Mathematical Modelling and Numerical Simulation with Applications, 2025, 5(1), 143–171

https://dergipark.org.tr/en/pub/mmnsa
ISSN Online: 2791-8564 / Open Access
https://doi.org/10.53391/mmnsa.1571964

RESEARCH PAPER

Effect of chaos on the performance of spider wasp meta-heuristic optimization algorithm for high-dimensional optimization problems

Haneche Nabil [1,*,‡] and **Hamaizia Tayeb** [2,‡]

[1]Applied Mathematics & Modeling Laboratory, Department of Mathematics, University of Mentouri Brothers, 25000 Constantine, Algeria, [2]Mathematical Modeling & Simulation Laboratory, Department of Mathematics, University of Mentouri Brothers, 25000 Constantine, Algeria

*Corresponding Author
‡nabil.haneche@doc.umc.edu.dz (Haneche Nabil); el.tayyeb@umc.edu.dz (Hamaizia Tayeb)

Abstract

The spider wasp optimization (SWO) algorithm is a new nature-inspired meta-heuristic optimization algorithm based on the hunting, nesting, and mating behaviors of female spider wasps. This paper aims to apply chaos theory to the steps of the SWO algorithm in order to increase its convergence speed. Four versions of chaotic algorithms are constructed using the traditional spider wasp optimizer. The proposed chaotic spider wasp optimization (CSWO) algorithms select various chaotic maps and adjust the main parameters of the SWO optimizer to ensure the balance between exploration and exploitation stages. Furthermore, the constructed CSWO algorithms are benchmarked on eight well-known test functions divided into unimodal and multimodal problems. The experimental results and statistical analysis are carried out to demonstrate that CSWO algorithms are very suitable for searching optimal solutions for the benchmark functions. Specifically, the implementation of chaotic maps can significantly enhance the performance of the SWO algorithm. As a result, the new algorithm has high flexibility and outstanding robustness, which we can apply to engineering design problems.

Keywords: Meta-heuristic optimizer; chaotic map; spider wasp algorithm; benchmark function

AMS 2020 Classification: 37N40; 68T20; 90C59; 92B20

1 Introduction

Optimization is a structured method used to determine decision variables while adhering to various constraints to either maximize or minimize the cost function. The constraints, cost function, and design variables are the fundamental components of every optimization problem. Optimization approaches have significant applications in engineering, image processing, wireless sensor

➤ **Received:** 22.10.2024 ➤ **Revised:** 17.02.2025 ➤ **Accepted:** 12.03.2025 ➤ **Published:** 31.03.2025

networks, and bioinformatics [1]. A number of real-world problems exhibit high non-convexity and non-linearity, typically due to the presence of several design variables and fundamental constraints. Furthermore, there is no guarantee of achieving the optimal global solution. These practical problems present challenges that motivate scientists to develop new and effective methods for better results. Optimization methods may be classified into two fundamental categories: gradient-based deterministic methods and stochastic non-traditional methods [2]. Deterministic methods have constraints when solving problems that include search spaces that are discontinuous, nonconvex, high-dimensional, and have non-differentiable objective functions. In contrast, stochastic-based algorithms do not depend on gradient-based information. Instead, they rely on stochastic methods inside the search space to reduce these constraints. Meta-heuristic algorithms (MAs) are extensively used in stochastic-based methods due to their wide use over different techniques. Meta-heuristic algorithms provide a significant capacity for fully examining the solution space and effectively adopting what is the optimal solution. As a result, in recent years, many researchers have worked to introduce novel meta-heuristic algorithms and enhance the performance of current methods [3].

Recently, several nature-inspired meta-heuristic algorithms have been developed. These artificial algorithms imitate the behaviors of existing species or natural phenomena. Thus, these algorithms have been proposed and used as effective approaches for solving several optimization problems [4]. However, these MAs often demonstrate higher levels of sensitivity when they involve adjusting user-defined parameters. MAs may not precisely attain the global optimal solution, which is another disadvantage [5]. There are two categories of MAs: single solution-based and population-based [6]. The single-solution approach to optimization includes evaluating one solution. In contrast, with the population-based method, solutions are generated during each optimization step. Population-based meta-heuristic algorithms start the optimization process by generating a set of random individuals. Each of them signifies a possible optimal solution. The population will be gradually replaced by substituting the current population with a new generation using certain stochastic operators.

Considering the wide range of these algorithms, they always share a key characteristic: search processes may be categorized into exactly two stages, namely exploration and exploitation [7]. As a result, in the first stages of the search process, a carefully designed optimizer must show exploration behaviors that are sufficiently mixed with randomness in order to provide a greater number of random solutions. Furthermore, it enhanced multiple elements of the search space. After the exploration stage is completed, the exploitation stage is performed. The optimizer accelerates the search process by emphasizing a particular area instead of the whole search space, emphasizing near-optimal solutions. A successful optimizer must achieve an adequate and accurate balance between the exploration and exploitation stages. On the other hand, the probability of being trapped in local optima and overcoming partial convergence challenges increases. According to the No Free Lunch theorem [8], all the proposed metaheuristic algorithms exhibit similar average performance when solving a possible optimization problem. Meanwhile, no algorithm can be regarded as absolutely efficient. As a result, this theorem encourages the research and enhancement of more effective optimization algorithms.

Chaos theory focuses on the study of unpredictable and irregular system motions that are highly sensitive to initial conditions. A deterministic nonlinear dynamical system is called chaotic if it has at least one positive Lyapunov exponent. Several practical applications of chaos were shown in the literature, including in biology [9–11], ecology [12], infectious diseases [13], control [14], cryptosystems [15], and secure communication [16, 17]. Traditional optimization techniques, such as gradient, Newton, and Hessians methods, may successfully determine global optimum solutions for continuously differentiable functions, demonstrating fast convergence and high

accuracy. However, these classical optimization techniques can become trapped in local optima when solving optimization problems that require various multimodal functions [18].

Motivation

The chaotic spider wasp optimization algorithm is developed to address the limitations of traditional optimization methods, such as getting trapped in local minima and being time-consuming. Inspired by the hunting, nesting, and mating behaviors of female spider wasps, this algorithm offers a novel approach to solving complex optimization problems with improved exploration and exploitation capabilities. By using chaotic maps, the algorithm is better able to avoid local optima and reach faster convergence. This makes it more useful and efficient for many situations.

Contributions

The main contributions of the current paper are summarized as follows:

- An improved optimization algorithm inspired by hunting, nesting, and mating behaviors of female spider wasps is developed based on chaos theory.
- A detailed comparison between the traditional SWO algorithm and the CSWO algorithm is presented.
- The performance analysis and speed convergence of four different counterparts of the CSWO algorithm are analyzed through several unimodal and multimodal benchmark functions.
- The experimental results show that the CSWO algorithm has better performance compared to the SWO counterpart and is more efficient for solving real-world optimization problems.

The remainder of this paper is organized as follows: In Section 2, the literature review is provided. Section 3 reports a short introduction to the chaos theory and some properties of chaotic systems. In addition, a general concept of using chaos in optimization algorithms is presented. Section 4 introduces five well-known 1-D chaotic maps, their chaotic behaviors, and Lyapunov exponents. In Section 5, the traditional SWO algorithm is presented. Section 6 introduces new counterparts of the SWO algorithm that are based on chaos theory. Section 7 deals with experimental analysis and statistical testing, in which the CSWO algorithms are benchmarked on eight test functions. A qualitative analysis is presented in order to compare the traditional SWO and the proposed chaotic methods. Section 8 gives the discussion and conclusion.

2 Literature review

Recently, there has been increasing interest in the study and application of meta-heuristic algorithms for solving optimization problems. In the scientific literature, population-based meta-heuristic algorithms can be divided into four main categories based on their basic concepts: evolutionary algorithms [19], physics-based algorithms [20], human-based algorithms [21], and swarm-based algorithms [22]. Evolutionary algorithms mimic the mechanisms of biological evolution, such as recombination and mutation. The Genetic Algorithm [23], Biogeography-Based Optimizer [24], and Mind Evolutionary Optimizer [25] are all considered the most important evolutionary algorithms. Algorithms inspired by physical phenomena use hypotheses based on scientific concepts, such as gravitation and magnetic attraction. Some examples are the Gravitational Search Algorithm [26] and the Energy Valley Optimizer [27]. Human-based machine agents frequently mimic certain human behaviors. Socio Evolution and Learning Optimization [28], Human Felicity Algorithm [29], and Social Network Search [30] are a few examples of this classification. Swarm-based multi-agent systems imitate the social behaviors shown by animals that live in swarms or groups. Particle Swarm Optimization [31] and Salp Swarm Algorithm [32] are considered the most significant meta-heuristic algorithms in this specific field.

Nowadays, swarm-based multi-agent systems have attracted increased interest due to their various sources of inspiration and efficiency in solving an extensive variety of optimization problems. A new optimization algorithms in this field have been developed, including the Mountain Gazelle Optimizer [33], Spotted Hyena Optimizer [34], Honey Bee Mating Optimization [35], Butterfly Optimization Algorithm [36], Ant Lion Optimizer [37], Harris Hawks Optimizer [38], Bat-Inspired Algorithm [39], Fruit Fly Optimization Algorithm [40], Whale Optimization Algorithm [41], Grasshopper Optimization Algorithm [42], Artificial Gorilla Troops Optimizer [43], Grey Wolf Optimizer [44], Marine Predators Algorithm [45], Hunger Games Search [46], Aquila Optimizer [47], and many others. Because of their stochastic nature, meta-heuristic algorithms have enhanced flexibility for escaping constraint in local optima. These algorithms may be used across numerous fields according to their efficiency, adaptability, and highly effective performance. The main challenge in designing any meta-heuristic algorithm arises from the stochastic nature of the optimization process, requiring sufficient balance between exploration and exploitation stages [48]. The exploration stage allows the optimizer to fully investigate the search space on a global scale. Additionally, the population faces sudden and significant changes during this period. On the other hand, the exploitation stage focuses on improving possible solutions that were discovered in the exploration stage. In this context, the population undergoes small and sudden fluctuations.

The SWO algorithm is a novel meta-heuristic optimizer developed to solve continuous optimization problems. In particular, it can solve complex nonlinear engineering optimization problems by mimicking biological or physical phenomena [49]. The SWO algorithm was created from a mathematical model of the three different behaviors shown by female spider wasps, including nesting, hunting, and mating behaviors.

The literature has extensively studied the application of chaos theory in the development of optimization algorithms. In [50], the authors have enhanced the Chaotic Whale Optimization Algorithm by incorporating various chaotic maps to improve its performance and achieve the global optimum for several test functions. Arora et al. [51] have developed a novel meta-heuristic optimization algorithm called the Grasshopper Optimization Algorithm inspired by grasshoppers' swarming behavior. To enhance global convergence, chaos theory was included in the optimization process, using chaotic maps to balance exploration and exploitation over the optimization process. In [52], the authors have developed an improved meta-heuristic optimization algorithm called the Chaotic Bird Swarm Algorithm. In order to improve this algorithm's exploitation performance, they used different chaotic maps. Kiani et al. [53] have proposed the Chaotic Sand Cat Swarm Optimization, and they introduced chaotic maps to enhance the performance of this algorithm. In addition, they applied the chaotic algorithm to a total of 39 functions and multidisciplinary problems. Arora et al. [54] have introduced chaos into the Butterfly Optimization Algorithm in order to increase its performance and convergence speed. They concluded that using chaos can enhance the optimization process to exploit the algorithm for solving engineering design problems. Shinde et al. [55] have presented a developed counterpart of the meta-heuristic Sine-Cosine Algorithm, which is based on chaos theory. The suggested algorithm is inspired by the sine and cosine classical functions. Using different chaotic maps, they replaced the random parameters in the traditional algorithm with chaotic variables to enhance the performance of the proposed algorithm. Hamaizia and Lozi [56] have developed a novel strategy for global search and multidimensional chaotic attractors using a locally averaged method. In addition, they examined the robustness of the suggested approach using several benchmark functions.

