



Impact of popular chaotic maps on population initialization of Sailfish Optimizer Algorithm – An experimental study

Saboohi Naeem Ahmed^{a,b}, Adnan A. Siddiqui^a,
M. R. Tanweer^c, and Narmeen Bawany^d

^aGraduate School of Engineering and IT, Hamdard University, Madinat al-Hikma Karachi, Pakistan

^bKarachi Institute of Economics and Technology, Block F, North Nazimabad, Karachi, 74600, Pakistan

^cKarachi School of Business and Leadership, Dawood Society, Karachi, 74800, Pakistan

^dJinnah University for Women, 5C Nazimabad, Karachi, 74600, Pakistan

✉ saboohinaeem@gmail.com, saboohi.naeem@hamdard.edu, adnan.siddiqui@hamdard.edu.pk, rizwan.tanweer@ksbl.edu.pk, narmeen.bawany@juw.edu.pk

KEYWORDS

sailfish optimizer;
chaotic maps;
chaos; improve
initial population;
swarm intelligence

ABSTRACT

Initial population in a swarm intelligence-based optimization algorithm plays an important role in avoiding local optima, early convergence, and global space exploration. If initial values provide proper coverage of the search space, then it may quickly lead to the optimal solution, whereas limited coverage may stick an algorithm into local optima. Several techniques are presented by researchers to properly initialize the population instead of random numbers. Chaos theory is a well-known concept utilized in designing an improved variant of various existing nature-inspired optimization algorithms. However, its impact on the initial population of a metaheuristic was never thoroughly studied before. This study investigates the impact of ten chaotic maps on the initial population of a sailfish optimizer algorithm (SFO). SFO is a recent metaheuristic algorithm which mimics the hunting behavior of sailfish. Ten chaotic maps have been used to initialize sailfish and sardine populations. All techniques (with or without chaotic maps) are tested with twenty-three classical benchmark functions. The numerical results, ranking and statistical analysis show that chaotic maps are useful in avoiding local optima and are effective for improving the global search capabilities of a metaheuristic.

1. Introduction

Swarm Intelligence (SI) based optimization algorithms are a broad research field and have received much attention after the inception of Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995). PSO is a nature-inspired optimization algorithm simulating the behavior of a swarm of particles, where each particle represents a potential solution to a problem. These particles move through a search space, adjusting their positions based on their own experiences and their neighbors. All such algorithms are generally population-based (Boussaïd et al., 2013), which involves the interaction and cooperation of multiple agents or individuals (e.g., particles, birds, ants, bees) within a population. These agents collectively work together to explore the search space and find optimal or near-optimal solutions to a problem.

The realm of literature demonstrates significant effort in the development of SI based algorithms inspired by various real-world principles, for instance, Artificial Bee Colony (ABC) (Karaboga & Basturk, 2007), Ant Colony Optimization (ACO) (Dorigo & Di Caro, 1999), Firefly Algorithm (FA) (Yang, 2009), Bat Algorithm (BA) (Yang & Gandomi, 2012), Cuckoo Search (CS) (Yang & Deb, 2009), Whale Optimization algorithm (WOA) (Mirjalili & Lewis, 2016), and Sailfish Optimizer Algorithm (SFO) (Shadravan et al., 2019). The growing interest of the research community in this domain can be attributed to the «No Free Lunch (NFL) Theorem» (Wolpert & Macready, 1997) which states that there is no one-size-fits-all algorithm that works best for all possible problem instances. Therefore, it is possible that an algorithm works better for one set of problems and may not be suitable enough to solve the problems in another set of functions. NFL theory promotes the development of new algorithms or even modifications in the existing techniques to find a better version of the same algorithm. This modification process typically involves changing input parameters (population, convergence factor or any other internal parameter), changing position update equations or modifying the values on which an algorithm operates (Mahmood et al., n.d.).

In SI based algorithms, each particle in the population typically represents a potential solution or a decision variable within a solution. These particles communicate, cooperate, and adjust their positions or behaviors based on certain rules, principles, or heuristics. Various studies have shown that SI based algorithms are sensitive to initial conditions, including initial population and parameter settings (Tizhoosh, 2005; Elaziz & Mirjalili, 2019). On the other hand, some algorithms are not sensitive to the initial conditions (Yang et al., 2018), for example, Differential Evolution (DE) (Storn & Price, 1997). Nevertheless, initial population plays an important role in the fast convergence of SI based algorithms and promotes the exploration of a wider search space. Studies have examined the influence of the initial population on the effectiveness of a metaheuristic algorithm.

