slime mould algorithm [11] is a smart bio-inspired optimizer based on the spreading and foraging behaviour of slime mould in nature. The slime mould refers to Physarum polycephalum, a eukaryotic that lives in a humid and cold environment. The process of developing a slime mould can be categorized into three stages: the search for food, surrounding food, and digesting food. For the duration of the movement, slime mould extends through the front end to form a fan-like shape, trailed by a network of interconnected veins where the cytoplasm can flow, as illustrated in Figure 1. When slime mould gets closer to the food source, its biological oscillator generates a propagating wave to increase the cytoplasm via the vein. The increase in cytoplasmic flow leads to an expansion of the vein diameter. Conversely, as the flow decreases, the vein contracts as a result of the decrease in diameter. Based on such a combined negative-positive feedback mechanism, slime mould may create the best pathway to attach food. Even if the quality of food sources is different, slime mould is also capable of automatically establishing a reasonable route towards that with a higher concentration to meet the nutritional requirements of maximum concentration [26]. According to the characteristics of slime mould, the mathematical model of SMA is presented as follows.

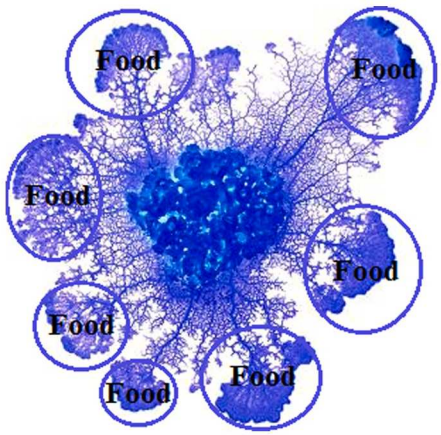


FIGURE 1. Foraging behaviour of slime mould [12]

In accordance with biological knowledge, slime mould can find food through its smell in the air, and its contraction pattern at this time is formulated as

$$\overrightarrow{X(t+1)} = \begin{cases} rand \cdot (U_b - L_b) + L_b, & rand < z \quad (1.1) \\ \overrightarrow{X_b(t)} + \overrightarrow{vb} \cdot \left(W \cdot \overrightarrow{X_A(t)} - \overrightarrow{X_B(t)} \right), & r_1 < p \quad (1.2) \\ \overrightarrow{vc} \cdot \overrightarrow{X(t)}, & r_1 \geq p \quad (1.3) \end{cases}$$

where $\overrightarrow{X(t)}$ is the current location of slime mould; $\overrightarrow{X_b(t)}$ stands for the optimal location found so far; $\overrightarrow{X_A(t)}$ and $\overrightarrow{X_B(t)}$ are two entities chosen from the population; z is a constant, and the original literature indicates that the algorithm performs best for most applications when $z = 0.03$; U_b and L_b are the upper and lower bounds of the search space for a certain problem; the value of $\overrightarrow{vc}$ is linearly decreasing from 1 to 0; $rand$ and r_1 are random numbers within the interval $[0, 1]$.

The value of $\overrightarrow{vb}$ oscillates randomly in the range $[-a, a]$ as

$$\overrightarrow{vb} = [-a, a] \quad (2)$$

$$a = \operatorname{arc\,tanh}\left(1 - \frac{t}{max\_iter}\right) \quad (3)$$

where t is the current iteration; max_iter represents the maximum iterations.

The mutation coefficient W mathematically emulates the oscillation frequency of slime mould at various food concentrations, which is beneficial for slime mould approaching the food faster when finding a high-quality food resource. At the same time, it inclines to move more slowly towards the food of poor quality. This screening method guarantees that the algorithm can converge to the optimal result with a good exploitation tendency and exploratory ability. And W is calculated by the following formula:

$$W(index(i)) = \begin{cases} 1 + r_1 \cdot \log\left(\dfrac{bestF - S(i)}{bestF - worstF} + 1\right), & condition \\ 1 - r_1 \cdot \log\left(\dfrac{bestF - S(i)}{bestF - worstF} + 1\right), & others \end{cases} \quad (4)$$

$$index = sort(S) \quad (5)$$

where $bestF$ and $worstF$ stand for the best fitness and worst fitness obtained so far respectively; $S(i)$ represents the fitness of $\overrightarrow{X(t)}$; $condition$ denotes the first half of populations in the $S(i)$ ranking; and $index$ represents the sequence of fitness values sorted.

As for the parameter p in Equation (1) it could be computed as follows:

$$p = \tanh|S(i) - DF| \tag{6}$$

where DF is the optimal fitness in all iterations.

Slime mould is mainly influenced by propagating waves produced from the biological oscillator, which regulate the flow of cytoplasm in the vein to make itself able to move to a better position of food concentration. Regardless of whether slime mould has found a favourable food origin, they would still extract several organisms to explore other regions in an attempt to discover another higher-quality food source. With the help of $\overrightarrow{vb}$, $\overrightarrow{vc}$ and W, the simulation of the variation of vein width is completed. Figure 2 illustrates the possible position under the different conditions in Equation (1). And the corresponding execution steps of the SMA algorithm are presented as follows:

Step 1. Initialize the population size N, the maximum number of iterations max_iter, and the dimension of search space D;

Step 2. Randomly generate the initial positions of the slime mould population X_i $(i = 1, 2, \ldots, N)$ according to Equation (7), and let $t = 0$ be the current number of iterations:

$$X_{N\times D} = rand(N, D) \times (U_b - L_b) + L_b \tag{7}$$

where U_b and L_b are the upper and lower boundaries of the search space, respectively;

Step 3. Calculate the fitness values $\{f(X_i), (i = 1, 2, \ldots, N)\}$ of each slime mould and sort them in ascending order to obtain $S(i)$. Also, record both the best fitness value and the worst fitness value;

Step 4. Calculate the mutation coefficient W by Equation (4);

Step 5. Update the current global optimal fitness and optimal solution, and use Equation (3) to renew the value of parameter a;

Step 6. Randomly generate a number $rand$ in the interval $[0, 1]$. If $rand < z$, update the position of slime mould individual by using Equation (1.1) and turn to **Step 9**; Otherwise, turn to **Step 7**;

Step 7. Update the values of the parameter p, vb and vc according to Equations (2), (3) and (6);

Step 8. Randomly generate a number r_1 in the interval $[0, 1]$. If $r_1 < p$, update the position of slime mould individual by using Equation (1.2), and if $r_1 \geq p$, update the current position according to Equation (1.3);

Step 9. Output the global optimal fitness and optimal solution if the maximum iterations number $(t = max\_iter)$ is achieved; if not, return to **Step 3**.

Algorithm 1 summarizes the pseudo-code of the basic SMA.

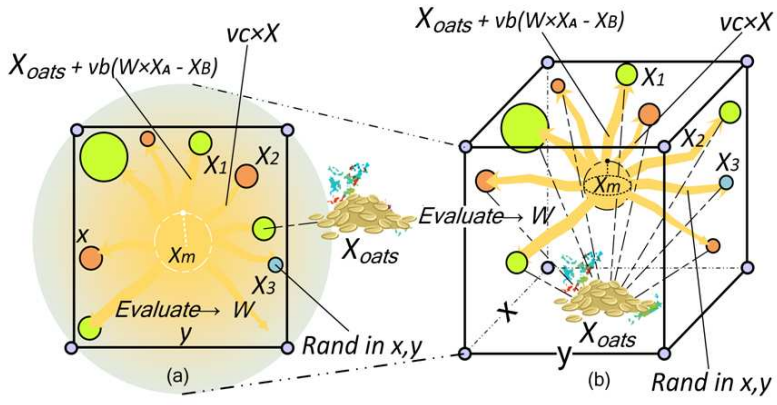


FIGURE 2. Possible positions in 2-dimension and 3-dimension [11]

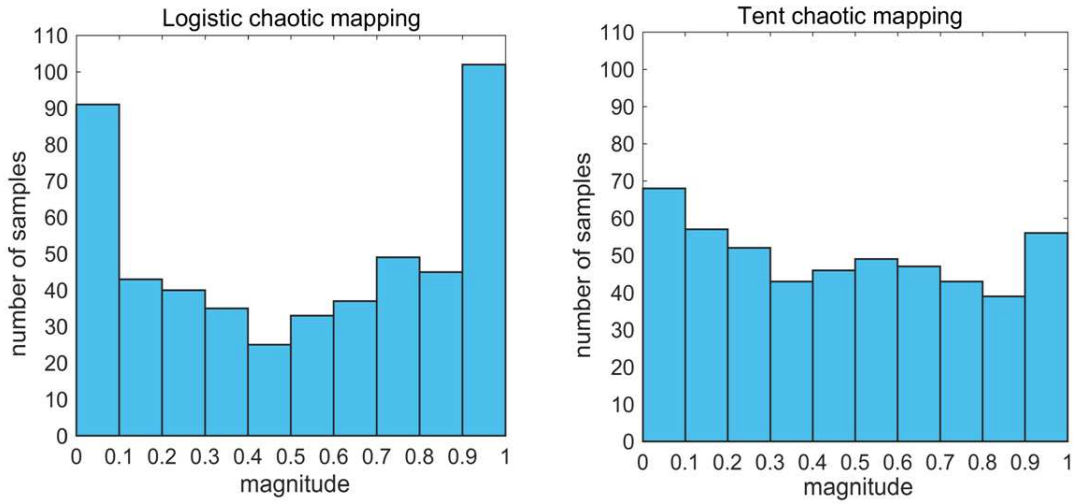
ALGORITHM 1. Slime mould algorithm

```
Set the population size N, the maximum iterations max_iter, and spatial dimension D
Initialize the positions of slime mould X_i (i = 1, 2, ..., N), and let t = 0
While (t ≤ max_iter)
    Compute the fitness value of each slime mould
    Compute W according to Equation (4)
    Update the global optimal fitness and optimal solution
    Update a according to Equation (3)
    For each slime mould do
        If rand < z
            Use Equation (1.1) to update the position
        Else
            Use Equations (2), (3) and (6) to update p, vb and vc
            If r_1 < p
                Use Equation (1.2) to update the position
            Else
                Use Equation (1.3) to update the position
            End If
        End If
    End For
    t++
End While
Output: Global optimal fitness and optimal solution
```

3. **Improved Slime Mould Algorithm.** Although the original slime mould algorithm has already achieved remarkable performance, some properties still have room for improvements to accomplish more powerful global search capability. Consequently, we offer a new variant of SMA, which integrates three improved strategies. More details are given in the consecutive subsections.