Based on the best knowledge gained from the literature review, there are only a few papers that integrate chaos theory in meta-heuristic optimization algorithms, so it is necessary to develop a new meta-heuristic algorithm based on chaotic systems. However, crucial properties of discrete

chaotic dynamic systems, such as sensitivity to initial conditions, ergodicity, and unpredictability, are prone to designing an optimizer. In the next section, based on the mathematical modeling of female spider wasp behaviors, an improved chaotic meta-heuristic algorithm is developed to handle optimization applications to address this research gap.

3 Chaotic optimization algorithm

In this section, we will describe some properties of chaos phenomena, followed by an optimization algorithm using chaotic maps.

Chaos

Nonlinear systems often exhibit chaotic behavior. It describes an example of irregular motions exhibited by deterministic systems inside a bounded phase space. Chaos theory is explained as the phenomenon known as the "butterfly effect", which was first described by Lorenz in 1963. Lorenz observed that slight variations in initial conditions could result in significantly different outcomes in future scenarios. Chaos is a result of the unpredictability produced by deterministic dynamical systems. Three fundamental properties characterize the chaotic systems: [57]

- ergodicity. Chaos has the ability to go through all possible states within a given range without repetition.
- sensibility. A very common characteristic of chaotic systems is their sensitive dependence on initial conditions. The system's behaviour may rapidly diverge with slightly different conditions, making it unpredictable.
- regularity. Chaos is exhibited by deterministic dynamical systems.

Chaos is a complex and unpredictable phenomenon that exhibits non-linear behavior. The ergodicity of chaos implies that using chaotic variables for optimization may provide an advantage compared to random searches with stochastic variables. It has been able to prevent algorithms from getting trapped in local optima. As a result, it is frequently used for optimization problems.

A general idea of a chaotic optimization algorithm

A random-based optimization algorithm that uses random number sequences obtained from chaotic maps instead of random number generators is called a chaotic optimization algorithm (COA). Its properties include simple integration, quick execution, and effective methods for avoiding local optimization. Consequently, it has enormous potential as a tool for engineering applications [58]. The COA is a highly efficient method for solving the optimization problems of a nonlinear multimodal function with boundary constraints. Chaos, unlike stochastic searches that rely on probabilities, may do comprehensive searches at faster rates due to its lack of repetition.

The COA generally has two main stages: the global stage and the local stage, which are often characterized by chaotic methods. Firstly, in the global stage, chaotic points are selected from the search domain $[L, U]$ based on a specific chaotic model. Next, the objective function is determined at various positions, and the point with the minimum objective function is chosen as the current optimum. Furthermore, it is assumed that during the local stage, the current optimum will be nearly the global optimum after a certain number of iterations. The current optimum is regarded as a center with minimal chaotic disturbances, whereas the global optimum is determined through an extensive search. The chaos phenomenon is characterized by randomness. Usually, a deterministic function can display chaotic behavior for some initial conditions and parameter values. These functions are so-called chaotic maps.

4 Chaotic maps

This section presents five one-dimensional non-invertible chaotic maps that are used to generate chaotic sequences. We use the logistic, Gauss/mouse, sinusoidal, piecewise, and tent chaotic maps in our study [59]. The details of these chaotic maps are shown in Table 1. The chaotic behavior of

Table 1. Chaotic maps

Chaotic map	Equation	Range	Parameter	Initial condition
Logistic	$x_{n+1} = \mu x_n (1 - x_n)$	$[0, 1]$	$\mu = 3.9$	$x_0 = 0.6$
Gauss/mouse	$x_{n+1} = \begin{cases} 0 & x_n = 0, \\ \frac{1}{x_n} - \lfloor \frac{1}{x_n} \rfloor & \text{otherwise} \end{cases}$	$[0, 1]$		$x_0 = 0.7$
Sinusoidal	$x_{n+1} = a x_n^2 \sin(\pi x_n)$	$[0, 1]$	$a = 2.3$	$x_0 = 0.9$
Piecewise	$x_{n+1} = \begin{cases} \frac{x_n}{p} & 0 \le x_n < p, \\ \frac{x_n - p}{0.5 - p} & p \le x_n < \frac{1}{2}, \\ \frac{1 - p - x_n}{0.5 - p} & \frac{1}{2} \le x_n < 1 - p, \\ \frac{1 - x_n}{p} & 1 - p \le x_n < 1 \end{cases}$	$[0, 1]$	$p = 0.4$	$x_0 = 0.8$
Tent	$x_{n+1} = \begin{cases} \frac{x_n}{0.7} & x_n < 0.7, \\ \frac{10}{3}(1 - x_n) & x_n \ge 0.7 \end{cases}$	$[0, 1]$		$x_0 = 0.4$

the proposed maps is depicted in Figure 1.

The rationale behind selecting chaotic maps, such as logistic, Gauss/mouse, sinusoidal, piecewise,

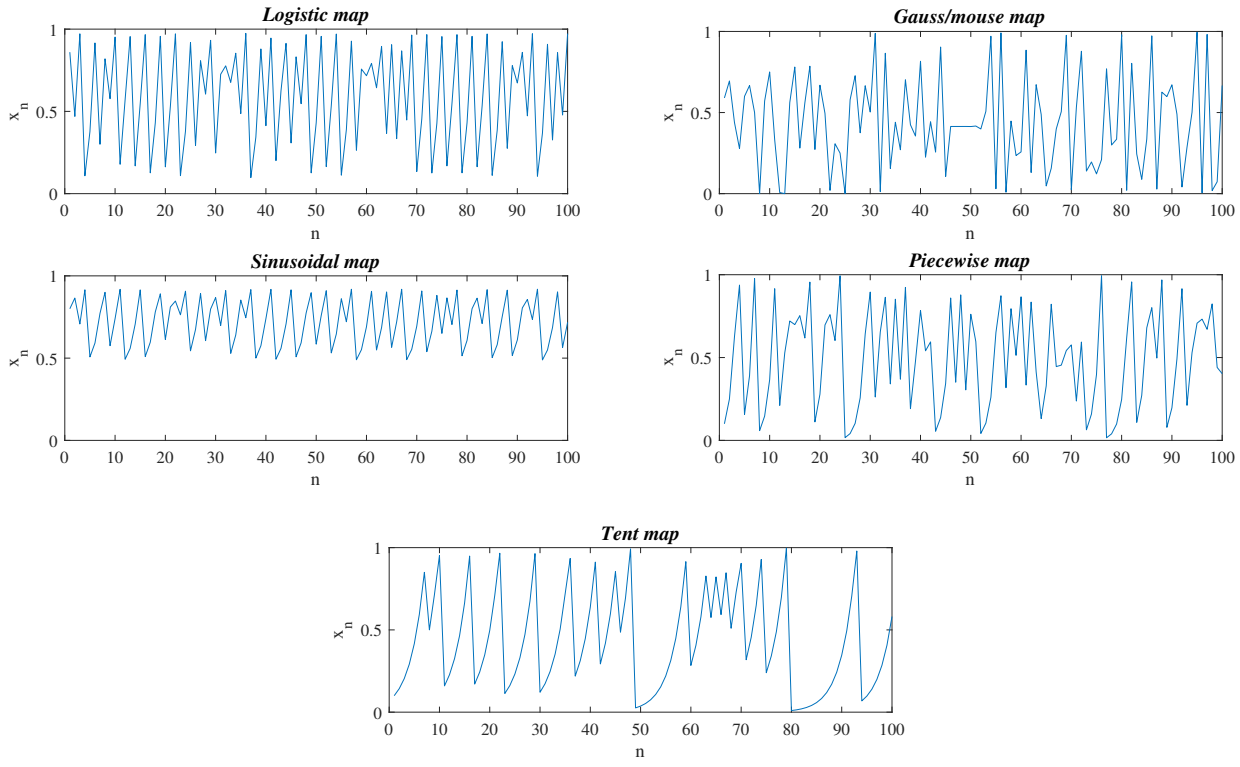

Figure 1. Visualization of chaotic maps

and tent maps, to replace random parameters in the SWO algorithm lies in the following reasons:

1. Chaotic maps provide better exploration and exploitation.
   - Exploration. Chaotic maps generate sequences that are deterministic yet appear random.

These sequences can help the SWO algorithm explore the search space more effectively than purely random numbers, as they avoid premature convergence to local optima.
- Exploitation. Chaotic maps can also provide fine-grained control over the search process, allowing the SWO algorithm to exploit promising regions more efficiently.

2. Avoidance of randomness pitfalls. Traditional random number generators may lead to uneven exploration of the search space, causing the algorithm to get stuck in suboptimal regions. On the other hand, chaotic maps provide a more structured and diverse exploration, reducing the likelihood of stagnation.
3. Diversity in search patterns. Each chaotic map has unique dynamics and properties. By incorporating multiple chaotic maps, the SWO algorithm can leverage different patterns of exploration, ensuring a more robust search process.
4. Improved convergence and stability. Chaotic maps can help the SWO algorithm converge faster to the global optimum by maintaining a balance between exploration and exploitation. They also reduce the risk of premature convergence, which is common in traditional random-based algorithms.

Note that we can use other chaotic maps not listed here to enhance the performance of the traditional SWO algorithm if they generate chaotic numbers in the range $[0, 1]$ with absolute value. In this paper, we select five chaotic maps that generate chaotic numbers in the range $[0, 1]$, which is consistent with the range of random parameters in the SWO algorithm.

Quantitative measure of chaos

In chaos theory, the rate of divergence or convergence of nearby trajectories of a deterministic dynamical system is evaluated by using Lyapunov exponents. In particular, when the maximum Lyapunov exponent (MLE) is positive, the system is chaotic. For a one-dimensional dynamic system, the Lyapunov exponent is defined as [60]

$$\lambda = \lim_{n\to\infty} \frac{1}{n} \sum_{j=0}^{n-1} \ln |f'(x_j)|, \tag{1}$$

where n is the maximum iteration number, $f'(x_j)$ is the derivative of $f(x_j)$.
Based on the above formula, we compute the maximum Lyapunov exponent for the five chaotic maps for 1000 iterations. The average values of MLEs are given in Table 2.

Table 2. Maximum Lyapunov exponent of the ten chaotic maps

Map	Logistic	Gauss/mouse	Sinusoidal	Piecewise	Tent
MLE	0.693	0.721	0.682	1.532	0.693

It can be shown from Table 2 that the maximum Lyapunov exponent is positive for the five maps, meaning that these maps exhibit chaotic behavior. Therefore, they may be used with accuracy in chaotic optimisation algorithms.