Researchers have long been studying the impact of the initial population by using techniques other than pseudo random numbers. Maaranen et al. (2007) studied the effects of quasi random sequence and spatial point processes on the initial population of GA. Multiple probability distributions were used by Li et al. (2020) and J. O. Agushaka & Ezugwu (2021) to generate the initial populations of different metaheuristic algorithms. Research studies used low discrepancy sequences to generate the initial population (Agushaka & Ezugwu, 2020; Ashraf et al., 2021). Quasi random sequences were used to generate the initial population in Slezhkin & Hodashinsky (2021). All aforementioned techniques showed significant improvement in the performance of algorithms used.

Chaos exhibits characteristics commonly found in nonlinear systems. In recent times, chaos theory has been regularly integrated into certain metaheuristics to mitigate the problem of premature convergence. Hefny & Azab (2010), Ma (2012) and El Ghouate et al. (2024) developed a chaotic PSO, a chaotic GA and chaotic Kepler algorithm, respectively, by embedding a logistic map into the search

equation and the initial particles. Chaotic WOA was designed by Kaur & Arora (2018) and Sayed et al. (2018) using chaotic maps. A chaotic GWO was developed by Lu et al. (2020) to generate internal parameters through chaotic maps. In a similar way, various recent research studies have also utilized chaotic maps to modify search strategy, initial population or internal parameters to improve their techniques (Abdel-Salam et al., 2024; Luo et al., 2024; Mohamed et al., 2024).

Drawing inspiration from the effective fusion of chaotic-based strategies with metaheuristics, the chaotic maps based SFO is developed in this work. SFO is a recent SI based optimization algorithm that mimics the hunting behavior of sailfish. It uses two sets of population, sailfish and sardines. The sailfish set is the candidate solution, and a sardine population is used to increase the diversity of the search space. In this study, both populations (sailfish and sardines) are initialized using ten common chaotic maps and compared with the original SFO in which the population is randomly generated.

To the best of the authors' knowledge, no prior research has explored the combination of SFO with various chaotic maps in the context of population initialization. In Y. Zhang & Mo (2022) only tent map was used to initialize the population and it was combined with other strategies to prepare a new variant of SFO. The primary objective of our work is to initialize particles of SFO with ten chaotic maps and study its effects on the diversity of the search space. Experiments and statistical findings indicate that the proposed approach is promising for SFO to avoid local optima with better exploration of the search space.

The main contributions of this research are summarized below:

- A comprehensive survey of population initialization methods and applications of chaos theory in metaheuristic algorithms.
- Experiments aimed at improving the diversity of the search space of the SFO algorithm by employing ten popular chaotic maps.
- An evaluation of the performance of chaotic SFO using classical benchmark functions.
- A ranking of the techniques applied in this work according to mean and best performance.
- A statistical analysis validating the significance of chaotic map application for population initialization as compared to the random initial values.

The rest of the paper is organized as follows. Section 2 presents the literature review. Section 3 provides a brief overview of the SFO algorithm. The proposed methodology of chaotic SFO is presented in Section 4. Experimental simulation, results, analysis, and discussion are provided in Section 5. Section 6 concludes the research and presents future research directions.

2. Literature Review

In this section we first review the studies that have examined the impact of the initial population and then we present the literature related to chaotic sequences and other population initialization methods. Chaotic sequences have also been integrated into the internal parameters, or search equations of various metaheuristic algorithms, which are reviewed at the end. Population initialization plays a critical role in swarm and evolutionary optimization algorithms specifically due to the lack of diversity within the population that leads to premature convergence and a tendency to get stuck in local optima. Very few studies have investigated the impact of the initial population on the performance of a metaheuristic algorithm in general and little work is devoted to testing the impact of chaotic maps on the initial population.

Traditionally, the initial population of any optimization algorithm was generated with random distributions (normal or uniform). Researchers kept working on improvements by enhancing internal parameters or search equations until Maaranen et al. (2004) presented series of concepts in which the genetic algorithm (GA)'s initial population was generated with quasi-random numbers. Similarly, the study of Maaranen et al. (2007) examined the impact of generating the initial population using different point generators. Both studies successfully concluded with the statement that the initial population is meaningful and has an impact on optimal value generation. To enhance the effectiveness of the KH algorithm, the study of O. J. Agushaka & Ezugwu (2020) aimed to explore the impact of employing three low-discrepancy sequences, namely, the Faure sequence, Sobol sequence, and Van der Corput sequence as initializations for the KH algorithm. A more detailed and comprehensive approach was presented by Li et al. (2020) where Latin hypercube sampling (LHS) and seven distributions (beta, uniform, normal, Logarithmic normal, exponential, Rayleigh, Weibull distributions) were applied to generate initial solutions of differential evolution (DE), particle swarm optimization (PSO), cuckoo search (CS), artificial bee colony (ABC) and genetic algorithm (GA).