3.1. **Tent chaotic mapping.** It is indicated that abundant diversity of the initial population can considerably improve the performance of the algorithm both in terms of convergence rate and accuracy [27,28]. When the original SMA is used to resolve optimization problems, the population is generated randomly in the solution space depending on Equation (7), which cannot guarantee the distribution is uniform, thus leading to poor population diversity and decreasing the search efficiency at the later phase of the iteration process.

Chaotic mapping is a complex dynamic method found in nonlinear systems with the feature of unpredictability, regularity, and ergodicity [29,30]. Owe to its dynamic properties, the introduction of chaotic mapping for generating initial populations can not only promote population diversity but also enhance the overall exploration capability of the algorithm. Moreover, the chaotic mapping is greatly helpful to make the whole search process more rapidly, which provides a key contribution to speeding up the convergence. At present, Logistic chaotic mapping and Tent chaotic mapping are two commonly used chaos. To contrast the effects of Logistic chaotic mapping and Tent chaotic mapping, Figure 3 displays the distribution of chaotic sequences generated by two methods in the range $[0, 1]$ with the population size 500, respectively. It can be seen from the figure that the sequences generated by Logistic chaotic mapping are more likely to be in the interval $[0, 0.1]$ and $[0.9, 1]$ than the others, while those generated by Tent chaotic mapping are relatively uniform in the feasible domain and better than the former [31]. Therefore, Tent chaotic mapping is selected to replace the original random distribution mode to strengthen the performance of the algorithm in this paper. And its mathematical formulation is



(a) Logistic chaotic mapping (b) Tent chaotic mapping

FIGURE 3. Distribution of Logistic and Tent chaotic mapping

given by

$$X_{n+1} = \begin{cases} 2X_n, & 0 \le X_n < 0.5 \\ 2(1 - X_n), & 0.5 \le X_n \le 1 \end{cases} \tag{8}$$

Equation (8) after the Bernoulli shift transformation can be expressed as

$$X_{n+1} = (2X_n) \bmod 1 \tag{9}$$

For floating-point numbers in the interval $[0, 1]$, when the computer conducts the Tent chaotic mapping, it actually shifts the binary number of the fractional part unsigned to the left. However, due to the limited word length of the computer, there exists a small period $(0.2, 0.4, 0.6, 0.8)$ in the Tent chaotic mapping [32,33]. Once the generated X_n falls into the small period, it will converge to the invariant point 0 after a certain number of unsigned left shifts, thus compromising the randomness and ergodicity of the sequence. Hence, the steps to generate the Tent chaotic sequence in the feasible domain are as follows:

Step 1. Randomly generate the default value X_0 in the range $[0, 1]$ (pay attention to avoiding X_0 falling into a small period), expressed as the flag group S, $S(1) = X_0$, $n = j = 1$;

Step 2. Iterate according to Equation (9) and generate variables X_n, $n = n + 1$;

Step 3. When the maximal iteration number is reached, carry out **Step 5**. Otherwise, if the generated X_n falls into a small period, turn to **Step 4**. And if X_n does not appear above, turn to **Step 2**;

Step 4. Reset the default value by $X_n = S(j + 1) = S(j) + \varepsilon$, $j = j + 1$, and turn to **Step 2**;

Step 5. Terminate the program and save the generated sequence $X = \{X_0, X_1, \ldots, X_n\}$.

3.2. **Pinhole imaging learning.** In the original SMA, depending on the algorithm's position updated rules, a new candidate position of slime mould is available by directing the current individual to the global optimum $\left(\overrightarrow{X_b}\right)$. As the positions of the current individual continue to change, it may occur in the later iteration that all the remaining slime moulds in the population are clustered around the global best individual, thus leading to a decrease in the population diversity of the algorithm and even premature convergence to the local minimum [20].

Opposition-based learning (OBL) [34] as a robust optimization mechanism in the field of intelligence computation was proposed by Tizhoosh in 2005. Its main idea is to calculate and evaluate the current feasible solution as well as the corresponding inverse solution simultaneously, and then select the better one to proceed with the next generation. By introducing OBL to the meta-heuristic algorithms can effectively enlarge the search space and increase the possibility of obtaining a better solution close to the global optimum. Inspired by OBL strategy, this study designs a new technique called pinhole imaging learning (PIL) to update the current global optimal solution $\left(\overrightarrow{X_b}\right)$ dimension-by-dimension for the sake of boosting the exploration capability and helping the algorithm get rid of the local optimum. The PIL strategy overcomes the limitation that only the fixed reverse individual can be generated in OBL and incorporates a common optics phenomenon (i.e., pinhole imaging) in which the light source passes diagonally through one side of the pinhole plate to another side and its corresponding reverse image will be displayed on the screen at this time. Some concepts about PIL strategy are described as follows.

Definition 3.1. *(PIL point) Let the vector* $\overrightarrow{X_b} = (X_{b,1}, X_{b,2}, \ldots, X_{b,D})$ *be the global optimal solution in the current population, where* $X_{b,j}$ *is a point of* $\overrightarrow{X_b}$ *in the* j*th dimension with* $X_{b,j} \in [a_j, b_j]$, a_j *and* b_j *stand for the upper and lower limits of the search boundary in the* j*th dimension,* $j = 1, 2, \ldots, D$, *and* D *is the spatial dimension. Then the relative PIL point of* $X_{b,j}$ *can be defined as* $X^*_{b,j}$.

Suppose that in a certain space, $X_{b,j}$ (the jth dimension optimal point) is the projection on the x-axis of a light source p with height h. Besides, there is a pinhole plate placed at the base point O (the midpoint of the upper and lower boundaries of the search space a_j and b_j). Through the process of pinhole imaging, an inverted image p' of height h' can be obtained on the screen and its projection on the coordinate axis is $X^*_{b,j}$ (the PIL point), as shown in Figure 4(a). From the figure, the following equation can be obtained based on the laws of pinhole imaging and similar triangles:

$$\frac{(a_j + b_j)/2 - X_{b,j}}{X^*_{b,j} - (a_j + b_j)/2} = \frac{h}{h'} \tag{10}$$

Let $h/h' = n$, n is called the scaling factor. And the reverse point $X^*_{b,j}$ generated by the pinhole imaging learning operator can be calculated by transforming Equation (10):

$$X^*_{b,j} = \frac{(a_j + b_j)}{2} + \frac{(a_j + b_j)}{2n} - \frac{X_{b,j}}{n} \tag{11}$$

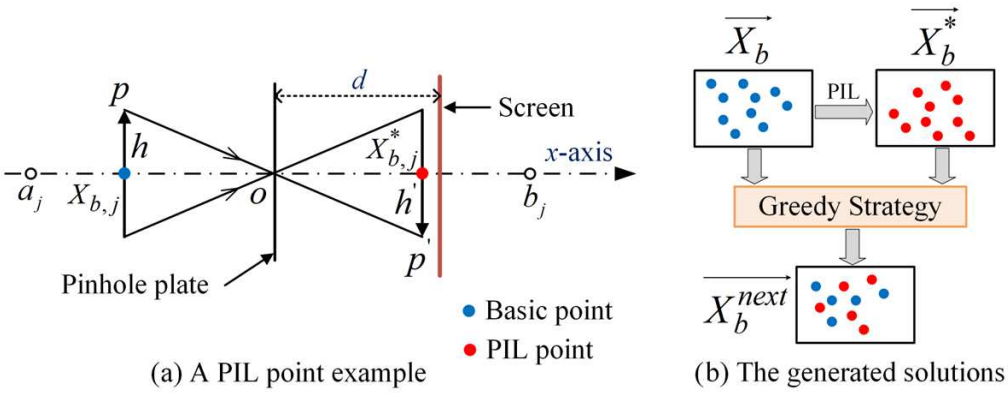


FIGURE 4. The principle of PIL strategy

Obviously, when $n = 1$, Equation (11) could be simplified to the formula of general OBL strategy acting upon $X_{b,j}$:

$$X^*_{b,j} = (a_j + b_j) - X_{b,j} \tag{12}$$

So, we could regard the opposition-based learning as a peculiar case of the pinhole imaging learning strategy. The reverse point obtained based on the OBL strategy is fixed, but by dynamically changing the scaling factor n, the proposed PIL is possible to achieve a better individual position thus enhancing the population diversity even further.

The dimension-by-dimension search model of the PIL strategy takes advantage of updated achievements of each dimension until all components of the D-dimensional space have been completed according to Equation (11), as shown in Figure 4(b). However, there is no guarantee that the newly generated PIL point $X^*_{b,j}$ is always better than the original optimal point $X_{b,j}$, so it is necessary to filter them via the greedy strategy [35], the implementation of which is described as follows:

$$X^{next}_{b,j} = \begin{cases} X^*_{b,j}, & f\left(X^*_{b,j}\right) < f(X_{b,j}) \\ X_{b,j}, & f\left(X^*_{b,j}\right) \geq f(X_{b,j}) \end{cases}, \quad j = 1, 2, \ldots, D \tag{13}$$

where $f(\cdot)$ represents the fitness function (for the minimum problems). If the newly generated point is better than the original point, i.e., $f\left(X^*_{b,j}\right) < f(X_{b,j})$, the updated result $X^*_{b,j}$ will be retained. Nevertheless, it will be discarded and conduct the update of the $(j + 1)$th dimension. Eventually, all the selected points $X^{next}_{b,j}$ will be consolidated into a new candidate solution $\overrightarrow{X^{next}_b}$ to participate in the subsequent optimization. The pseudo-code of PIL strategy is shown in Algorithm 2.