5 Spider wasp optimization algorithm

The spider wasp optimization algorithm is a nature-inspired meta-heuristic algorithm that imitates the hunting, nesting, and mating behaviors of female spider wasps used in optimization problems. This work will develop a novel variant of the optimization strategy inspired by the hunting and nesting behaviors of some wasp species, as well as their practice of required brood parasitism,

which involves dropping a single egg into each spider's abdomen. Firstly, female spider wasps investigate the surrounding habitats in search of appropriate spiders. They then immobilize and transport the spiders to pre-prepared nests that are perfect for their needs. This behavior serves as the primary motivation for the SWO algorithm. Once they have located adequate prey and nests, they proceed to pull them into the nests. They then place an egg on the spider's abdomen, closing the nest. The SWO approach randomly distributes a specified number of female wasps over the search space. Each individual will systematically explore the search region in continuous motion, looking for a spider suitable for the gender of its offspring, as determined by the haplodiploid sex-determination system intrinsic to all hymenopterans. The search depends on their predatory and tracking behaviors. After finding suitable spiders, female spider wasps will remove them from the center region of the spider's web and systematically search the ground six times to recover any spiders that have fallen from the web [61]. Next, the female wasps will attack the victim and try to paralyze it for transmission to the selected nest. After putting an egg inside the spider's abdomen, the female wasp next closes the nest.

Figure 2. The female spider wasp in nature engages in hunting behavior

The following is a brief description of the wasp behaviors investigated in this study:

- Searching behavior. This behavior includes an aggressive search of prey during the first stages of optimization to determine a spider appropriate to larval growth.
- Following and escaping behavior: once they locate their prey or spiders, they may make an effort to quickly escape the central area of the spider web. As a result, the female wasp chases them, immobilizing and pulling the most suitable one.
- Nesting behavior. This behavior mimics the way in which prey is dragged to nests that are suitable in size for both the prey and the egg.
- Mating behavior. This behavior emulates the characteristics of the offspring produced by hatching the egg via the uniform crossover operator between male and female wasps, controlled by a certain probability called the crossover rate.

In the following, we will present the mathematical model for these four behaviors.

Hunting and nesting behavior

The female spider wasp initiates a first search, referred to as an "exploration operator", to discover potential prey. Once the target is identified, the entity transmits a signal to its operator responsible for exploiting the situation, initiating the process of approaching and launching an assault. The

mathematical specifics of these two operators are shown here.

Search stage (Exploration operator)

As previously mentioned, the female spider wasp initiates this operation at the onset of its search for its preferred food. This behaviour may be mathematically represented by the following expression:

$$\vec{x}_i^{t+1} = \vec{x}_i^t + \mu_1 * \left(\vec{x}_a^t - \vec{x}_b^t\right), \tag{2}$$

where a and b are two indices being randomly chosen from the current population, which are used to find the direction of investigation by the female wasps, and μ_1 is used to ascertain the consistent movement in this particular direction by means of the following equation:

$$\mu_1 = |r_n| * r_1, \tag{3}$$

where r_1 is a random number in $[0, 1]$ and r_n is a random number that has been picked from a normal distribution. If the female wasps are unable to grab it, prey that falls from the orb may be lost. In order to locate the missing prey, they use an alternative exploration approach, which may be precisely modeled using the following mathematical formula:

$$\vec{x}_i^{t+1} = \vec{x}_c^t + \mu_2 * \left(\overrightarrow{L} + \overrightarrow{r_2} * (\overrightarrow{U} - \overrightarrow{L})\right), \tag{4}$$

$$\mu_2 = B * \cos(2\pi l), \tag{5}$$

$$B = \frac{1}{1 + \exp(l)}, \tag{6}$$

where c, an index that represents the position of the dropped prey, is randomly chosen from the population. $\overrightarrow{L}$ and $\overrightarrow{U}$ represent the lower and upper bounds, respectively. l is a number randomly chosen from the interval $[-2, -1]$, whereas, $\vec{r_2}$ is a random vector in $[0, 1]$. The value of μ_2, which is between the range of -0.8 to 0.8, defines the direction of the search. This helps to prevent any incorrect direction that may be determined by Eq. (2). In order to enhance investigation and identify the most favorable areas, we assume that the following tradeoff between Eq. (2) and Eq. (4) is satisfied.

$$\vec{x}_i^{t+1} = \begin{cases} Eq.\ (2) & r_3 < r_4, \\ Eq.\ (4) & otherwise, \end{cases} \tag{7}$$

where r_3 and r_4 are two random numbers chosen from the range $[0, 1]$.

Following and escaping stage (exploration and exploitation operator)

Upon locating its prey, the wasp initiates pursuit of the spider. This behavior may be mathematically modeled in the following manner:

$$\vec{x}_i^{t+1} = \vec{x}_i^t + C * |2 * \overrightarrow{r_5} * \vec{x}_a^t - \vec{x}_i^t|, \tag{8}$$

$$C = \left(2 - 2 * \left(\frac{t}{t_{max}}\right)\right) * r_6, \tag{9}$$

where a is an index randomly selected from the population. t and t_{max} represent the current and maximum evaluations, respectively. $\overrightarrow{r_5}$ is a random vector in $[0,1]$, and r_6 is a random number in $[0,1]$. C is a parameter that modulates the wasp's speed according to distance, starting at a speed of two and progressively decreasing to zero. As the female wasp chases the spider, the distance between them gradually increases. This time period is mostly defined by exploitation. As the distance expands, exploitation evolves into exploration. This behavior is mimicked using the following formula:

$$\vec{x}_i^{t+1} = \vec{x}_i^t * \overrightarrow{v_c}, \tag{10}$$

where $\overrightarrow{v_c}$ is a vector of numerical values in the range $[-k,k]$ according to the normal distribution. k is given by the following formula:

$$k = 1 - \frac{t}{t_{max}}. \tag{11}$$

The next equation will be used for achieving the tradeoff between Eq. (8) and Eq. (10).

$$\vec{x}_i^{t+1} = \begin{cases} Eq.\ (8) & r_3 < r_4, \\ Eq.\ (10) & otherwise. \end{cases} \tag{12}$$

Further, the tradeoff between searching in Eq. (7) and Eq. (12) is described by the following equation:

$$\vec{x}_i^{t+1} = \begin{cases} Eq.\ (7) & p < k, \\ Eq.\ (12) & otherwise, \end{cases} \tag{13}$$

where p represents a random number in the range $[0,1]$.

Nesting behavior (exploitation operator)

Female wasps retrieve the damaged spider and bring it back to their nest. Spider wasps have the ability to excavate and construct chambers in the ground, build nests using mud on leaves or rocks, and make use of pre-existing nests or holes. Spider wasps exhibit diverse nesting behaviours, and as a result, the SWO algorithm employs two equations to represent these behaviors accurately. The first equation evaluates the spider's attraction to an area that provides the best conditions for nesting with its egg on its abdomen. This equation is given by:

$$\vec{x}_i^{t+1} = \vec{x}_i^t + \cos(2\pi l) * \left(\vec{x}^* - \vec{x}_i^t\right), \tag{14}$$

where $\vec{x}^*$ represents the optimum solution gained so far. The second equation establishes the nest at the position of a female spider, selected randomly from the population. This equation has a supplementary step size to guarantee that no two nests are constructed at the same location. Thus, we have the following equation:

$$\vec{x}_i^{t+1} = \vec{x}_a^t + r_3 * |\delta| * \left(\vec{x}_a^t - \vec{x}_i^t\right) + (1 - r_3) * \overrightarrow{H} * \left(\vec{x}_b^t - \vec{x}_c^t\right), \tag{15}$$

where a, b, and c represent the indices of three solutions randomly selected from the population. δ is a number determined by the Levy flight, and r_3 is a random number in the range $[0,1]$. $\overrightarrow{H}$ is a

binary vector that represents when a step size must be performed to avoid the construction of two nests at the same location. $\overrightarrow{H}$ is given as:

$$\overrightarrow{H} = \begin{cases} 1 & \overrightarrow{r_4} > \overrightarrow{r_5}, \\ 0 & otherwise, \end{cases} \tag{16}$$

where $\overrightarrow{r_4}$ and $\overrightarrow{r_5}$ are two random vectors in $[0, 1]$. In order to update each solution during the optimisation process, a random swap is performed between Eq. (14) and Eq. (20) using the following formula:

$$\vec{x}_i^{t+1} = \begin{cases} Eq.\ (14) & r_3 < r_4, \\ Eq.\ (20) & otherwise. \end{cases} \tag{17}$$

The tradeoff between hunting and nesting behaviors may be expressed using the following equation:

$$\vec{x}_i^{t+1} = \begin{cases} Eq.\ (13) & i < N * k, \\ Eq.\ (17) & otherwise, \end{cases} \tag{18}$$

where N represents the population size.

Mating behavior

At this stage, spider wasps have an important capacity to determine gender. This depends on the size of the egg. Smaller spider wasps imply males, and bigger wasps imply females. In our approach, each spider wasp represents a possible solution in the current generation, while the spider wasp egg signifies the newly generated possible solution in that same generation. The new solutions, also known as spider wasp eggs, are described by the following equation:

$$\vec{x}_i^{t+1} = Crossover\left(\vec{x}_i^t, \vec{x}_m^t, CR\right), \tag{19}$$

where *Crossover* represents the uniform crossover operator applied to the vectors $\vec{x}_i^t$ and $\vec{x}_m^t$ for the female and male spider wasps, respectively, with a probability CR. The male spider wasp is generated using the SWO algorithm to exhibit distinct characteristics from the female wasps, according to the following formula:

$$\vec{x}_i^{t+1} = \vec{x}_i^t + \exp(l) * |\beta| * \vec{v}_1 + (1 - \exp(l)) * |\beta_1| * \vec{v}_2, \tag{20}$$

where β and β_1 are two numbers picked randomly from the normal distribution. $\vec{v}_1$ and $\vec{v}_2$ are two vectors constructed using the following formula:

$$\begin{aligned} \vec{v}_1 &= \begin{cases} \vec{x}_a - \vec{x}_i & f\left(\vec{x}_a\right) < f\left(\vec{x}_i\right), \\ \vec{x}_i - \vec{x}_a & otherwise, \end{cases} \\ \vec{v}_2 &= \begin{cases} \vec{x}_b - \vec{x}_c & f\left(\vec{x}_b\right) < f\left(\vec{x}_c\right), \\ \vec{x}_c - \vec{x}_b & otherwise, \end{cases} \end{aligned} \tag{21}$$

where a, b, and c are distinct indices representing three solutions randomly chosen from the population. $f(\vec{x}_i)$ is the objective function that represents an individual in the population. Finally,

we denote by (TR) the tradeoff rate that determines the compromise between hunting, nesting, and mating behaviors.

Population reduction and conserving memory

After the female spider lays an egg on the host's abdomen, she closes the nest and discreetly vacates the location. This hypothesis suggests that the female's role in the optimization process is almost complete, and transferring the function evaluation to other wasps for the remainder of the process may provide enhanced results. Over the iteration, some wasps in the population will be killed to provide more function evaluations to the surviving wasps. This mechanism reduces population variety, hence accelerating convergence towards the near-optimal solution. In each iteration of the function evaluations, the size of the new population will be adjusted according to the following equation:

$$N = N_{min} + (N - N_{min}) \times k, \tag{22}$$

where N_{min} denotes the minimal population size required to prevent the optimization process from being trapped in local minima. To enhance efficiency, the SWO applies a memory preservation method that transmits the highest rank of each wasp to the next generation. The proposed new location of each wasp is evaluated against its current position, and if it is worse, the next solution is substituted. The pseudo-code of the SWO algorithm is shown in Algorithm 1.

Algorithm 1: Pseudo-code of SWO algorithm

Input: N, N_{min}, CR, TR, t_{max}
Output: $\vec{x}_i^*$

```
Initialize N female wasps, x_i^t (i = 1, 2, ..., N), using Eq. (2)
Compute f(x_i^t) while storing x*
t = 1;
while (t < t_max) do
    r_6: generating a random number in the interval [0, 1]
    if (r_6 < TR) then
        for i = 1 : N do
            Update the position of x_i^t using Eq.(18) to x_i^{t+1}
            Compute f(x_i^{t+1})
            t = t + 1;
        end for
        else
        for i = 1 : N do
            Applying Eq. (19)
            t = t + 1;
        end for
    end if
    Applying Conserving Memory
    Updating N using Eq. (22)
end while
```

6 The improved chaotic spider wasp optimization algorithm

According to Eq. (3) and Eq. (9), the parameters r_1 and r_6 are the main variables of female spider wasps' convergence toward their objective throughout the SWO algorithm iterations. Chaotic maps have the capacity to enhance the performance of optimization methods. In the standard SWO algorithm, there is no need to preserve linearly decreasing values. In fact, using a chaotic variable that changes r_1 and r_2 may be better for the search. According to the SWO, the values obtained through the chaotic map must fall within the range of $[0,1]$. In the current work, the values of r_1 and r_6 highlighted in Algorithm 1 are substituted with the values generated by chaotic maps to give chaotic behavior to the r_1 and r_6 parameters. This may also lead to the approach converging to the optimal value quickly, as explained in the next section. As a result, this study focuses on the task of adjusting the values of r_1 and r_6 using several chaotic maps. Ten different kinds of SWO employ distinct chaotic maps. In the rest, we will construct four variants of the SWO by adopting the following manner:

- Eq. (23) is obtained by substituting the random parameter r_1 in the search stage that initiates the spider wasp optimization algorithm with the sequence obtained from ten different chaotic maps, where $r_1(t)$ represents the value that results from the chaotic map over the t-th iteration.
- Eq. (24) is obtained by substituting the random parameter r_6 in the following and escaping stage found in the spider wasp optimization algorithm with the sequence obtained from ten different chaotic maps, where $r_6(t)$ represents the value that results from the chaotic map over the t-th iteration.

$$CSWOA1 : \mu_1(t) = |r_n| * r_1(t), \tag{23}$$

$$CSWOA2 : C(t) = \left(2 - 2 * \left(\frac{t}{t_{max}}\right)\right) * r_6(t), \tag{24}$$

$$CSWOA3 : \mu_1(t) = |r_n| * r_1(t) \quad \text{and} \quad C(t) = \left(2 - 2 * \left(\frac{t}{t_{max}}\right)\right) * r_6(t). \tag{25}$$

CSWOA3 has been constructed by combining the CSWOA1 and CSWOA2 algorithms. In order to enrich the content of this study, the CSWOA4 is created by substituting the random number p in Eq. (13) in the SWO with chaotic maps. Thus, the parameter p will behave chaotically.

$$CSWOA4 : \vec{x}_i^{t+1} = \begin{cases} Eq.\ (7) & p(t) < k, \\ Eq.\ (12) & otherwise, \end{cases} \tag{26}$$

where $p(t)$ represents the value that results from the chaotic map over the t-th iteration.

The next section presents a comparative analysis using the distinct chaotic parameters mixed with the traditional SWO algorithm. Furthermore, the combination of these parameters (SWO with chaotic r_1, SWO with chaotic r_6, SWO with chaotic r_1 and chaotic r_6, and SWO with chaotic p) has been implemented and tested using several benchmark functions.

The key distinctions between these algorithms lie in the incorporation of chaos theory into their parameters, which enhances their exploration, exploitation, and convergence properties. In CSWOA1, CSWOA2, and CSWOA4, the integration of chaotic maps is applied only in one parameter (r_1, r_6, and p, respectively). However, in CSWOA3, we have incorporated both the parameters r_1 and r_6 with chaotic maps. This makes CSWOA3 more robust and effective for solving complex optimization problems compared to other algorithms. Integrating more than one chaotic map in CSWOA3 can improve its ability to explore the search space thoroughly because

each chaotic map generates more diverse and unpredictable sequences. The chaotic behaviors help the algorithm's convergence, enhancing both speed and precision, particularly in complex and multimodal optimization problems, by avoiding premature convergence. Therefore, by adopting two chaotic parameters in CSWOA3, we obtain a better balance between exploration and exploitation. This results in improved convergence and higher-quality solutions. In conclusion, incorporating chaos theory into CSWOA1, CSWOA2, CSWOA3, and CSWOA4 adds stochasticity and unpredictability, which improves the algorithm's ability to escape local optima and explore the search space more effectively.

In practice, the chaotic spider wasp optimization algorithm avoids local optima by employing several strategies:

- Using chaotic maps allows providing randomness and diversity, where the algorithm can escape local optima by generating diverse solutions that might not be reachable through deterministic methods.
- Balancing exploration and exploitation through a dual population strategy. The algorithm mimics the behavior of spider wasps, which use two populations: spiders (prey) and wasps (predators). The interactions between these two populations ensure a balance between exploration (wasps searching for spiders) and exploitation (spiders trying to escape). This dual strategy helps avoid stagnation in local optima by maintaining diversity in the search process.
- Dynamically adjusting parameters to adapt to the search process. This allows the algorithm to switch between exploration and exploitation stages, reducing the risk of getting trapped in local optima.
- Introducing random perturbations to escape stagnation. These perturbations help the algorithm explore new regions of the search space, even after it has found a promising solution.
- Maintaining population diversity through fitness-based selection in order to prioritize better solutions to contribute to the search process. These mechanisms collectively enable the CSWO algorithm to explore the search space more effectively and avoid getting stuck in suboptimal solutions.

The following objectives may help demonstrate the theoretical efficiency of the suggested chaotic algorithms:

- The chaotic r_1 aids CSWO in dynamically updating the locations of female spider wasps in a chaotic manner, which can improve the exploration process.
- As we mentioned, C in Eq. (9) is a distance-controlling factor that determines the speed of the wasp when it starts chasing the prey (spider). Thus, the chaotic r_6 provides greater speed to CSWO in the exploitation stage than the SWO because the r_6 may have various values.
- Various chaotic maps for r_1, r_6 and p provide better exploration and exploitation behaviors for the CSWO algorithm.
- Chaotic maps aid the CSWO in escaping local optima when confronted with this problem.

The pseudo-code of the CSWO algorithm is shown in Algorithm 2.

For completeness, we provide a comparison of computational complexity between SWO and CSWO algorithms in Table 3.

From Table 3, it is shown that the computational complexity of CSWO is slightly higher than SWO due to the additional chaotic behavior. The complexity for SWO can be considered as $O(t_{max} \cdot N)$, while $O(t_{max} \cdot N \cdot C)$ for CSWO, where N is the population size, t_{max} is the maximum number of iterations, and C is the complexity introduced by the chaotic maps. On the other hand, the time

Algorithm 2: Pseudo-code of CSWO algorithm

Input: N, N_{min}, CR, TR, t_{max}
Output: $\vec{x}_i^*$

```
Initialize N female wasps, x_i^t (i = 1, 2, ..., N), using Eq. (2)
Evaluate each x_i^t and calculate the fitness of each search agent
x* = the best search agent
Initialize the value of the chaotic map x_0 randomly
t = 1;
while (t < t_max) do
    Update the chaotic number using the chaotic map function
    if (r_6 < TR) then
        for i = 1 : N do
            Update the position of x_i^t using Eq. (18) to x_i^{t+1}
            Compute f(x_i^{t+1})
            t = t + 1;
        end for
        else
        for i = 1 : N do
            Applying Eq. (19)
            t = t + 1;
        end for
    end if
    Applying Conserving Memory
    Updating N using Eq. (22)
end while
```

Table 3. Comparison of computational complexity of SWO and CSWO algorithms

Feature	SWO	CSWO
Initialization	Random population of agents	Random population of agents
Evaluation	Objective function	Objective function
Position update	Random and local search	Chaotic maps enhance search
Mating behavior	Information exchange	Information exchange
Selection	Best solution selected	Best solution selected
Computational complexity	$O(t_{max} \cdot N)$	$O(t_{max} \cdot N \cdot C)$
Time complexity	$O(t_{max} \cdot D \cdot N) + O(t_{max} \cdot D \cdot N)$	$O(t_{max} \cdot D \cdot N \cdot C) + O(t_{max} \cdot D \cdot N)$
Exploration capability	Standard	Enhanced by chaotic behavior
Exploitation capability	Standard	Enhanced by chaotic behavior

complexity for the SWO algorithm is designed as:

$$\begin{aligned} T(SWO) &= T(\text{Hunting and Nesting behaviors}) + T(\text{Mating behavior}) \\ &= O(t_{max} \cdot D \cdot N) + O(t_{max} \cdot D \cdot N), \end{aligned} \tag{27}$$

where D is the dimension of the search space. For the CSWO algorithm, the time complexity is designed as:

$$T(CSWO) = O(t_{max} \cdot D \cdot N \cdot C) + O(t_{max} \cdot D \cdot N). \tag{28}$$

Therefore, CSWO generally offers better performance in terms of exploration and exploitation due to the integration of chaotic maps, but at the cost of increased both computational and time complexities compared to the standard SWO algorithm.

7 Experimental setup and result discussions

The accuracy of the proposed meta-heuristic algorithms will be compared to traditional SWO using a set of eight well-known unimodal or multimodal benchmark functions. Unimodal benchmark functions provide only one optima and are very suitable for evaluating and comparing exploitation strategies. In contrast, multimodal benchmark functions include several optima, which renders them more complex than unimodal functions. The term "global optima" means that there exists a single optima, whereas the rest are known as "local optima". The key property of any efficient meta-heuristic algorithm is its capacity to avoid local optima and determine the global optimum. The primary goal of multimodal benchmark functions is to evaluate the exploration's performance in order to avoid trapping in local optima. Table 4 presents a summary of the test functions, including their range of optimization variables, their dimension Dim, and their minima f_{min}. Furthermore, the topologies of benchmark functions are represented in Figure 3. As shown in Table 4, the minima of all the proposed test functions is 0, except for the Schwefel function. Among the proposed benchmark functions, the unimodal functions are F_1, F_2, F_3, F_7 and F_8. In contrast, F_4, F_5 and F_6 are multimodal functions.

Table 4. List of eight benchmark functions

Function name	Formula	Dim	Search space	f_{min}
Sphere	$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
Quartic Noise	$F_2(x) = \sum_{i=1}^{n} ix_i^4 + rand(0,1)$	30	$[-1.28, 1.28]$	0
Rosenbrock	$F_3(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	30	$[-30, 30]$	0
Griewank	$F_4(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]$	0
Rastrigin	$F_5(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i + 10)\right]$	30	$[-5.12, 5.12]$	0
Schwefel	$F_6(x) = -\sum_{i=1}^{n} x_i \sin(\sqrt{\lvert x_i \rvert})$	30	$[-500, 500]$	-418.9829$\times D$
Schwefel 2.21	$F_7(x) = \max\{\lvert x_i \rvert, 1 \leq i \leq n\}$	30	$[-100, 100]$	0
Schwefel 2.22	$F_8(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, 10]$	0

We can measure each algorithm's performance using three distinct statistical tests: the best, the mean of the fitness function, and the standard deviation (STD).