ALGORITHM 2. Pinhole imaging learning strategy

Input: Current global optimal solution $\overrightarrow{X_b} = (X_{b,1}, X_{b,2}, \ldots, X_{b,D})$, and scaling factor n

1. **For** ($j = 1$ to D) **do**
2. Generate the PIL point $X^*_{b,j}$ according to Equation (11)
3. Evaluate the fitness value of the PIL point
4. **If** $f\left(X^*_{b,j}\right) < f(X_{b,j})$ // Greedy strategy
5. $X^{next}_{b,j} = X^*_{b,j}$
6. **Else**
7. $X^{next}_{b,j} = X_{b,j}$
8. **End If**
9. **End For**

Output: New candidate solution $\overrightarrow{X^{next}_b} = \left(X^{next}_{b,1}, X^{next}_{b,2}, \ldots, X^{next}_{b,D}\right)$

3.3. **Nonlinear inertia weight.** Theoretically verified by Eberhart and Kennedy [6], the inertia weight ω is an essential factor that affects the exploration and exploitation abilities of PSO algorithm. If the inertia weight is $\omega > 1$, the algorithm will quickly diverge, and if the inertia factor $\omega < 0$, the algorithm will easily fall into a stagnant state. Many researchers have suggested that it is necessary to use a larger inertia weight in the early stage of search to enhance the exploration ability of the algorithm and obtain the promising region quickly. While in the later stage, it is desirable to change a smaller inertia weight for a better exploitation capability and improve the optimization accuracy [36]. If the relationship between exploration and exploitation is not well balanced, it will cause inefficient search and even premature convergence to a local optimum [37]. Usually, the inertia weight can be varied in a linear decreasing strategy or nonlinear attenuation,

but the latter allows the better algorithm diversity. Hence, in order to further speed up the convergence and foster an effective transition between the exploration and exploitation, this paper draws on the PSO algorithm, and designs a nonlinear inertia weight that combines the beta distribution and inverse incomplete gamma function to modify the position updating mechanism of slime mould, which is expressed as

$$\omega(t) = \omega_{\min} + \frac{\omega_{\max} - \omega_{\min}}{\lambda} \times gammaincinv(\lambda, a_0) + \sigma betarnd(b_1, b_2) \tag{14}$$

$$a_0 = 1 - \frac{t}{max\_iter} \tag{15}$$

where $\omega_{\min}$ and $\omega_{\max}$ are the maximum and minimum values of ω, respectively; $(1/\lambda)$ $gammaincinv(\lambda, a_0)$ denotes the inverse incomplete gamma function, $\lambda \geq 0$; σ is the control factor; $betarnd(b_1, b_2)$ denotes the random numbers that obey beta distribution, $b_1 > 0$, $b_2 > 0$; t is the current iteration; and max_iter is the maximum iteration.

Figure 5 shows the characteristics of the inverse incomplete gamma function. From the figure, it can be seen that inverse incomplete gamma function has the property of nearly linear decrease at the beginning of the iteration but exponential decrease at the later stage. Therefore, the first two terms of Equation (14) ensure that the inertia weight can be changed from $\omega_{\max}$ to $\omega_{\min}$ so as to facilitate a better balance between the exploration and exploitation stages.

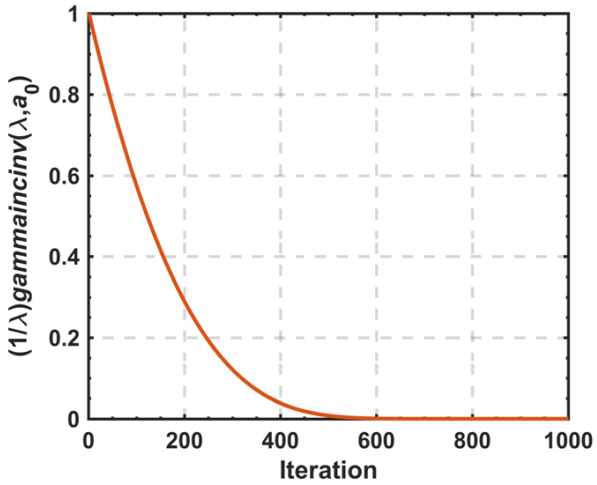


FIGURE 5. Characteristics of the inverse incomplete gamma function

In addition, the beta distribution as a set of continuous probability distributions defined in the interval $[0, 1]$, has the ability to approximate many forms of distribution, such as the common normal distribution, and uniform distribution [38]. The probability density function of beta distribution $f(x)$ is defined as

$$f(x; b_1, b_2) = \frac{1}{B(b_1, b_2)} x^{b_1 - 1}(1 - x)^{b_2 - 1}, \quad 0 < x < 1 \tag{16}$$

Figure 6 plots the selected densities of beta distribution with different parameters b_1 and b_2. When $b_1 = b_2$, the probability density of the beta distribution is symmetric at the line $x = 1/2$; when $b_1 < b_2$, the density is skewed to the left; when $b_1 > b_2$, the density is skewed to the right. It is noted that the uniform distribution is a special instance of the beta distribution $b_1 = b_2 = 1$. Since all particles approach towards the optimal solution in the later stages of the search, the population diversity also gradually decreases [39]. So, the third term of Equation (14) incorporates the beta distribution to accommodate the

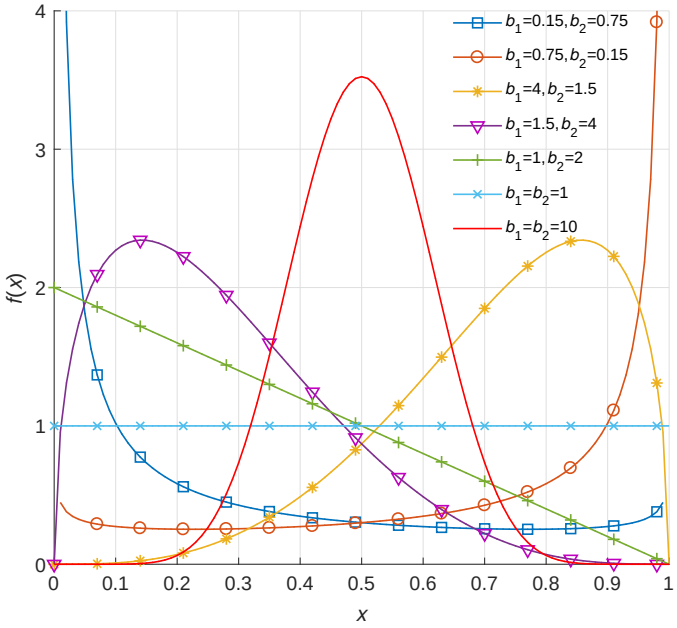


FIGURE 6. Image of the beta density distribution

value distribution of the inertia weight, and adopts the control factor σ before *betarnd* to rectify the deviation, thereby maintaining the stability of the algorithm and making the variation of the weight more reasonable.

Finally, the position updated method of SMA with the nonlinear inertia weight is represented as follows:

$$\overrightarrow{X(t+1)} = \begin{cases} rand \cdot (U_b - L_b) + L_b, & rand < z \quad (17.1) \\ \overrightarrow{X_b(t)} + \omega(t) \cdot \overrightarrow{vb} \cdot \left(W \cdot \overrightarrow{X_A(t)} - \overrightarrow{X_B(t)} \right), & r_1 < p \quad (17.2) \\ \omega(t) \cdot \overrightarrow{vc} \cdot \overrightarrow{X(t)}, & r_1 \geq p \quad (17.3) \end{cases}$$

3.4. **ISMA algorithm description.** Incorporating the improvements described in Subsections 3.1-3.3 above, the specific execution steps of the proposed improved slime mould algorithm in this paper are presented as follows:

Step 1. Initialize the population size N, the maximum number of iterations *max_iter*, and the dimension of search space D;

Step 2. Apply Tent chaotic mapping to generating the initial positions of the population X_i $(i = 1, 2, \ldots, N)$, and let $t = 0$ be the current number of iterations;

Step 3. Update the inertia weight ω according to Equation (14);

Step 4. Calculate the fitness values $\{f(X_i), (i = 1, 2, \ldots, N)\}$ of each slime mould and sort them in ascending order to obtain $S(i)$. Also, record both the best fitness value and the worst fitness value;

Step 5. Calculate the mutation coefficient W by Equation (4);

Step 6. Update the current global optimal fitness and optimal solution, and use Equation (3) to renew the value of parameter a;

Step 7. Randomly generate a number *rand* in the interval $[0, 1]$. If $rand < z$, update the position of slime mould individual by using Equation (17.1) and turn to **Step 10**; Otherwise, turn to **Step 8**;

Step 8. Update the values of the parameter p, vb and vc according to Equations (2), (3) and (6);

Step 9. Produce a random number r_1 in the interval $[0, 1]$. If $r_1 < p$, update the position of slime mould individual by using Equation (17.2), and if $r_1 \geq p$, update the current position according to Equation (17.3);

Step 10. Perform PIL strategy to update the global optimal solution dimension-by-dimension according to Algorithm 2;

Step 11. Output the global optimal fitness and optimal solution if the maximum iterations number ($t = max\_iter$) is achieved; if not, return to **Step 3**.

Algorithm 3 outlines the pseudo-code of ISMA.

ALGORITHM 3. Improved slime mould algorithm

Initialization:
1. Set the population size N, the maximum iterations max_iter, and spatial dimension D
2. Set $t = 0$, and Tent chaotic mapping is applied to generating the positions of slime mould X_i $(i = 1, 2, \ldots, N)$

Iteration:
3. **While** $(t \leq max\_iter)$
4. Use Equation (14) to update inertia weight ω
5. Compute the fitness of each slime mould
6. Compute W according to Equation (4)
7. Update the current global optimal fitness and optimal solution
8. Update a according to Equation (3)
9. **For** each slime mould in population **do**
10. **If** $rand < z$
11. Use Equation (17.1) to update the position
12. **Else**
13. Use Equations (2), (3) and (6) to update p, vb and vc
14. **If** $r_1 < p$
15. Use Equation (17.2) to update the position
16. **Else**
17. Use Equation (17.3) to update the position
18. **End If**
19. **End If**
20. Perform PIL strategy to update the global optimal solution
21. **End For**
22. t++
23. **End While**

Output: Global optimal fitness and optimal solution

4. **Numerical Experiment and Result Analysis.** To assess the feasibility and robustness of the proposed algorithm, several experiments are conducted in this section, covering the effectiveness analysis of three improved strategies and the performance comparison of ISMA with other state-of-the-art algorithms. All experiments were executed on MATLAB R2017a with Windows 10 system, and the hardware platform was configured as Intel (R) Core (TM) i5-7400 CPU @ 3.00GHz and 16GB RAM.

4.1. **Benchmark functions.** In this paper, 15 classical benchmark functions were selected from [26,40] to evaluate the global search capability of each algorithm. The expression, space dimension, definition domain, and theoretical optimum of these functions are listed in Table 1. The unimodal functions ($F_1 \sim F_7$) are desperately hard to converge to only one global minimum, so they are employed to investigate the convergence speed and solution precision of the algorithm. In comparison with unimodal functions, the multimodal functions ($F_8 \sim F_{12}$) incorporate multiple local optimums in the search space. Hence, they

TABLE 1. Benchmark functions

Function	Dim	Range	$F_{\min}$
$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, 10]$	0
$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	30	$[-100, 100]$	0
$F_4(x) = \max_i \{ \lvert x_i \rvert, 1 < i < n \}$	30	$[-100, 100]$	0
$F_5(x) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$	30	$[-30, 30]$	0
$F_6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	30	$[-100, 100]$	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1]$	30	$[-1.28, 1.28]$	0
$F_8(x) = \sum_{i=1}^{n} \left[x_i^2 - 10 \cos(2\pi x_i) + 10 \right]$	30	$[-5.12, 5.12]$	0
$F_9(x) = 20 - 20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i) \right) + e$	30	$[-32, 32]$	0
$F_{10}(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	30	$[-600, 600]$	0
$F_{11}(x) = \frac{\pi}{n} \left\{ 10 \sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_i + 1)] + (y_n + 1)^2 \right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a < x_i < a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	30	$[-50, 50]$	0
$F_{12}(x) = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_i + 1) \right] + (x_n - 1)^2 \times \left[1 + \sin^2(2\pi x_n) \right] \right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	$[-50, 50]$	0
$F_{13}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \left(j + \sum_{i=1}^{2} (x_i - a_{ij})^6 \right)^{-1} \right)^{-1}$	2	$[-65, 65]$	0.998
$F_{14}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1 (b^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]$	4	$[-5, 5]$	0.0003
$F_{15}(x) = - \sum_{i=1}^{4} c_i \exp\left(- \sum_{j=1}^{6} a_{ij} (x_j - p_{ij})^2 \right)$	6	$[0, 1]$	-3.32

are utilized to assess whether the algorithm has the potential to evade local optimum and seek the global best settlement. The fix-dimension multimodal functions ($F_{13} \sim F_{15}$) are a composition of the first two categories but with lower dimensions, which are used to measure if the relationships between the exploration and development phases have been perfectly balanced.

4.2. **Effectiveness analysis of improved strategies.** This paper presents Tent chaotic mapping, pinhole-imaging-based learning, and nonlinear inertia weight to overcome the defects of the original algorithm. So, this subsection is dedicated to validating the effectiveness of each strategy on the performance improvement of ISMA. Five algorithms, the original SMA, SMA with only Tent chaotic mapping (CSMA), SMA with only pinhole imaging learning (OSMA), SMA with only nonlinear inertia weight (WSMA), and the proposed ISMA were chosen for comparison experiments. For a fair comparison, the maximum iteration and population size were set to 1000 and 50, respectively. The other best parameters were set after a lot of tests and trials. In original SMA, $z = 0.03$; in CSMA, $z = 0.03$; in OSMA, $n = 12000$; in WSMA, $\omega_{\min} = 0.4$; $\omega_{\max} = 0.9$; $\lambda = 0.01$; $\sigma = 0.01$; $b_1 = 1$; $b_2 = 2$; in ISMA, $z = 0.03$; $n = 12000$; $\omega_{\min} = 0.4$; $\omega_{\max} = 0.9$; $\lambda = 0.01$; $\sigma = 0.01$; $b_1 = 1$; $b_2 = 2$. Besides, all algorithms were run independently 30 times in

TABLE 2. Comparison results of improved strategies

F		SMA	CSMA	OSMA	WSMA	ISMA
F_1	Avg	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Std	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_2	Avg	3.84E-82	6.47E-132	3.01E-139	5.82E-210	**4.94E-324**
	Std	2.10E-81	3.54E-131	1.65E-138	0.00E+00	**0.00E+00**
F_3	Avg	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Std	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_4	Avg	1.91E-72	1.82E-117	2.72E-140	1.03E-204	**3.49E-309**
	Std	7.01E-72	9.98E-117	1.34E-139	0.00E+00	**0.00E+00**
F_5	Avg	2.20E-01	2.09E-01	8.32E-02	3.67E-02	**1.83E-02**
	Std	5.01E-01	3.76E-01	1.03E-01	2.75E-02	**1.41E-02**
F_6	Avg	8.82E-03	1.88E-03	1.00E-03	1.12E-04	**3.85E-05**
	Std	1.13E-02	1.26E-03	7.50E-04	5.25E-05	**2.10E-05**
F_7	Avg	1.04E-04	7.32E-05	6.75E-05	3.14E-05	**3.10E-05**
	Std	9.02E-05	5.68E-05	4.69E-05	2.62E-05	**2.16E-05**
F_8	Avg	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Std	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_9	Avg	**8.88E-16**	**8.88E-16**	**8.88E-16**	**8.88E-16**	**8.88E-16**
	Std	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_{10}	Avg	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Std	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_{11}	Avg	9.47E-05	8.13E-05	5.79E-05	6.11E-05	**2.93E-05**
	Std	1.95E-04	1.73E-04	9.84E-05	5.78E-05	**2.81E-05**
F_{12}	Avg	7.20E-04	6.14E-04	1.74E-04	5.30E-05	**2.17E-05**
	Std	9.75E-04	5.55E-04	2.44E-04	2.98E-05	**9.93E-06**
F_{13}	Avg	**9.98E-01**	**9.98E-01**	**9.98E-01**	**9.98E-01**	**9.98E-01**
	Std	1.61E-14	9.95E-15	1.65E-14	9.54E-16	**6.33E-16**
F_{14}	Avg	4.33E-04	4.43E-04	**3.29E-04**	4.18E-04	3.32E-04
	Std	2.23E-04	2.60E-04	**5.49E-05**	1.79E-04	1.12E-04
F_{15}	Avg	−3.21E+00	−3.24E+00	−3.27E+00	−3.24E+00	**−3.27E+00**
	Std	3.63E-02	5.54E-02	6.03E-02	5.54E-02	**5.42E-02**

each benchmark function to reduce the random factors. Table 2 summarizes the average fitness (Avg) and standard deviation (Std) of each algorithm over 30 independent runs, where the best result obtained is highlighted in bold. The average fitness characterizes the convergence of the algorithm, and the better average fitness means the solution precision of the algorithm is higher. While the standard deviation presents the robustness of the algorithm, the lower the standard deviation indicates that the experimental results are much closer to the mean.

Investigating the data in Table 2, we found that improvements designed in the previous section have a synergistic influence on boosting the performance of the original SMA. Regardless of the average fitness and standard deviation, ISMA provides the best results on functions F_2, F_4, F_5, F_6, F_7, F_{11}, F_{12}, and F_{15}. For function F_{13}, all methods can obtain the same optimal value, but the standard deviation of ISMA is less than others. For function F_{14}, ISMA performs worse than OSMA but still ranks second among all algorithms. Five algorithms achieve the same results in the remaining functions. The experimental results reflect that the optimization precision and global search ability of the proposed algorithm are greatly enhanced to some extent, which is due to the rich population diversity support by Tent chaotic mapping.

Figure 7 visualizes the convergence curves of nine selected functions. It can also be seen that the convergence rate of ISMA is better than that of the original SMA, CSMA, OSMA and WSMA on unimodal functions F_1, F_3, F_5, and F_7. As well, the optimization precision is higher. For multimodal functions F_9 and F_{10}, the optimization precision of five algorithms is almost similar, while ISMA has a faster convergence rate from the beginning of the iteration. For functions F_{11} and F_{12}, the original SMA fails to find the global minimum. However, the improved ISMA algorithm is still capable of avoiding the

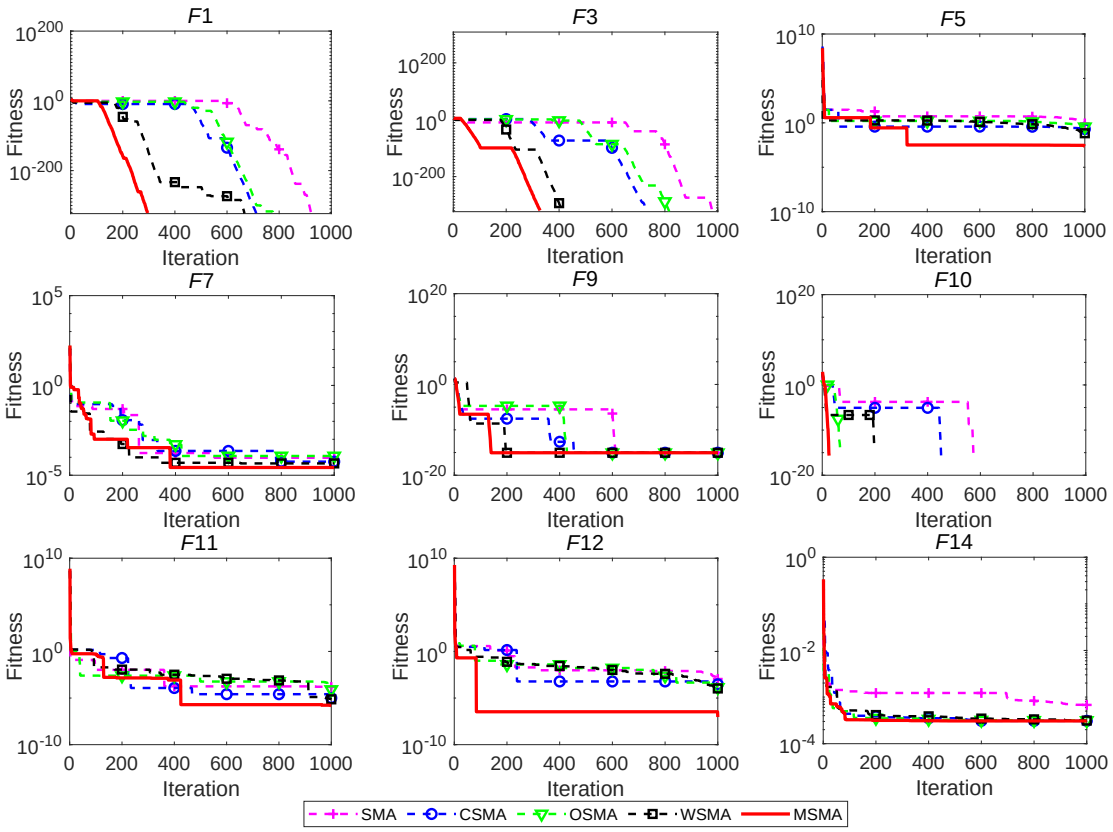


FIGURE 7. Convergence curves of improved strategies on nine benchmark functions

local optimum and achieving the global optimal value rapidly, which owes to pinhole imaging learning strategy. For fix-dimension multimodal function F_{14}, ISMA converges faster and the solution precision is higher than the original SMA.

Combined with the solution precision and stability performance, it is proved that the proposed ISMA is superior to the original SMA in optimization precision, convergence speed and the capability to evade the local optimum.