1. **Statistical mean:** represents the average value of the best fitness function F_* obtained after performing the algorithm T_{max} iterations. It is computed as follows:

$$Mean = \frac{1}{T_{max}} \sum_{i=1}^{T_{max}} F_*^i. \tag{29}$$

2. **Statistical best:** represents the minimum value of the best fitness function F_* obtained after performing the algorithm T_{max} iterations, i.e;

$$Best = \min_{i=1}^{T_{max}} F_*^i. \tag{30}$$

3. **Statistical standard deviation:** is used as a performance test to verify the algorithm's stability

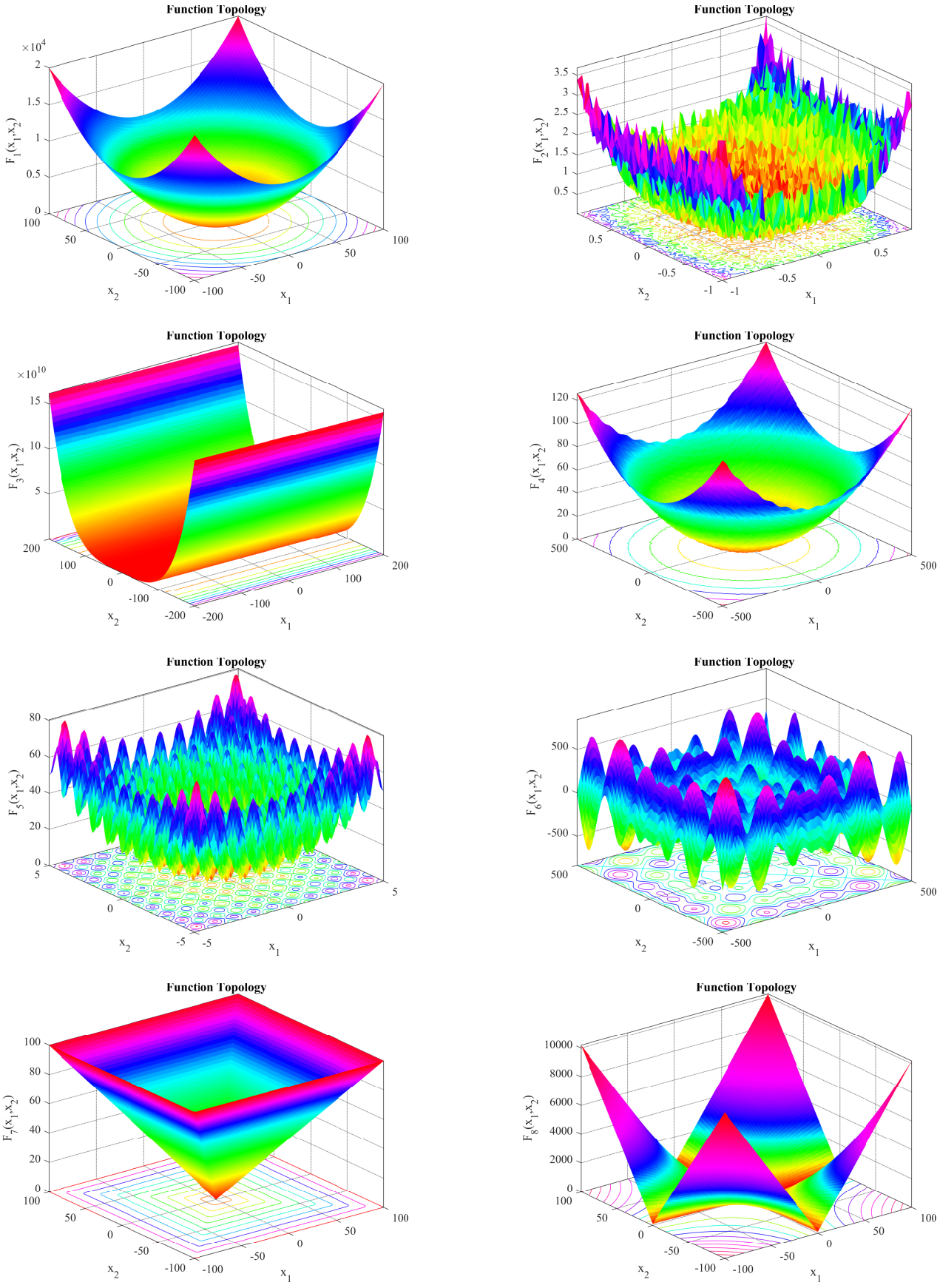

Figure 3. Topologies of the benchmark functions

and robustness. A lower standard deviation in the obtained solutions means that the algorithm accurately finds good solutions. It can be determined as:

$$STD = \sqrt{\frac{1}{T_{max}-1}\sum_{i=1}^{T_{max}}\left(F_*^i - Mean\right)^2}. \tag{31}$$

The performance of CSWO with chaos

All algorithms in the following numerical simulations were implemented using MATLAB software with the Microsoft Windows 10 operating system. All simulations are performed on the same PC with an Intel(R) Core(TM) i5-6300U 2.4 processor and 8GB of RAM. The proposed meta-heuristic algorithms are evaluated on various well-known benchmark functions using different chaotic maps. Their details can be found in Table 4. For all algorithms, the number of population is 30, the number of iterations is 500, and the number of independent runs is 10. The results of five chaotic maps applied from CSWO1 to CSWO4 are displayed in Table 6, where the best, the mean (average), and the STD of the best solutions obtained in the last iteration in simulation are illustrated. Function values showing the most optimal results are emphasized in bold. It can be shown from Table 6 that CSWO algorithms provide better results as compared to SWO algorithm. In particular, Gauss/mouse, piecewise, and tent maps yields better results. In contrast, logistic and sinusoidal maps perform less well when we implement CSWO compared to the SWO algorithm. Therefore, Gauss/mouse, piecewise, and tent maps may effectively improve the performance of SWO algorithm. Table 6 shows that the tent and piecewise-based SWO algorithms consistently generates the best solutions over all test functions. In the following, considering the mean and standard deviation statistical tests, a comparison is conducted between test functions that have been optimized using the spider wasp optimization algorithm (SWO) and test functions that have been optimized using chaotic maps.

- For function F_1, Gauss/mouse map-based CSWO 1, CSWO 2, and CSWO 3 algorithms, along with sinusoidal map-based CSWO 2, CSWO 3, and piecewise map-based CSWO 1, CSWO 2, and CSWO 3 algorithms, yield better results than SWO algorithm.
- For function F_2, all logistic map-based algorithms from CSWO 1 to CSWO 4, Gauss/mouse map-based CSWO 2, CSWO 3, and CSWO 4 algorithms, sinusoidal map-based CSWO 1, and CSWO 2 algorithms, piecewise map-based CSWO 2, CSWO 3, and CSWO 4 algorithms, and all tent map-based algorithms from CSWO 1 to CSWO 4 give better solutions than SWO algorithm.
- For function F_3, logistic map-based CSWO 4 algorithm, Gauss/mouse map-based CSWO 4 algorithm, piecewise map-based CSWO 2 and CSWO 3 algorithms, and tent map-based CSWO 3 and CSWO 4 algorithms yield better results compared to SWO algorithm.
- For functions F_4 and F_5, it has been demonstrated that most chaotic algorithms accurately provide the minima of these functions, which is 0, with the exception of some algorithms such as Gauss/mouse map-based CSWO 3 and CSWO 4 algorithms, and sinusoidal map-based CSWO 1 and CSWO 4 algorithms. As a result, the logsitic, piecewise, and tent maps can accurately improve the SWO algorithm's performance.
- For function F_6, most of the chaotic algorithms using logistic, Gauss/mouse, sinusoidal, piecewise, and tent maps give better results than SWO algorithm. Thus, using chaos can improve the research performance for minima of function F_6.
- For function F_7, logistic map-based CSWO 1, CSWO 2, and CSWO 4 algorithms, Gauss/mouse map-based CSWO 2 algorithm, sinusoidal map-based CSWO 3 algorithm, and tent map-based CSWO 2 algorithm provide better solutions compared to SWO algorithm. Obviously, using

chaotic maps for function F_7 can greatly enhance the optimization process's performance.
- For function F_8, logistic map-based CSWO 1, CSWO 2, and CSWO 4 algorithms, Gauss/mouse map-based CSWO 1 and CSWO 2 algorithms, sinusoidal map-based CSWO 1 and CSWO 3 algorithms, piecewise map-based CSWO 1, CSWO 3, and CSWO 4 algorithms, and all tent map-based algorithms from CSWO 1 to CSWO 4 display better solutions as compared to SWO algorithm.

According to the comparative study, the F_4 and F_5 functions demonstrated higher efficiency compared to the other functions in algorithms based on logistic, Gauss/mouse, sinusoidal, piecewise, and tent chaotic maps.

In the resolution of diverse optimization problems, both runtime and solution accuracy are of crucial importance. Table 5 shows the average runtime of the algorithm using different chaotic maps. From the table, it is shown that the runtime decreases when integrating chaotic maps into

Table 5. Runtime of SWO and CSWO algorithms (Unit: second)

Map	Metrics	SWO	CSWOA1	CSWOA2	CSWOA3	CSWOA4
No map	Time	45.2	N/A	N/A	N/A	N/A
	Rank	N/A	N/A	N/A	N/A	N/A
Logistic	Time	N/A	29.1	29.3	28.67	28.89
	Rank	N/A	3	4	1	2
Gauss/mouse	Time	N/A	27.4	28.32	27.1	28.12
	Rank	N/A	2	4	1	3
Sinusoidal	Time	N/A	28.56	29.94	28.42	28.16
	Rank	N/A	3	4	2	1
Piecewise	Time	N/A	31.62	34.26	30.99	31.44
	Rank	N/A	3	4	1	2
Tent	Time	N/A	31.84	31.72	30.25	30.59
	Rank	N/A	4	3	1	2
	Mean Rank	N/A	3.00	3.80	1.20	2.00
	Final Rank	5	3	4	1	2

the SWO algorithm. From the last row of the table, it can be seen that the average runtime of the CSWOA3 is ranked 1.20, placing it first overall, which is better compared to the SWO, CSWOA1, CSWOA2, CSWOA3, and CSWOA4. Therefore, incorporating chaos theory in the traditional SWO algorithm enhances its convergence speed, which ensures the efficiency of the CSWO algorithm for solving various optimization problems.

Qualitative analysis

A qualitative study has been conducted on several benchmark functions. Figure 4(a)-Figure 4(h) illustrate the convergence of several benchmark functions using the CSWO algorithm. These graphs provide an additional explanation of each algorithm's convergence rate, showing the best optimal solution obtained from 10 iterations of the algorithm using the tent chaotic map.

Figure 4(a) represents the convergence curves obtained using the tent map on the F_1 Sphere function, which has 0 as a global minimum. From Figure 4(a), CSWO 3 has the fastest convergence rate to the global solution. Similarly, CSWO 1 is very close to CSWO 3, which provides a very good convergence rate. On the other hand, SWO yields the slowest convergence rate when it comes to determining the global minimum during the optimization process.

Figure 4(b) depicts the convergence curves obtained using the tent map on the F_2 Quartic noise function, which has 0 as a global minimum. From Figure 4(b), CSWO 3 has the fastest convergence

rate to the global solution. Similarly, CSWO 4 is very close to CSWO 3, which gives a very good convergence rate. On the other hand, SWO yields the slowest convergence rate when it comes to determining the global minimum during the optimization process.