Furthermore, the mean absolute error (MAE) for five algorithms based on 15 benchmark functions is calculated, which is a valid statistical approach to display the difference between the theoretical value and obtained result, and its mathematical formula is as follows:

$$\mathrm{MAE} = \sum_{i=1}^{N_f} |m_i - o_i| / N_f \tag{18}$$

where m_i represents the mean of solution results, o_i denotes the theoretical optimum of each function, and N_f denotes the number of benchmark functions used.

It can be seen from Figure 8 that the MAE of ISMA is the minimum among all variant versions, which is reduced by 79.91% in comparison with the original algorithm. This once provides strong evidence to demonstrate the excellent performance of the proposed multi-strategy combination algorithm statistically.

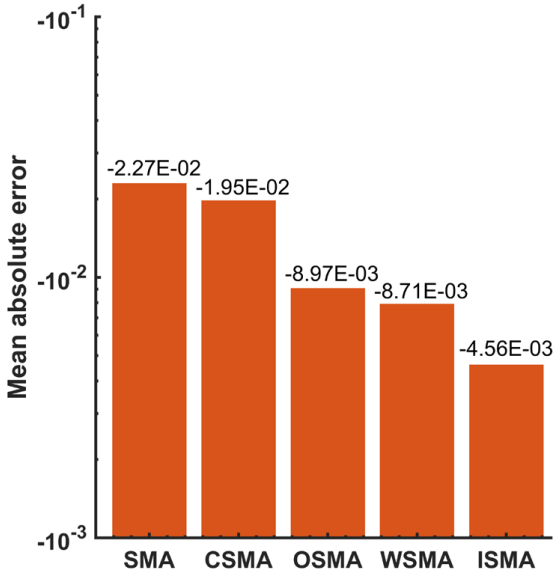


FIGURE 8. Comparison of MAE with improved strategies

4.3. **Comparison of ISMA with other meta-heuristic algorithms.** In order to comprehensively demonstrate the superiority of the proposed ISMA, it is compared against other six up-to-date meta-heuristic algorithms in this subsection, namely, particle swarm optimization (PSO) [6], sine and cosine algorithm (SCA) [41], grey wolf optimizer (GWO) [8], whale optimization algorithm (WOA) [9], multi-verse optimizer (MVO) [42], and modified PSO with an adaptive acceleration coefficient (TACPSO) [43]. Likewise, the maximum iterations and population size were set to 1000 and 50, respectively, and all the other parameters of algorithms were set the same as those recommended in the original literature, as shown in Table 3. Each algorithm was executed independently 30 times on benchmark functions. The average fitness and standard deviation obtained from the simulation experiments are outlined in Table 4, and the number in bold stands for the best value.

TABLE 3. Parameter settings

Algorithm	Parameter settings
PSO	$c_1 = 2$; $c_2 = 2$; $\omega_{\min} = 0.4$; $\omega_{\max} = 0.9$
SCA	$A = 2$
GWO	$a = [0, 2]$
WOA	$a_1 = [0, 2]$; $a_2 = [-2, -1]$; $b = 1$
MVO	Existence proability $\in$ [0.2 1]; Travelling distance rate $\in$ [0.6 1]
TACPSO	$c_1 = 2$; $c_2 = 2$; $\omega_{\min} = 0.4$; $\omega_{\max} = 0.9$
ISMA	$z = 0.03$; $n = 12000$; $\omega_{\min} = 0.4$; $\omega_{\max} = 0.9$; $\lambda = 0.01$; $\sigma = 0.01$; $b_1 = 1$; $b_2 = 2$

TABLE 4. Comparison results of ISMA with other algorithms

F		PSO	SCA	GWO	WOA	MVO	TACPSO	ISMA
F_1	Avg	2.02E-01	7.31E-03	1.93E-70	7.10E-172	1.71E-01	9.73E-07	**0.00E+00**
	Std	1.70E-01	1.98E-02	2.82E-70	8.54E-172	3.92E-02	3.79E-06	**0.00E+00**
F_2	Avg	6.98E-01	1.00E-05	4.68E-41	7.86E-110	2.92E-01	3.40E-01	**2.44E-285**
	Std	2.66E-01	1.75E-05	4.26E-41	2.78E-109	8.41E-02	1.83E+00	**0.00E+00**
F_3	Avg	5.99E+01	1.63E+03	2.26E-20	1.07E+04	1.94E+01	1.98E+02	**0.00E+00**
	Std	1.43E+01	2.46E+03	5.69E-20	6.11E+03	8.63E+00	9.10E+02	**0.00E+00**
F_4	Avg	1.39E+00	1.28E+01	1.41E-17	3.35E+01	7.02E-01	2.44E+00	**3.08E-312**
	Std	2.00E-01	1.05E+01	3.04E-17	3.03E+01	2.93E-01	1.38E+00	**0.00E+00**
F_5	Avg	2.18E+02	9.85E+01	2.67E+01	2.66E+01	5.76E+02	5.29E+01	**1.90E-02**
	Std	1.04E+02	1.48E+02	6.84E-01	3.04E-01	8.14E+02	3.79E+01	**1.52E-02**
F_6	Avg	1.63E-01	4.23E+00	4.05E-01	1.31E-02	1.83E-01	**1.19E-07**	3.72E-05
	Std	8.74E-02	3.03E-01	3.45E-01	4.63E-02	5.50E-02	**3.28E-07**	1.32E-05
F_7	Avg	1.62E+00	1.66E-02	3.73E-04	9.09E-04	1.29E-02	2.67E-02	**4.55E-05**
	Std	1.14E+00	1.24E-02	2.19E-04	9.61E-04	4.52E-03	1.06E-02	**3.37E-05**
F_8	Avg	9.55E+01	1.88E+01	4.66E-01	**0.00E+00**	1.09E+02	5.74E+01	**0.00E+00**
	Std	2.91E+01	2.80E+01	1.69E+00	**0.00E+00**	2.92E+01	1.27E+01	**0.00E+00**
F_9	Avg	6.42E-01	1.26E+01	1.45E-14	4.20E-15	1.10E+00	1.00E+00	**8.88E-16**
	Std	4.55E-01	9.50E+00	1.89E-15	2.27E-15	7.27E-01	8.74E-01	**0.00E+00**
F_{10}	Avg	1.95E-02	8.79E-02	1.93E-03	1.19E-03	4.28E-01	2.01E-02	**0.00E+00**
	Std	1.40E-02	1.59E-01	5.33E-03	6.51E-03	8.15E-02	2.04E-02	**0.00E+00**
F_{11}	Avg	1.59E-03	1.49E+03	2.32E-02	8.71E-04	9.57E-01	1.62E-01	**3.85E-05**
	Std	3.06E-03	8.18E+03	1.51E-02	1.76E-03	1.00E+00	1.91E-01	**3.46E-05**
F_{12}	Avg	5.60E-02	5.33E+02	2.74E-01	5.26E-02	3.30E-02	1.68E-02	**2.39E-05**
	Std	4.14E-02	2.90E+03	1.57E-01	9.29E-02	1.39E-02	3.81E-02	**1.35E-05**
F_{13}	Avg	2.09E+00	1.40E+00	3.29E+00	1.72E+00	**9.98E-01**	**9.98E-01**	**9.98E-01**
	Std	1.70E+00	8.07E-01	3.52E+00	1.89E+00	5.29E-12	9.22E-17	**2.67E-17**
F_{14}	Avg	7.99E-04	9.09E-04	3.68E-03	6.63E-04	5.17E-03	3.69E-04	**3.64E-04**
	Std	2.10E-04	4.16E-04	7.59E-03	3.97E-04	1.18E-02	2.32E-04	**6.10E-05**
F_{15}	Avg	−3.26E+00	−3.00E+00	−3.27E+00	−3.27E+00	−3.25E+00	−3.27E+00	**−3.28E+00**
	Std	6.03E-02	1.96E-01	6.73E-02	7.79E-02	5.85E-02	5.99E-02	**4.96E-02**

As seen from Table 4, on unimodal benchmark functions ($F_1 \sim F_7$), ISMA achieves all the best results in terms of the average fitness value and standard deviation except for function F_6. It is notable that ISMA presents theoretical optimum (0) on functions F_1 and F_3. As for function F_6, the performance of ISMA is worse than TACPSO but still ranks second. These results show that ISMA has outstanding search precision and local exploitation capability.

Similarly, on multimodal benchmark functions ($F_8 \sim F_{12}$), the average fitness and standard deviation of ISMA are obviously better than the rest of the algorithms except

for function F_8. Although ISMA and WOA obtain the same performance on function F_8, it is better than others. Since the multimodal benchmark functions contain many local minima, this means that ISMA can effectively bypass the local optimum and find the global best solution.

The fix-dimension multimodal functions ($F_{13} \sim F_{15}$) involve few local optima, which are designed to evaluate the stability of the algorithm in switching between global exploration and local exploitation processes. As far as average fitness is concerned, ISMA performs the same as MVO and TACPSO but better than others on function F_{13}. For functions F_{14} and F_{15}, only ISMA can generate results similar to the theoretically optimal value. In terms of standard deviation, ISMA is still the champion algorithm on all test functions.

Based on the experimental results from Table 4, a summary can be drawn that ISMA has a great ability to address function optimization problems.

The convergence curves of seven different algorithms for nine selected benchmark functions are illustrated in Figure 9. It is apparent that ISMA has a faster convergence speed and higher optimization precision compared with other competitors except for functions F_9 and F_{14}. For function F_9, the convergence precision of ISMA is similar to GWO as well as WOA, but ISMA converges faster at the beginning and achieves the global optimum with only a few number of iterations. For function F_{14}, even though ISMA has the same performance as TACPSO, it still outperforms all other algorithms. These show that the proposed algorithm could obtain the global optimum with a rapid convergence rate and high degree of robustness.