Figure 4(c) illustrates the convergence curves obtained using the tent map on the F_3 Rosenbrock function, which has 0 as a global minimum. As can be shown, CSWO 2 with CSWO 3 and CSWO 1 have the fastest convergence rate to the global solution. In contrast, SWO and CSWO 4 provide the slowest convergence rate over the maximum number of iterations.

Figure 4(d) displays the convergence curves obtained using the tent map on the F_4 Greiwank function. F4 has the property of being slightly easier to solve for dimensions that are higher rather than lower [62]. From Figure 4(d), it is shown that all of the algorithms failed to find the global solution over the maximum number of iterations, and they crashed before the first 60 iterations. Over the search process, CSWO 3 and CSWO 4 have the best convergence rate towards the global optimum.

Figure 4(e) shows the convergence curves obtained using the tent map on the F_5 Rastrigin function. As can be observed, all of the algorithms failed to find the global solution over the maximum number of iterations, and they crashed before the first 50 iterations. During the search process, CSWO 2 and CSWO 4, followed by SCWO 3 and SWO, have the best convergence rate towards the global optimum.

Figure 4(f) illustrates the convergence curves obtained using the tent map on the F_6 Schwefel function. As can be shown, CSWO 1 and CSWO 2, followed by CSWO 3 and CSWO 4, have the fastest convergence rate to the global solution. In contrast, SWO displays the slowest convergence rate over the maximum number of iterations.

Figure 4(g) represents the convergence curves obtained using the tent map on the unimodal F_7 Schwefel 2.21 function, which has 0 as a global minimum. As we can see, CSWO 1 has the fastest convergence rate to the global solution compared to the other algorithms. In particular, SWO provides the slowest convergence rate over the maximum number of iterations. As a result, the meta-heuristic algorithm's performance is improved when using chaotic maps.

Figure 4(h) shows the convergence curves obtained using the tent map on the unimodal F_8 Schwefel 2.22 function, which has 0 as a global minimum. Compared to the other algorithms, CSWO 1 and CSWO 3 exhibit the fastest convergence rate to the global solution. In particular, SWO and CSWO 4 display the slowest convergence rate over the maximum number of iterations. Finally, we can conclude that using chaos in meta-heuristic optimization algorithms provides better performance in the search process for the optimum solution.

8 Discussion and conclusions

This work has presented a novel variant of the SWO algorithm enhanced by chaos theory, developing the CSWO algorithm. Five chaotic maps were selected for improving the traditional SWO algorithm's efficiency by adjusting parameters. Inspired by the behaviors of female spider wasps, the CSWO algorithm imitates searching for a spider, escaping a falling spider, nesting the entrapped spider, and mating behavior during egg-laying. The robustness of the CSWO algorithm was analyzed using eight benchmark functions to evaluate exploitation, exploration, capacity to escape local optima, and convergence speed. The study showed that chaotic maps, particularly the tent and piecewise maps, significantly improved SWO performance. The incorporating of chaos increases the search speed for the best solution by replacing pseudo-random numbers with chaotic variables, enhancing the convergence rate during the optimization process.

Table 6. Statistical tests on benchmark functions using 5 chaotic maps on CSWO

Map	Algorithms		Functions							
			F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
No map	SWO	Best	2.26E−154	1.44E−05	8.44E−06	**0.00E+00**	**0.00E+00**	−1.97E+04	9.70E−82	6.02E−79
		Mean	5.14E−111	1.57E−04	1.10E−03	**0.00E+00**	**0.00E+00**	−1.85E+04	2.80E−68	5.16E−56
		STD	1.62E−110	1.40E−04	1.18E−03	**0.00E+00**	**0.00E+00**	1.24E+04	8.85E−68	1.63E−55
Logistic	CSWO 1	Best	8.93E−155	6.27E−06	3.13E−06	**0.00E+00**	**0.00E+00**	**−4.18E+03**	4.59E−77	3.34E−78
		Mean	2.50E−105	1.51E−04	2.09E−03	**0.00E+00**	**0.00E+00**	−3.67E+03	1.14E−71	3.69E−72
		STD	7.91E−105	1.76E−04	2.77E−03	**0.00E+00**	**0.00E+00**	8.32E+02	3.61E−71	7.63E−72
	CSWO 2	Best	3.90E−156	2.23E−05	4.78E−05	**0.00E+00**	**0.00E+00**	**−4.18E+03**	4.51E−81	1.39E−78
		Mean	7.88E−104	1.16E−04	1.43E−03	**0.00E+00**	**0.00E+00**	−3.49E+03	1.11E−71	**4.21E−75**
		STD	2.69E−103	8.84E−05	1.68E−03	**0.00E+00**	**0.00E+00**	8.94E+02	3.46E−71	**1.11E−74**
	CSWO 3	Best	1.51E−158	8.70E−06	5.17E−05	**0.00E+00**	**0.00E+00**	**−4.18E+03**	4.42E−77	9.84E−81
		Mean	5.94E−106	1.03E−04	1.39E−03	**0.00E+00**	**0.00E+00**	−3.54E+03	4.98E−57	2.47E−53
		STD	1.87E−105	1.18E−04	1.26E−03	**0.00E+00**	**0.00E+00**	8.32E+02	1.57E−56	7.81E−53
	CSWO 4	Best	1.90E−150	1.17E−05	5.85E−06	**0.00E+00**	**0.00E+00**	**−4.18E+03**	8.86E−76	3.97E−77
		Mean	5.98E−72	1.52E−04	2.00E−04	**0.00E+00**	**0.00E+00**	−3.50E+03	1.14E−71	1.35E−60
		STD	1.88E−71	1.33E−04	**1.87E−04**	**0.00E+00**	**0.00E+00**	8.86E+02	3.43E−71	4.29E−60
Gauss/mouse	CSWO 1	Best	6.96E−159	6.73E−06	2.03E−05	**0.00E+00**	**0.00E+00**	**−4.18E+03**	3.03E−80	5.66E−81
		Mean	1.41E−127	1.61E−04	2.22E−03	**0.00E+00**	**0.00E+00**	−3.28E+03	1.62E−52	4.23E−58
		STD	4.47E−127	1.57E−04	3.94E−03	**0.00E+00**	**0.00E+00**	8.47E+02	5.12E−52	1.33E−57
	CSWO 2	Best	9.29E−161	1.50E−05	1.62E−06	**0.00E+00**	**0.00E+00**	**−4.18E+03**	4.79E−79	2.05E−78
		Mean	**6.75E−142**	1.54E−04	2.09E−03	**0.00E+00**	**0.00E+00**	−3.34E+03	2.55E−66	1.99E−57
		STD	**2.11E−141**	1.06E−04	3.70E−03	**0.00E+00**	**0.00E+00**	8.99E+02	8.07E−66	6.31E−57
	CSWO 3	Best	8.15E−161	9.89E−06	1.34E−05	1.03E−04	1.34E−05	−4.13E+03	1.42E−78	2.60E−78
		Mean	1.11E−107	1.00E−04	2.09E−03	2.10E−02	1.72E−02	−2.45E+03	3.39E−59	6.41E−54
		STD	3.53E−107	8.44E−05	2.17E−03	2.89E−02	3.70E−02	7.33E+02	1.07E−58	2.02E−53
	CSWO 4	Best	**7.41E−166**	9.05E−06	1.63E−05	1.11E−16	1.42E−14	−2.33E+03	**4.15E−83**	**1.57E−83**
		Mean	8.11E−78	8.83E−05	8.77E−04	2.60E−07	2.77E−11	−2.09E+03	5.93E−40	1.27E−37
		STD	2.54E−77	6.07E−05	1.61E−03	7.76E−07	5.51E−11	1.94E+02	1.87E−39	3.68E−37

Map	Algorithms		Functions							
			F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
Sinusoidal	CSWO 1	Best	5.92E−160	1.20E−05	2.87E−05	1.04E−11	5.68E−14	−3.00E+03	2.79E−78	3.09E−79
		Mean	2.33E−105	1.17E−04	2.14E−03	4.21E−09	9.41E−10	−2.31E+03	2.12E−58	9.32E−63
		STD	7.37E−105	9.45E−05	3.08E−03	5.29E−09	9.69E−10	4.12E+02	6.72E−58	2.94E−62
	CSWO 2	Best	9.09E−155	1.12E−05	3.40E−05	**0.00E+00**	**0.00E+00**	**−4.18E+03**	1.41E−80	3.34E−81
		Mean	3.59E−128	1.12E−04	2.03E−03	**0.00E+00**	**0.00E+00**	−3.53E+03	1.72E−62	3.07E−55
		STD	1.13E−127	8.10E−05	3.23E−03	**0.00E+00**	**0.00E+00**	8.71E+02	5.46E−62	9.39E−55
	CSWO 3	Best	7.14E−157	3.41E−05	7.68E−06	3.33E−15	**0.00E+00**	**−4.18E+03**	2.39E−77	4.48E−82
		Mean	4.63E−111	1.79E−04	1.52E−03	4.67E−08	1.74E−10	−3.82E+03	**7.41E−73**	4.73E−70
		STD	1.46E−110	1.24E−04	3.35E−03	1.47E−07	3.31E−10	7.67E+02	**1.73E−72**	1.49E−69
	CSWO 4	Best	5.78E−22	2.25E−05	2.01E−03	**0.00E+00**	**0.00E+00**	−3.88E+03	8.78E−12	6.18E−10
		Mean	4.17E−08	1.79E−04	5.47E−02	9.15E−08	1.00E−12	−2.75E+03	1.35E−05	8.43E−05
		STD	1.32E−07	1.40E−04	5.92E−02	1.16E−07	2.68E−12	4.66E+02	4.23E−05	2.52E−04
Piecewise	CSWO 1	Best	9.29E−161	1.50E−05	**1.62E−06**	**0.00E+00**	**0.00E+00**	**−4.18E+03**	4.79E−79	2.05E−78
		Mean	**6.75E−142**	1.76E−04	2.09E−03	**0.00E+00**	**0.00E+00**	−3.40E+03	2.55E−66	1.99E−57
		STD	**2.11E−141**	1.52E−04	3.70E−03	**0.00E+00**	**0.00E+00**	8.45E+02	8.07E−66	6.31E−57
	CSWO 2	Best	5.85E−159	6.67E−06	3.61E−05	**0.00E+00**	**0.00E+00**	**−4.18E+03**	7.40E−81	1.04E−79
		Mean	2.95E−141	1.12E−04	6.93E−04	**0.00E+00**	**0.00E+00**	−3.66E+03	5.68E−54	4.72E−53
		STD	9.35E−141	9.67E−05	4.95E−04	**0.00E+00**	**0.00E+00**	8.16E+02	1.79E−53	1.49E−52
	CSWO 3	Best	1.07E−154	1.60E−05	1.04E−05	**0.00E+00**	**0.00E+00**	**−4.18E+03**	6.25E−81	6.19E−79
		Mean	3.83E−125	1.21E−04	7.81E−04	**0.00E+00**	**0.00E+00**	−3.89E+03	5.39E−66	6.43E−61
		STD	1.21E−124	9.55E−05	9.79E−04	**0.00E+00**	**0.00E+00**	6.14E+02	1.69E−65	1.84E−60
	CSWO 4	Best	7.21E−160	7.85E−06	2.81E−05	**0.00E+00**	**0.00E+00**	**−4.18E+03**	7.53E−80	1.96E−80
		Mean	9.23E−99	6.79E−05	1.26E−03	**0.00E+00**	9.81E−05	**−4.03E+03**	1.55E−39	4.22E−75
		STD	2.91E−98	7.46E−05	1.40E−03	**0.00E+00**	3.10E−04	**4.74E+02**	3.27E−39	1.31E−74