Although the above experimental results have proved that ISMA has a strong global search capability in low-dimensional functions, most current algorithms are highly prone to be ineffective in solving optimization problems as the number of spatial dimensions

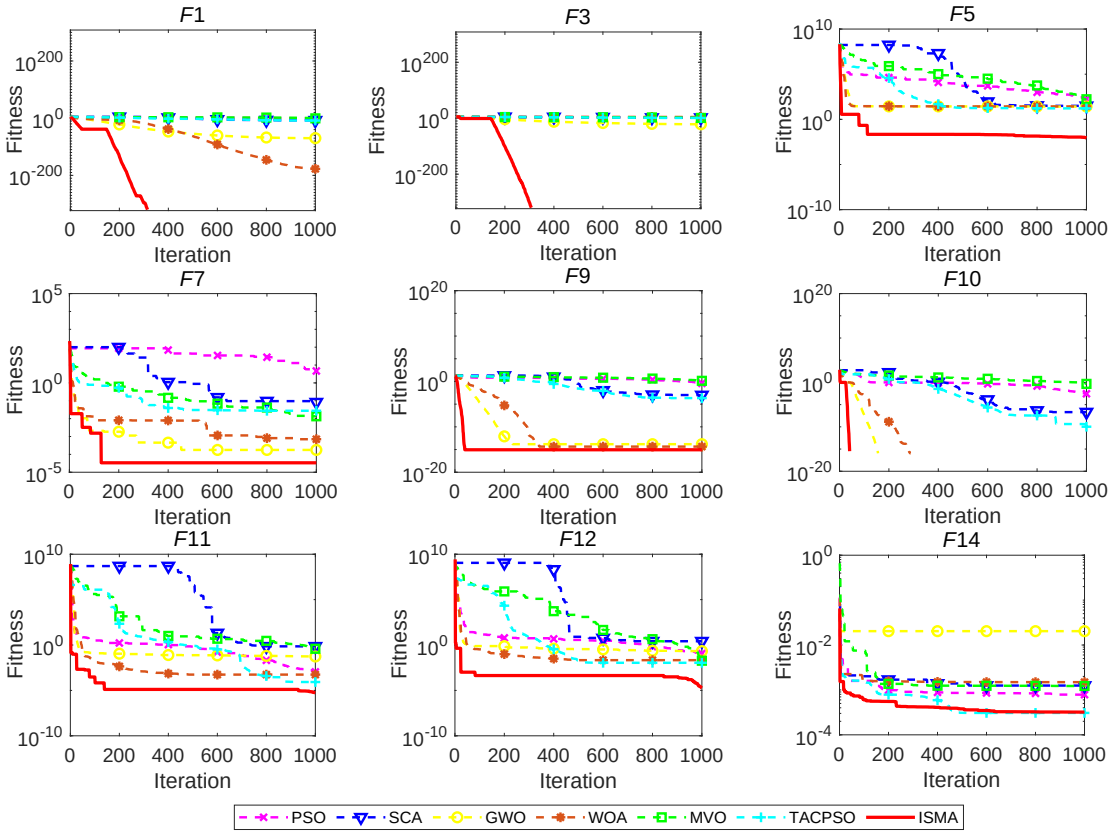


FIGURE 9. Convergence curves of seven algorithms on nine benchmark functions

increases. To investigate the scalability of ISMA, the proposed algorithm is used to optimize twelve benchmark functions $F_1 \sim F_{12}$ selected from Table 1 with higher dimensions (i.e., 50 and 100 dimensions). The average fitness and standard deviation values obtained by ISMA and other six algorithms are reported in Tables 5 and 6, respectively. Besides, the best result of each function has been highlighted in bold.

TABLE 5. Comparison results of ISMA with other algorithms on $F_1 \sim F_{12}$ (50D)

F-50D		PSO	SCA	GWO	WOA	MVO	TACPSO	ISMA
F_1	Avg	5.39E+00	2.73E+01	1.47E-51	2.85E-167	1.39E+00	7.43E-01	**0.00E+00**
	Std	1.81E+00	6.17E+01	2.41E-51	0.00E+00	2.89E-01	1.34E+00	**0.00E+00**
F_2	Avg	7.65E+00	3.18E-03	1.31E-30	1.10E-106	5.72E+01	6.60E+00	**0.00E+00**
	Std	1.73E+00	4.17E-03	1.11E-30	5.88E-106	7.82E+01	8.17E+00	**0.00E+00**
F_3	Avg	8.54E+02	2.75E+04	8.88E-09	8.70E+04	8.87E+02	5.38E+03	**0.00E+00**
	Std	2.35E+02	1.01E+04	3.68E-08	2.46E+04	2.33E+02	4.76E+03	**0.00E+00**
F_4	Avg	2.98E+00	5.28E+01	1.66E-11	5.72E+01	4.87E+00	1.77E+01	**0.00E+00**
	Std	2.84E-01	1.05E+01	1.56E-11	3.00E+01	2.78E+00	2.90E+00	**0.00E+00**
F_5	Avg	2.38E+03	2.60E+05	4.70E+01	4.70E+01	4.77E+02	3.32E+03	**2.04E-02**
	Std	9.47E+02	4.46E+05	7.16E-01	3.65E-01	7.36E+02	1.64E+04	**2.41E-02**
F_6	Avg	5.38E+00	3.14E+01	1.60E+00	6.17E-02	1.36E+00	3.05E-01	**1.44E-04**
	Std	2.09E+00	3.14E+01	5.03E-01	5.71E-02	2.81E-01	6.38E-01	**9.18E-05**
F_7	Avg	1.73E+02	6.74E-01	9.05E-04	1.40E-03	4.96E-02	1.22E-01	**2.21E-05**
	Std	6.88E+01	1.50E+00	4.25E-04	1.43E-03	1.60E-02	3.68E-02	**1.86E-05**
F_8	Avg	2.75E+02	6.44E+01	1.89E-15	3.79E-15	2.60E+02	1.37E+02	**0.00E+00**
	Std	3.63E+01	5.17E+01	1.04E-14	2.08E-14	3.94E+01	3.22E+01	**0.00E+00**
F_9	Avg	2.89E+00	1.48E+01	2.75E-14	3.97E-15	2.01E+00	3.23E+00	**8.88E-16**
	Std	4.15E-01	8.54E+00	3.20E-15	2.59E-15	5.50E-01	1.21E+00	**0.00E+00**
F_{10}	Avg	1.28E-01	1.27E+00	4.48E-04	4.22E-03	8.02E-01	1.69E-01	**0.00E+00**
	Std	4.56E-02	8.96E-01	2.45E-03	1.61E-02	5.95E-02	2.34E-01	**0.00E+00**
F_{11}	Avg	8.42E-02	6.22E+05	6.01E-02	2.70E-03	2.75E+00	2.44E+00	**4.86E-06**
	Std	6.76E-02	1.95E+06	1.84E-02	3.22E-03	1.15E+00	1.24E+00	**1.07E-05**
F_{12}	Avg	1.41E+00	1.15E+06	1.41E+00	2.00E-01	2.63E-01	1.26E+01	**2.95E-05**
	Std	5.33E-01	2.01E+06	2.75E-01	1.76E-01	1.07E-01	8.36E+00	**2.67E-05**

It can be seen from Table 5, when the problem dimension is set to 50, ISMA significantly outperforms other six algorithms in terms of the average fitness and standard deviation for all benchmark functions. It should be noted that ISMA discovers the theoretical optimal solution on functions F_1, F_2, F_3, F_4, F_8, and F_{10}. Moreover, in Table 6, when the problem dimension is set to 100, ISMA remains the champion among all competitors. However, the rest six algorithms are trapped into the local optimum to some extent, and the solution accuracy has a significant decline. Combining the solution results of the 50-dimensional and 100-dimensional problems, ISMA displays pretty strong scalability in the spatial dimension, that is to say, with the increase of the problem dimension, the overall performance of ISMA does not deteriorate significantly, and it is still able to provide high-quality solutions effectively and get rid of the local optimum.

5. **ISMA for Engineering Design Problems.** Lately, the resolution of structural design problems with stochastic optimizers has been widely used in the field of engineering design. Hence, to further validate the effectiveness of ISMA, the proposed algorithm is applied in this section to dealing with two mechanical design problems: the design of three-bar truss and the design of pressure vessel. For convenience, the death penalty method in [44] is implemented to address infeasible solutions based on the constraints.

TABLE 6. Comparison results of ISMA with other algorithms on $F_1 \sim F_{12}$ (100D)

F-100D		PSO	SCA	GWO	WOA	MVO	TACPSO	ISMA
F_1	Avg	8.56E+01	4.00E+03	1.26E-34	7.22E-167	2.33E+01	8.97E+02	**0.00E+00**
	Std	2.15E+01	2.98E+03	1.28E-34	0.00E+00	3.34E+00	4.26E+02	**0.00E+00**
F_2	Avg	1.05E+02	8.88E-01	6.73E-21	3.31E-108	1.34E+20	6.33E+01	**0.00E+00**
	Std	2.54E+01	2.31E+00	3.27E-21	9.16E-108	7.34E+20	2.35E+01	**0.00E+00**
F_3	Avg	1.12E+04	1.71E+05	5.84E-01	7.22E+05	3.18E+04	5.18E+04	**0.00E+00**
	Std	2.61E+03	3.88E+04	1.63E+00	1.12E+05	4.96E+03	1.84E+04	**0.00E+00**
F_4	Avg	8.33E+00	8.44E+01	1.10E-03	7.19E+01	4.40E+01	4.23E+01	**0.00E+00**
	Std	9.98E-01	3.86E+00	5.62E-03	2.78E+01	8.06E+00	4.79E+00	**0.00E+00**
F_5	Avg	8.05E+04	4.91E+07	9.73E+01	9.72E+01	1.11E+03	1.45E+05	**4.28E-02**
	Std	2.13E+04	2.53E+07	9.37E-01	4.16E-01	7.09E+02	1.31E+05	**8.39E-02**
F_6	Avg	8.54E+01	5.50E+03	7.98E+00	5.71E-01	2.22E+01	8.93E+02	**1.05E-03**
	Std	2.23E+01	3.31E+03	9.15E-01	2.10E-01	3.00E+00	4.85E+02	**1.02E-03**
F_7	Avg	1.26E+03	5.08E+01	1.66E-03	1.23E-03	2.41E-01	1.83E+00	**2.21E-05**
	Std	2.63E+02	2.53E+01	7.77E-04	1.41E-03	6.12E-02	1.72E+00	**1.88E-05**
F_8	Avg	9.80E+02	2.08E+02	3.94E-01	2.81E-13	5.68E+02	3.52E+02	**0.00E+00**
	Std	1.19E+02	1.28E+02	1.45E+00	1.23E-12	6.92E+01	4.63E+01	**0.00E+00**
F_9	Avg	5.26E+00	1.84E+01	6.98E-14	3.26E-15	4.53E+00	8.75E+00	**8.88E-16**
	Std	3.41E-01	5.04E+00	4.99E-15	2.35E-15	4.19E+00	1.01E+00	**0.00E+00**
F_{10}	Avg	7.36E-01	5.63E+01	1.25E-03	2.15E-03	1.21E+00	1.13E+01	**0.00E+00**
	Std	8.95E-02	4.27E+01	3.89E-03	4.03E-03	2.99E-02	8.67E+00	**0.00E+00**
F_{11}	Avg	3.71E+00	1.43E+08	1.82E-01	6.10E-03	1.03E+01	2.35E+01	**4.48E-06**
	Std	2.24E+00	7.45E+07	2.61E-02	2.56E-03	2.96E+00	1.88E+01	**6.53E-06**
F_{12}	Avg	8.77E+01	2.19E+08	5.42E+00	6.48E-01	8.93E+01	1.89E+04	**9.38E-05**
	Std	2.61E+01	1.24E+08	3.39E-01	2.91E-01	3.21E+01	8.39E+04	**1.10E-04**

All the mentioned algorithms were independently executed 30 times for each issue, with the maximum iterations and population size set to 500 and 30, respectively.

5.1. **Three-bar truss design problem.** The three-bar truss design problem is a typical optimization problem composed of three constraints. Figure 10 illustrates the three-bar truss and its three variables (A_1, A_2, and A_3). The ultimate target of this problem is to find out two optimal cross-sectional areas (A_1 and A_2) to achieve the minimum weight of the truss when the pressure condition is satisfied, and its mathematical model is as follows:

Consider

$$\overrightarrow{X} = [x_1 \ x_2] = [A_1 \ A_2]$$

Objective

$$f\left(\overrightarrow{X}\right)_{\min} = \left(2\sqrt{2}x_1 + x_2\right) \times L$$

Subject to

$$g_1\left(\overrightarrow{X}\right) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leq 0$$

$$g_2\left(\overrightarrow{X}\right) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leq 0$$

$$g_3\left(\overrightarrow{X}\right) = \frac{1}{\sqrt{2}x_2 + x_1}P - \sigma \leq 0$$

Variable ranges

$$0 \leq x_1, x_2 \leq 1$$

where $L = 100$ cm, $P = 2$ KN/cm^2, $\sigma = 2$ KN/cm^2.

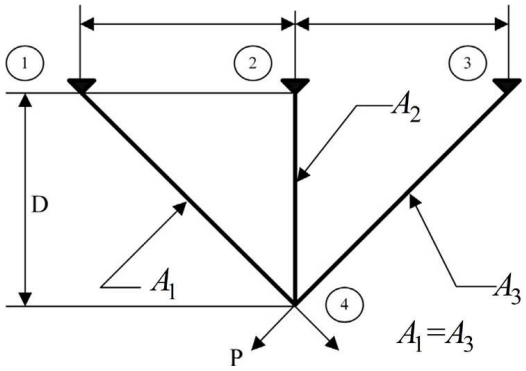


FIGURE 10. The design of three-bar truss problem

Table 7 lists the optimal results attained by ISMA and other five algorithms such as SMA [11], WOA [9], HHO [10], MTDE [45], and GDAFA [46]. From the table, we can detect that ISMA is the most efficient method in handling this problem among all competitors, and the minimum weight $f\left(\overrightarrow{X}\right)_{\min} = 263.8958$ is obtained at $\overrightarrow{X} = [0.7887\ 0.4083]$.

TABLE 7. Comparison results of three-bar truss problem

Algorithms	Optimal solution		Minimum weight
	$A_1(x_1)$	$A_2(x_2)$	
ISMA	0.7887	0.4083	**263.8958**
SMA	0.7882	0.4095	263.8959
WOA	0.8413	0.2767	265.6145
HHO	0.7861	0.4157	263.9009
MTDE	0.7877	0.4110	263.8977
GDAFA	0.7854	0.4177	263.9300

5.2. **Pressure vessel design problem.** The primary objective of this optimization problem is to discover four optimal parameters of cylindrical pressure vessel with minimum fabrication cost, namely the thickness of the shell (T_s), thickness of the head (T_h), inner radius (R), and length of the cylinder (L) as shown in Figure 11. The mathematical definition of the pressure vessel design problem is as follows:

Consider

$$\overrightarrow{z} = [z_1\ z_2\ z_3\ z_4] = [T_s\ T_h\ R\ L]$$

Objective

$$f\left(\overrightarrow{z}\right)_{\min} = 0.6224 z_1 z_3 z_4 + 1.778 z_2 z_3^2 + 3.1661 z_1^2 z_4 + 19.84 z_1^2 z_3$$

Subject to

$$g_1\left(\overrightarrow{z}\right) = -z_1 + 0.0193 z_3 \leq 0$$
$$g_2\left(\overrightarrow{z}\right) = -z_3 + 0.00954 z_3 \leq 0$$
$$g_3\left(\overrightarrow{z}\right) = -\pi z_3^2 z_4 - \frac{4}{3}\pi z_3^3 + 1296000 \leq 0$$
$$g_4\left(\overrightarrow{z}\right) = z_4 - 240 \leq 0$$

Variable ranges

$$0 \le z_1 \le 99, \quad 0 \le z_2 \le 99, \quad 10 \le z_3 \le 200, \quad 10 \le z_4 \le 200$$

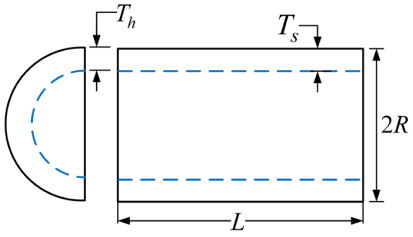


FIGURE 11. The design of pressure vessel problem

This case is resolved with ISMA and the results are compared with those revealed by SMA [11], MFO [47], GWO [8], MDDE [48], and CPSO [49]. The optimal solutions and corresponding minimum costs are tabulated in Table 8. Inspecting the data of this table, it is apparent that the proposed ISMA could outperform other well-known optimizers, and the lowest cost $f(\overrightarrow{z})_{\min} = 5974.6184$ is acquired when $\overrightarrow{z} = [0.8272\ 0.4089\ 42.8623\ 167.3954]$.

TABLE 8. Comparison results of pressure vessel design problem

Algorithms	Optimal solution				Minimum cost
	$T_s(z_1)$	$T_h(z_2)$	$R(z_3)$	$L(z_4)$	
ISMA	0.8272	0.4089	42.8623	167.3954	**5974.6184**
SMA	0.7931	0.3932	40.6711	196.2178	5994.1857
MFO	0.8125	0.4375	42.0984	176.6366	6059.7143
GWO	0.8125	0.4345	42.0892	176.7587	6051.5639
MDDE	0.8125	0.4375	42.0984	176.6360	6059.7017
CPSO	0.8125	0.4375	42.0912	176.7465	6061.0777

In short, these results provide strong evidence that ISMA is equally effective and feasible in practical engineering applications with significant progress.

6. **Conclusions.** In this paper, an improved slime mould algorithm (ISMA) was designed to overcome the shortcomings of slow convergence, poor optimization accuracy and the tendency to trap into local optimum in the original algorithm. First, Tent chaotic mapping was introduced to enrich the population diversity for slime mould. Then, the pinhole imaging learning technique was implemented to simultaneously evaluate the current feasible solution and corresponding inverse solution to increase the likelihood of obtaining the globally optimal result. Finally, a novel nonlinear inertia weight was formulated to ease the imbalance between exploration and exploitation stages of the algorithm and speed up the convergence. The performance of the proposed ISMA was tested on fifteen benchmark functions. Numerical experimental results demonstrate that ISMA with these improvements outperforms the original SMA and other six state-of-the-art meta-heuristic algorithms in solution precision, convergence speed, and the capability to evade the local optimum. In addition, to fully validate the utility of ISMA in real-world applications, we apply it to tackling two typical engineering design problems. The results reveal that ISMA provides the best solution among all competitors.

In future work, we intend to use the proposed ISMA to optimize parameters of the least-squares vector machine, and build a generalized model for shrinkage prediction of selective laser sintering (SLS) parts.

Acknowledgment. The authors are grateful to the editor and reviewers for their constructive comments and suggestions, which have improved the presentation, and this work is financially supported by the Fundamental Research Funds for the Central Universities No. 2572014BB06.

REFERENCES

[1] Q. Fan, H. Huang, K. Yang et al., A modified equilibrium optimizer using opposition-based learning and novel update rules, *Expert Systems with Applications*, vol.170, 2021.

[2] J. Islam, S. T. Meraj, B. M. Negash, K. Biswas, H. K. Alhitmi, N. I. Hossain, P. M. Vasant and J. Watada, Modified quantum particle swarm optimization for selective harmonic elimination (SHE) in a single-phase multilevel inverter, *International Journal of Innovative Computing, Information and Control*, vol.17, no.3, pp.959-971, 2021.

[3] R. Song, X. Zeng and R. Han, An improved multi-verse optimizer algorithm for multi-source allocation problem, *International Journal of Innovative Computing, Information and Control*, vol.16, no.6, pp.1845-1862, 2020.

[4] P. Hu, J.-S. Pan and S.-C. Chu, Improved binary grey wolf optimizer and its application for feature selection, *Knowledge-Based Systems*, vol.195, 2020.

[5] W. Romsai, A. Nawikavatan and D. Puangdownreong, Application of Levy-flight intensified current search to optimal PID controller design for active suspension system, *International Journal of Innovative Computing, Information and Control*, vol.17, no.2, pp.483-497, 2021.

[6] R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, *Proc. of the 6th International Symposium on Micro Machine and Human Science*, pp.39-43, 1995.

[7] M. Dorigo, M. Birattari and T. Stutzle, Ant colony optimization, *IEEE Computational Intelligence Magazine*, vol.1, no.4, pp.28-39, 2006.

[8] S. Mirjalili, S. M. Mirjalili and A. Lewis, Grey wolf optimizer, *Advances in Engineering Software*, vol.69, pp.46-61, 2014.

[9] S. Mirjalili and A. Lewis, The whale optimization algorithm, *Advances in Engineering Software*, vol.95, pp.51-67, 2016.

[10] A. A. Heidari, S. Mirjalili, H. Faris et al., Harris Hawks optimization: Algorithm and applications, *Future Generation Computer Systems*, vol.97, pp.849-872, 2019.

[11] S. Li, H. Chen, M. Wang et al., Slime mould algorithm: A new method for stochastic optimization, *Future Generation Computer Systems*, vol.111, pp.300-323, 2020.

[12] C. Kumar, T. D. Raj, M. Premkumar et al., A new stochastic slime mould optimization algorithm for the estimation of solar photovoltaic cell parameters, *Optik*, vol.223, 2020.

[13] M. Abdel-Basset, V. Chang and R. Mohamed, HSMA_WOA: A hybrid novel slime mould algorithm with whale optimization algorithm for tackling the image segmentation problem of chest X-ray images, *Applied Soft Computing*, vol.95, 2020.