Map	Algorithms		Functions							
			F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
Tent	CSWO 1	Best	1.46E−165	1.33E−05	2.54E−06	**0.00E+00**	**0.00E+00**	**−4.18E+03**	2.58E−78	2.90E−80
		Mean	8.80E−104	1.13E−04	2.11E−03	**0.00E+00**	**0.00E+00**	−3.66E+03	9.15E−56	1.21E−71
		STD	2.78E−103	1.18E−04	4.10E−03	**0.00E+00**	**0.00E+00**	6.89E+02	2.89E−55	3.81E−71
	CSWO 2	Best	1.18E−155	1.22E−06	1.94E−06	**0.00E+00**	**0.00E+00**	**−4.18E+03**	2.35E−76	1.13E−79
		Mean	6.30E−101	1.56E−04	1.70E−03	**0.00E+00**	**0.00E+00**	−3.40E+03	2.55E−68	4.42E−56
		STD	1.80E−100	1.36E−04	3.46E−03	**0.00E+00**	**0.00E+00**	1.00E+03	4.57E−68	1.39E−55
	CSWO 3	Best	4.67E−160	**9.74E−07**	1.55E−05	**0.00E+00**	**0.00E+00**	**−4.18E+03**	5.50E−83	2.23E−80
		Mean	5.26E−106	**3.98E−05**	8.78E−04	**0.00E+00**	1.58E−05	−3.79E+03	1.72E−54	7.21E−73
		STD	1.66E−105	**3.93E−05**	1.68E−03	**0.00E+00**	5.01E−05	8.08E+02	3.91E−54	2.28E−72
	CSWO 4	Best	3.35E−143	1.45E−05	3.01E−06	**0.00E+00**	**0.00E+00**	**−4.18E+03**	6.17E−72	4.01E−73
		Mean	7.16E−65	9.70E−05	**3.94E−04**	**0.00E+00**	**0.00E+00**	−3.70E+03	1.72E−34	2.50E−66
		STD	2.07E−64	6.70E−05	4.95E−04	**0.00E+00**	**0.00E+00**	8.02E+02	5.44E−34	4.91E−66

Chaotic maps can improve the SWO algorithm by enhancing exploration and exploitation, but they come with several challenges. Using chaotic maps increases computational overhead and complexity, which can slow down the algorithm. Performance is sensitive to map selection and parameters, leading to unpredictability. Chaotic behavior can cause excessive exploitation, reducing population diversity and causing premature convergence. Multiple chaotic maps introduce implementation challenges and increase complexity. They can also exhibit noise, potentially destabilizing the optimization process. Careful tuning and empirical validation are needed to effectively integrate chaotic maps and benefit from their behavior.

The CSWO algorithm, inspired by spider wasps and enhanced with chaotic maps, is a powerful metaheuristic optimization method applicable to many real-world scenarios. In engineering, it optimizes structures like bridges, airplane wings, and mechanical systems for efficiency and material use. In energy systems, it improves renewable energy configurations and power grid operations. For transportation, it resolves vehicle routing issues and optimizes traffic signals to reduce congestion. In healthcare, CSWO enhances medical imaging and drug design. Financial firms use it for portfolio optimization and algorithmic trading. In machine learning, it aids feature selection and hyperparameter tuning. Environmental applications include water resource management and crop planning. Telecommunications benefit from improved network design, while robotics and aerospace use it for route planning and system optimization. Chemical engineering leverages CSWO for process and reactor design. In education, it enhances curriculum design and training programs. Overall, CSWO is a versatile tool for solving optimization challenges across

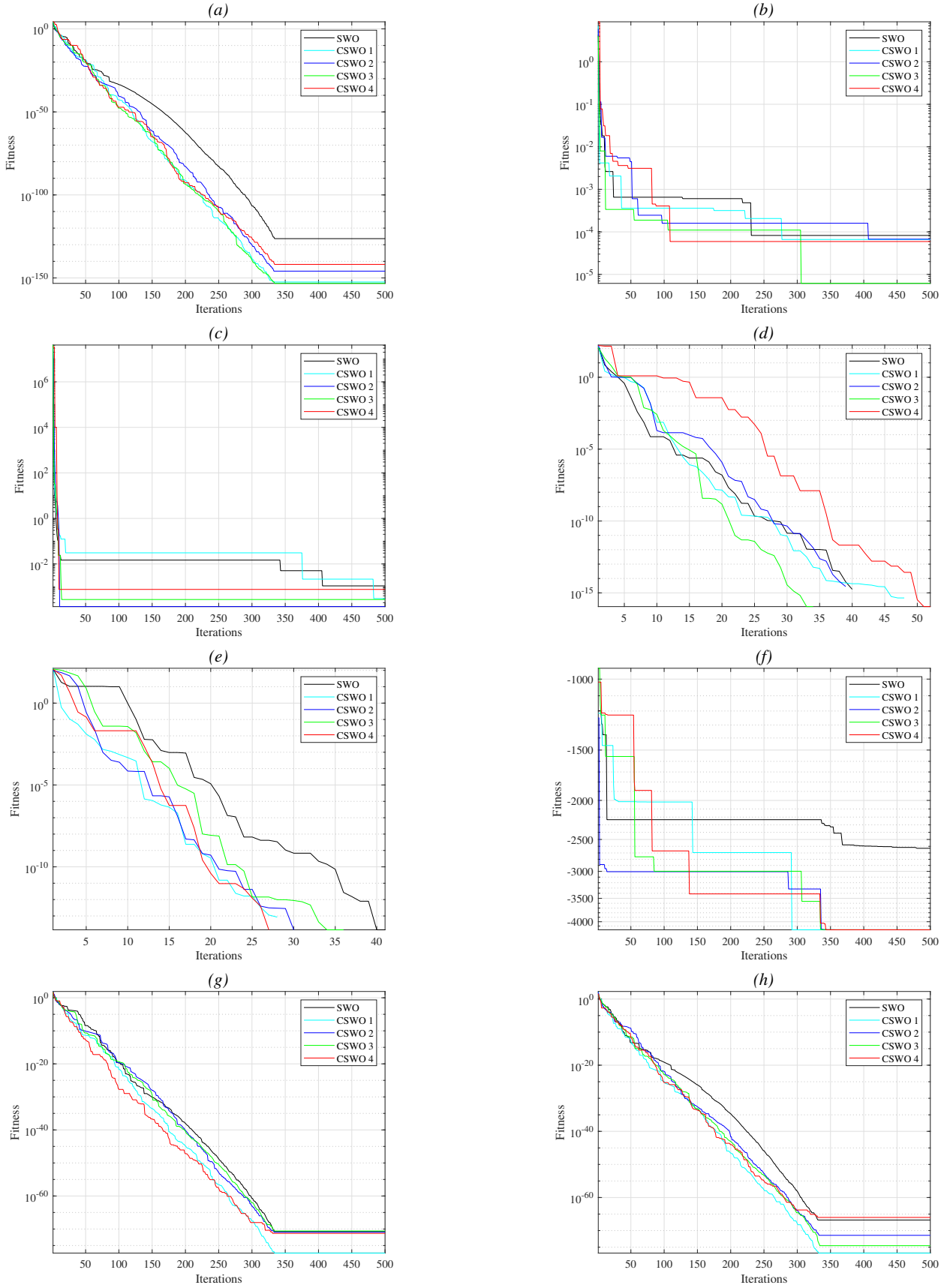

Figure 4. Convergence graph of each algorithm in solving: (a) F_1 Sphere; (b) F_2 Quartic noise; (c) F_3 Rosenbrock; (d) F_4 Griewank; (e) F_5 Rastrigin; (f) F_6 Schwefel; (g) F_7 Schwefel 2.21 and (h) F_8 Schwefel 2.22 benchmark functions

various domains, making it valuable for both researchers and practitioners.

Declarations

Use of AI tools

The authors declare that they have not used Artificial Intelligence (AI) tools in the creation of this article.

Data availability statement

No Data associated with the manuscript.

Ethical approval (optional)

The authors state that this research complies with ethical standards. This research does not involve either human participants or animals.

Consent for publication

Not applicable

Conflicts of interest

The authors declare that they have no conflict of interest.

Funding

No funding was received for this research.

Author's contributions

H.N.: Conceptualization, Methodology, Data Curation, Writing-Original Draft Preparation, Formal Analysis, Investigation, Software, Validation, Visualization. H.T.: Supervision, Conceptualization, Methodology, Investigation, Software, Formal Analysis, Writing Original Draft Preparation, Validation. All authors have read and agreed to the published version of the manuscript.

Acknowledgements

Not applicable

References

[1] Deepa, R. and Venkataraman, R. Enhancing Whale Optimization Algorithm with Levy Flight for coverage optimization in wireless sensor networks. *Computers & Electrical Engineering*, 94, 107359, (2021). [CrossRef]

[2] Kitayama, S., Arakawa, M. and Yamazaki, K. Differential evolution as the global optimization technique and its application to structural optimization. *Applied Soft Computing*, 11(4), 3792-3803, (2011). [CrossRef]

[3] Tomar, V., Bansal, M. and Singh, P. Metaheuristic algorithms for optimization: A brief review. *Engineering Proceedings*, 59(1), 238, (2024). [CrossRef]

[4] Alorf, A. A survey of recently developed metaheuristics and their comparative analysis. *Engineering Applications of Artificial Intelligence*, 117(A), 105622, (2023). [CrossRef]

[5] Rajwar, K., Deep, K. and Das, S. An exhaustive review of the metaheuristic algorithms for

search and optimization: taxonomy, applications, and open challenges. *Artificial Intelligence Review,* 56, 13187-13257, (2023). [CrossRef]

[6] Sowmya, R., Premkumar, M. and Jangir, P. Newton-Raphson-based optimizer: A new population-based metaheuristic algorithm for continuous optimization problems. *Engineering Applications of Artificial Intelligence,* 128, 107532, (2024). [CrossRef]

[7] Črepinšek, M., Liu, S.H. and Mernik, M. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR),* 45(3), 1-33, (2013). [CrossRef]

[8] Wolpert, D.H. and Macready, W.G. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation,* 1(1), 67-82, (1997). [CrossRef]

[9] Naik, P.A., Owolabi, K.M., Yavuz, M. and Zu, J. Chaotic dynamics of a fractional order HIV-1 model involving AIDS-related cancer cells. *Chaos, Solitons & Fractals,* 140, 110272, (2020). [CrossRef]

[10] Haneche, N. and Hamaizia, T. A three-dimensional discrete fractional-order HIV-1 model related to cancer cells, dynamical analysis and chaos control. *Mathematical Modelling and Numerical Simulation with Applications,* 4(3), 256-279, (2024). [CrossRef]

[11] Haneche, N. and Hamaizia, T. On the stability analysis for a three-dimensional fractional-order tümor system with obesity and immunosuppression. *Advances in Mathematical Sciences & Applications,* 33(2), p551, (2024). [CrossRef]

[12] Eskandari, Z., Naik, P.A. and Yavuz, M. Dynamical behaviors of a discrete-time prey-predator model with harvesting effect on the predator. *Journal of Applied Analysis and Computation,* 14(1), 283-297, (2024). [CrossRef]