[14] S. L. Zubaidi, I. H. Abdulkareem, K. S. Hashim et al., Hybridised artificial neural network model with slime mould algorithm: A novel methodology for prediction of urban stochastic water demand, *Water*, vol.12, no.10, 2020.

[15] Z. Chen and W. Liu, An efficient parameter adaptive support vector regression using K-means clustering and chaotic slime mould algorithm, *IEEE Access*, vol.8, pp.156851-156862, 2020.

[16] Z. Wang, H. Deng, X. Zhu and L. Hu, Application of improved whale optimization algorithm in multi-resource allocation, *International Journal of Innovative Computing, Information and Control*, vol.15, no.3, pp.1049-1066, 2019.

[17] R. E. Caraka, R. C. Chen, T. Toharudin, M. Tahmid, B. Pardamean and R. M. Putra, Evaluation performance of SVR genetic algorithm and hybrid PSO in rainfall forecasting, *ICIC Express Letters, Part B: Applications*, vol.11, no.7, pp.631-639, 2020.

[18] S. Suwannarongsri, Parallel cuckoo search for global optimization, *International Journal of Innovative Computing, Information and Control*, vol.17, no.3, pp.887-903, 2021.

[19] A. A. Ewees, L. Abualigah, D. Yousri et al., Improved slime mould algorithm based on firefly algorithm for feature selection: A case study on QSAR model, *Engineering with Computers*, 2021.

[20] K. Sun, H. Jia, Y. Li et al., Hybrid improved slime mould algorithm with adaptive β hill climbing for numerical optimization, *Journal of Intelligent & Fuzzy Systems*, vol.40, no.1, pp.1667-1679, 2021.

[21] T.-T. Nguyen, H.-J. Wang, T.-K. Dao et al., An improved slime mold algorithm and its application for optimal operation of cascade hydropower stations, *IEEE Access*, vol.8, pp.226754-226772, 2020.

[22] E. H. Houssein, M. A. Mahdy, M. J. Blondin et al., Hybrid slime mould algorithm with adaptive guided differential evolution algorithm for combinatorial and global optimization problems, *Expert Systems with Applications*, vol.174, 2021.
[23] A. A. El-Fergany, Parameters identification of PV model using improved slime mould optimizer and Lambert W-function, *Energy Reports*, vol.7, pp.875-887, 2021.
[24] Q. Fan, Z. Chen and Z. Xia, A modified salp swarm algorithm based on refracted opposition-based learning mechanism and adaptive control factor, *Journal of Harbin Institute of Technology*, vol.52, no.10, pp.183-191, 2020.
[25] C. Wang, L. Zhang and H. Zhu, An improved algorithm for instance segmentation lane detection of two branches, *ICIC Express Letters*, vol.14, no.12, pp.1169-1176, 2020.
[26] X. Zhang, K. Zhao and Y. Niu, Improved Harris Hawks optimization based on adaptive cooperative foraging and dispersed foraging strategies, *IEEE Access*, vol.8, pp.160297-160314, 2020.
[27] A. Deniz and H. E. Kiziloz, On initial population generation in feature subset selection, *Expert Systems with Applications*, vol.137, pp.11-21, 2019.
[28] H. Zhang, Y. Pan, J. Zhang, K. Dai and Y. Feng, Tent chaos and nonlinear convergence factor whale optimization algorithm, *International Journal of Innovative Computing, Information and Control*, vol.17, no.2, pp.687-700, 2021.
[29] C. A. Batista and R. L. Viana, Chaotic maps with nonlocal coupling: Lyapunov exponents, synchronization of chaos, and characterization of chimeras, *Chaos, Solitons & Fractals*, vol.131, 2020.
[30] M. Kaur, D. Singh, K. Sun et al., Color image encryption using non-dominated sorting genetic algorithm with local chaotic search based 5D chaotic map, *Future Generation Computer Systems*, vol.107, pp.333-350, 2020.
[31] D.-M. Zhang, Z.-Y. Chen, Z.-Y. Xin et al., Salp swarm algorithm based on craziness and adaptive, *Control and Decision*, vol.35, no.9, pp.2112-2120, 2020.
[32] N. Zhang, Z.-D. Zhao, X.-A. Bao et al., Gravitational search algorithm based on improved Tent chaos, *Control and Decision*, vol.35, no.4, pp.893-900, 2020.
[33] Y. Li, M. Han and Q. Guo, Modified whale optimization algorithm based on Tent chaotic mapping and its application in structural optimization, *KSCE Journal of Civil Engineering*, vol.24, no.12, pp.3703-3713, 2020.
[34] H. R. Tizhoosh, Opposition-based learning: A new scheme for machine intelligence, *International Conference on Computational Intelligence for Modelling, Control and Automation*, Vienna, Austria, pp.695-701, 2005.
[35] H. Dai, Y. Yang, J. Yin, H. Jiang and C. Li, Improved greedy strategy for cloud computing resources scheduling, *ICIC Express Letters*, vol.13, no.6, pp.499-504, 2019.
[36] H. Ding, Z. Wu and L. Zhao, Whale optimization algorithm based on nonlinear convergence factor and chaotic inertial weight, *Concurrency and Computation: Practice and Experience*, vol.32, no.24, 2020.
[37] J. S. D. Reddipogu and V. K. Elumalai, Hardware in the loop testing of adaptive inertia weight PSO-tuned LQR applied to vehicle suspension control, *Journal of Control Science and Engineering*, vol.2020, 2020.
[38] P. Maca and P. Pech, The inertia weight updating strategies in particle swarm optimisation based on the beta distribution, *Mathematical Problems in Engineering*, vol.2015, 2015.
[39] D.-M. Zhang, H. Xu, Y.-R. Wang et al., Whale optimization algorithm for embedded circle mapping and onedimensional oppositional learning based small hole imaging, *Control and Decision*, vol.36, no.5, pp.1173-1180, 2021.
[40] W. Long, J. Jiao, X. Liang et al., Pinhole-imaging-based learning butterfly optimization algorithm for global optimization and feature selection, *Applied Soft Computing*, vol.103, 2021.
[41] S. Mirjalili, SCA: A sine cosine algorithm for solving optimization problems, *Knowledge-Based Systems*, vol.96, pp.120-133, 2016.
[42] S. Mirjalili, S. M. Mirjalili and A. Hatamlou, Multi-verse optimizer: A nature-inspired algorithm for global optimization, *Neural Computing and Applications*, vol.27, no.2, pp.495-513, 2015.
[43] Z. Tang and D. Zhang, A modified particle swarm optimization with an adaptive acceleration coefficients, *2009 Asia-Pacific Conference on Information Processing*, pp.330-332, 2009.
[44] X.-S. Yang, *Optimization Techniques and Applications with Examples*, John Wiley & Sons, 2018.
[45] M. H. Nadimi-Shahraki, S. Taghian, S. Mirjalili et al., MTDE: An effective multi-trial vector-based differential evolution algorithm and its applications for engineering design problems, *Applied Soft Computing*, vol.97, 2020.

[46] J. Liu, Y. Mao, X. Liu et al., A dynamic adaptive firefly algorithm with globally orientation, *Mathematics and Computers in Simulation*, vol.174, pp.76-101, 2020.
[47] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, *Knowledge-Based Systems*, vol.89, pp.228-249, 2015.
[48] E. Mezura-Montes, C. Coello Coello, J. Velázquez-Reyes et al., Multiple trial vectors in differential evolution for engineering design, *Engineering Optimization*, vol.39, no.5, pp.567-589, 2007.
[49] Q. He and L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, *Engineering Applications of Artificial Intelligence*, vol.20, no.1, pp.89-99, 2007.

Author Biography

Yaning Xiao received the B.S. degree in mechanical and electrical engineering from Northeast Forestry University, China, 2020.

He is currently pursuing the M.S. degree in mechanical and electrical engineering at Northeast Forestry University. His research interests include mechanical & electrical integration, swarm intelligence algorithms, and signal processing. He has published 6 papers in journals so far.

Xue Sun received her B.S. degree in mechanical manufacturing and automation from Harbin Institute of Electrical Technology (now Harbin University of Science and Technology), China, 1991; the M.S. degree in mechanical manufacturing and automation from Harbin Engineering University, China, 2006.

Prof. Sun is currently a full-time professor at the College of Mechanical and Electrical Engineering, Northeast Forestry University, China. Her main research interests include robotics, artificial intelligence, and industrial process control. She has published over 20 papers in journals and conferences.

Yapeng Zhang received his B.S. degree automation from Shenyang University of Technology, China, 2020.

He is currently pursuing the M.S. degree in control theory and engineering at Northeast Forestry University. His research interests include intelligent control and industrial process control.

Yanling Guo received her B.S. degree in mechanical manufacturing and automation from Dalian University of Technology, China, 1984; the M.S. degree in mechanical manufacturing and automation from Harbin Institute of Technology, China, 1990; the Ph.D. degree in mechanical manufacturing and automation from Harbin Institute of Technology, China, 2001.

Prof. Guo is currently a full-time professor at the College of Mechanical and Electrical Engineering, Northeast Forestry University, China. Her main research interests include mechanical & electrical integration, additive manufacturing, and computer control system. She has published over 180 papers on well-known journals.

Yangwei Wang received his B.S. degree in mechanical manufacturing and automation from Nanjing University of Aeronautics and Astronautics, China, 2002; the M.S. degree in mechanical manufacturing and automation from Harbin Institute of Technology, China, 2007; the Ph.D. degree in mechanical manufacturing and automation from Harbin Institute of Technology, China, 2011.

Prof. Wang is currently a full-time professor at the College of Mechanical and Electrical Engineering, Northeast Forestry University, China. His main research interests include robotics, artificial intelligence, and smart materials. He has published over 30 papers on well-known journals. He is a Review Expert of the *Journal of Bionic Engineering*.

Jian Li received his B.S. degree in mechanical manufacturing and automation from Shandong University, China, 2005; the M.S. degree in mechanical manufacturing and automation from Harbin Institute of Technology, China, 2007; the Ph.D. degree in mechanical manufacturing and automation from Harbin Institute of Technology, China, 2011.

Prof. Li is currently a full-time professor at the College of Mechanical and Electrical Engineering, Northeast Forestry University, China. His main research interests include robotics, smart materials, and additive manufacturing. He has published over 30 papers on well-known journals. He is a Review Expert of the *Engineering Applications of Computational Fluid Mechanics*.