[13] Joshi, H. and Jha, B.K. Chaos of calcium diffusion in Parkinson's infectiousdisease model and treatment mechanism via Hilfer fractional derivative. *Mathematical Modelling and Numerical Simulation with Applications,* 1(2), 84-94, (2021). [CrossRef]

[14] Hammouch, Z., Yavuz, M. and Özdemir, N. Numerical solutions and synchronization of a variable-order fractional chaotic system. *Mathematical Modelling and Numerical Simulation with Applications,* 1(1), 11-23, (2021). [CrossRef]

[15] Shahna, K.U. Novel chaos based cryptosystem using four-dimensional hyper chaotic map with efficient permutation and substitution techniques. *Chaos, Solitons & Fractals,* 170, 113383, (2023). [CrossRef]

[16] Nabil, H. and Tayeb, H. A secure communication scheme based on generalized modified projective synchronization of a new 4-D fractional-order hyperchaotic system. *Physica Scripta,* 99(9), 095203, (2024). [CrossRef]

[17] Nabil, H. and Tayeb, H. A fractional-order chaotic Lorenz-based chemical system: Dynamic investigation, complexity analysis, chaos synchronization, and its application to secure communication. *Chinese Physics B,* 33(12), 120503, (2024). [CrossRef]

[18] Kvasov, D.E. and Mukhametzhanov, M.S. Metaheuristic vs. deterministic global optimization algorithms: The univariate case. *Applied Mathematics and Computation,* 318, 245-259, (2018). [CrossRef]

[19] Radhika, S. and Chaparala, A. Optimization using evolutionary metaheuristic techniques: a brief review. *Brazilian Journal of Operations & Production Management,* 15(1), 44-53, (2018). [CrossRef]

[20] Sridharan, S., Subramanian, R.K. and Srirangan, A.K. Physics based meta heuristics in manufacturing. *Materials Today: Proceedings,* 39(1), 805-811, (2021). [CrossRef]

[21] Trojovskỳ, P. A new human-based metaheuristic algorithm for solving optimization problems based on preschool education. *Scientific Reports,* 13(1), 21472, (2023). [CrossRef]

[22] Xie, L., Han, T., Zhou, H., Zhang, Z.R., Han, B. and Tang, A. Tuna swarm optimization: A novel Swarm-Based metaheuristic algorithm for global optimization. *Computational Intelligence and Neuroscience,* 2021(1), 9210050, (2021). [CrossRef]

[23] Katoch, S., Chauhan, S.S. and Kumar, V. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications,* 80, 8091-8126, (2021). [CrossRef]

[24] Santosa, B. and Safitri, A.L. Biogeography-based optimization (BBO) algorithm for single machine total weighted tardiness problem (SMTWTP). *Procedia Manufacturing,* 4, 552-557, (2015). [CrossRef]

[25] Bai, H., Cao, Q. and An, S. Mind evolutionary algorithm optimization in the prediction of satellite clock bias using the back propagation neural network. *Scientific Reports,* 13, 2095, (2023). [CrossRef]

[26] Rashedi, E., Nezamabadi-Pour, H. and Saryazdi, S. GSA: a gravitational search algorithm. *Information Sciences,* 179(13), 2232-2248, (2009). [CrossRef]

[27] Azizi, M., Aickelin, U.A., Khorshidi, H. and Baghalzadeh Shishehgarkhaneh, M. Energy valley optimizer: a novel metaheuristic algorithm for global and engineering optimization. *Scientific Reports,* 13(1), 226, (2023). [CrossRef]

[28] Kumar, M., Kulkarni, A.J. and Satapathy, S.C. Socio evolution & learning optimization algorithm: A socio-inspired optimization methodology. *Future Generation Computer Systems,* 81, 252-272, (2018). [CrossRef]

[29] Kazemi, M.V. and Veysari, E.F. A new optimization algorithm inspired by the quest for the evolution of human society: Human felicity algorithm. *Expert Systems with Applications,* 193, 116468, (2022). [CrossRef]

[30] Bayzidi, H., Talatahari, S., Saraee, M. and Lamarche, C.P. Social network search for solving engineering optimization problems. *Computational Intelligence and Neuroscience,* 2021(1), 8548639, (2021). [CrossRef]

[31] Gad, A.G. Particle swarm optimization algorithm and its applications: a systematic review. *Archives of Computational Methods in Engineering,* 29(5), 2531-2561, (2022). [CrossRef]

[32] Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H. and Mirjalili, S.M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software,* 114, 163-191, (2017). [CrossRef]

[33] Abdollahzadeh, B., Gharehchopogh, F.S., Khodadadi, N. and Mirjalili, S. Mountain gazelle optimizer: a new nature-inspired metaheuristic algorithm for global optimization problems. *Advances in Engineering Software,* 174, 103282, (2022). [CrossRef]

[34] Dhiman, G. and Kumar, V. Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications. *Advances in Engineering Software,* 114, 48-70, (2017). [CrossRef]

[35] Afshar, A., Haddad, O.B., Marino, M.A. and Adams, B.G. Honey-bee mating optimization (HBMO) algorithm for optimal reservoir operation. *Journal of the Franklin Institute,* 344(5), 452-462, (2007). [CrossRef]

[36] Arora, S. and Singh, S. Butterfly optimization algorithm: a novel approach for global optimization. *Soft Computing,* 23, 715-734, (2019). [CrossRef]

[37] Mirjalili, S. The ant lion optimizer. *Advances in Engineering Software,* 83, 80-98, (2015). [CrossRef]

[38] Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M. and Chen, H. Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems,* 97, 849-872, (2019). [CrossRef]

[39] Gandomi, A.H., Yang, X.S., Alavi, A.H. and Talatahari, S. Bat algorithm for constrained optimization tasks. *Neural Computing and Applications,* 22, 1239-1255, (2013). [CrossRef]

[40] Pan, W.T. A new fruit fly optimization algorithm: taking the financial distress model as an example. *Knowledge-Based Systems,* 26, 69-74, (2012). [CrossRef]

[41] Mirjalili, S. and Lewis, A. The whale optimization algorithm. *Advances in Engineering Software,* 95, 51-67, (2016). [CrossRef]

[42] Saremi, S., Mirjalili, S. and Lewis, A. Grasshopper optimisation algorithm: theory and application. *Advances in Engineering Software,* 105, 30-47, (2017). [CrossRef]

[43] Abdollahzadeh, B., Soleimanian Gharehchopogh, F. and Mirjalili, S. Artificial gorilla troops optimizer: a new nature-inspired metaheuristic algorithm for global optimization problems. *International Journal of Intelligent Systems,* 36(10), 5887-5958, (2021). [CrossRef]

[44] Mirjalili, S., Mirjalili, S.M. and Lewis, A. Grey wolf optimizer. *Advances in Engineering Software,* 69, 46-61, (2014). [CrossRef]

[45] Faramarzi, A., Heidarinejad, M., Mirjalili, S. and Gandomi, A.H. Marine Predators Algorithm: A nature-inspired metaheuristic. *Expert Systems with Applications,* 152, 113377, (2020). [CrossRef]

[46] Mehta, P., Yildiz, B.S., Sait, S.M. and Yildiz, A.R. Hunger games search algorithm for global optimization of engineering design problems. *Materials Testing,* 64(4), 524-532, (2022). [CrossRef]

[47] Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A.A., Al-Qaness, M.A. and Gandomi, A.H. Aquila optimizer: a novel meta-heuristic optimization algorithm. *Computers & Industrial Engineering,* 157, 107250, (2021). [CrossRef]

[48] Morales-Castañeda, B., Zaldivar, D., Cuevas, E., Fausto, F. and Rodríguez, A. A better balance in metaheuristic algorithms: Does it exist? *Swarm and Evolutionary Computation,* 54, 100671, (2020). [CrossRef]

[49] Abdel-Basset, M., Mohamed, R., Jameel, M. and Abouhawwash, M. Spider wasp optimizer: A novel meta-heuristic optimization algorithm. *Artificial Intelligence Review,* 56, 11675-11738, (2023). [CrossRef]

[50] Kaur, G. and Arora, S. Chaotic whale optimization algorithm. *Journal of Computational Design and Engineering,* 5(3), 275-284, (2018). [CrossRef]

[51] Arora, S. and Anand, P. Chaotic grasshopper optimization algorithm for global optimization. *Neural Computing and Applications,* 31, 4385-4405, (2019). [CrossRef]

[52] Ismail, F.H., Houssein, E.H. and Hassanien, A.E. Chaotic bird swarm optimization algorithm. In Proceedings, *of the International Conference on Advanced Intelligent Systems and Informatics (AISI 2018),* pp. 294-303, Cairo, Egypt, (2018, August). [CrossRef]

[53] Kiani, F., Nematzadeh, S., Anka, F.A. and Findikli, M.A. Chaotic sand cat swarm optimization. *Mathematics,* 11(10), 2340, (2023). [CrossRef]

[54] Arora, S. and Singh, S. An improved butterfly optimization algorithm with chaos. *Journal of*

Intelligent & Fuzzy Systems, 32(1), 1079-1088, (2017). [CrossRef]

[55] Shinde, V., Jha, R. and Mishra, D.K. Improved Chaotic Sine Cosine Algorithm (ICSCA) for global optima. *International Journal of Information Technology*, 16, 245-260, (2024). [CrossRef]

[56] Hamaizia, T., Lozi, R. and Hamri, N.E. Fast chaotic optimization algorithm based on locally averaged strategy and multifold chaotic attractor. *Applied Mathematics and Computation*, 219(1), 188-196, (2012). [CrossRef]

[57] Feng, J., Zhang, J., Zhu, X. and Lian, W. A novel chaos optimization algorithm. *Multimedia Tools and Applications*, 76, 17405-17436, (2017). [CrossRef]

[58] Shayeghi, H., Shayanfar, H.A., Jalilzadeh, S. and Safari, A. Multi-machine power system stabilizers design using chaotic optimization algorithm. *Energy Conversion and Management*, 51(7), 1572-1580, (2010). [CrossRef]

[59] Cisternas-Caneo, F., Crawford, B., Soto, R., Giachetti, G., Paz, A. and Peña Fritz, A. Chaotic binarization schemes for solving combinatorial optimization problems using continuous metaheuristics. *Mathematics*, 12(2), 262, (2024). [CrossRef]

[60] Fiedler, R., Hetzler, H. and Bäuerle, S. Efficient numerical calculation of Lyapunov-exponents and stability assessment for quasi-periodic motions in nonlinear systems. *Nonlinear Dynamics*, 112, 8299-8327, (2024). [CrossRef]

[61] Rayor, L.S. Attack strategies of predatory wasps (Hymenoptera: Pompilidae; Sphecidae) on colonial orb web-building spiders (Araneidae: Metepeira incrassata). *Journal of the Kansas Entomological Society*, 69(4), 67-75, (1996).

[62] Liang, J.J., Qin, A.K., Suganthan, P.N. and Baskar, S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, 10(3), 281-295, (2006). [CrossRef]

Mathematical Modelling and Numerical Simulation with Applications (MMNSA)
(https://dergipark.org.tr/en/pub/mmnsa)

How to cite this article: Nabil, H. & Tayeb, H. (2025). Effect of chaos on the performance of spider wasp meta-heuristic optimization algorithm for high-dimensional optimization problems. *Mathematical Modelling and Numerical Simulation with Applications*, 5(1), 143-171. https://doi.org/10.53391/mmnsa.1571964