Several mathematical techniques and sequences are available to generate uniform random numbers besides pseudo random numbers, researchers have applied these techniques on the initial population of an optimization algorithm to study their impact on convergence (Agushaka & Ezugwu, 2022). Slezkin & Hodashinsky (2021) examined various population initialization techniques for the swallow swarm algorithm and contrasted them in the context of optimizing fuzzy classifier parameters. The researchers explored methods that rely on quasi-random sequences, chaotic maps, and random value distributions as potential approaches to addressing this challenge. The work presented by J. O. Agushaka & Ezugwu (2021) delves into the examination and validation of various initialization methods (beta and uniform distributions, Rayleigh distribution, Latin hypercube sampling, and Sobol sequence) for the arithmetic optimization algorithm (Abualigah et al., 2021). Notably, it was found that the Beta distribution consistently outperformed the random number distribution across all scenarios. Additionally, the performance of initialization techniques such as uniform distribution, Rayleigh distribution, Latin hypercube sampling, and Sobol low-discrepancy sequence proved to be quite competitive when compared to the random number initialization method. A summary of these methods is presented in Table 1.

Table 1. Methods to generate initial population

Reference	Methods to Generate Initial Population
(Slezkin & Hodashinsky, 2021)	Quasi Random sequences
(Agushaka & Ezugwu, 2020)	Low discrepancy sequences
(Ruiye et al., 2018)	Latin hyper cube sampling
(Huang et al., 2019)	Opposition based learning (OBL)
(Elaziz & Mirjalili, 2019)	Differential evolution Chaotic map / Opposition based learning
(Chen et al., 2020) (Yin et al., 2020)	Logistic chaotic map
(Chen, 2020)	Bernoulli shift map
(Fan et al., 2021)	Tent chaotic map
(Zhang & Liu, 2019)	Good point set method
(Jin et al., 2021)	Random gaussian distribution
(Jianhao et al., 2021)	Chaotic map / Opposition based learning
(Adamuthe & Nitave, 2018)	Partially feasible solution with memory

More recently, novel methods have been proposed to generate initial population to better increase the diversity of the population. The research by Bangyal et al. (2021) introduced three novel low-discrepancy sequences, namely the WELL sequence, Knuth sequence, and Torus sequence, for the purpose of initializing the population within the search space. A novel strategy for initializing the population was devised by Deng et al. (2015) to enhance the performance of the genetic algorithm when tackling the traveling salesman problem (TSP). The authors leveraged the principles of the k-means algorithm to introduce a method for reconfiguring the travel route by re-establishing connections within each cluster. In another study, by Alkafaween et al. (2021), a new approach was put forward for the initial population seeding for the same TSP problem, known as the Multi Linear Regression Based Technique (MLRBT). This method partitions a given large-scale TSP problem into smaller sub-problems and applies the technique iteratively until the sub-problem size becomes sufficiently small.

Chaotic maps have been integrated into internal parameters and search strategy of several metaheuristic algorithms to improve the performance and global search behavior of an algorithm. In the study of Tang et al. (2018) ten chaotic maps were integrated into PSO and Krill Herd algorithm (Gandomi & Alavi, 2012) to improve the convergence capability and diversity of the solution; statistical test results proved significant improvement. Chaotic lightning search algorithm was presented by Ouertani et al. (2021), where all random values in the base algorithm were replaced with eleven chaotic maps and results were compared with the original version of LSA with random values—significant improvement was observed in the chaotic versions. Similarly, many chaotic versions of other metaheuristic algorithms are presented in Table 2.

Table 2. Metaheuristics using chaotic maps as improvement strategy

Algorithm Name and Reference	Chaotic Map	Applied to
Chaos Genetic Algorithm (Javidi & Hosseinpourfard, 2015)	Logistic and Tent	All random values
Chaotic Bat Algorithm (Gandomi & Yang, 2014)	Twelve maps	All internal parameters
Chaotic Cuckoo Search Algorithm (Wang & Zhong, 2015)	Twelve maps	Scaling factor and probability
Chaotic CS (Wang et al., 2016)	Twelve maps	Step size
Chaos Infused Moth Flame Optimizer (Gupta et al., 2022)	Eight maps	Distance update equation
Chaotic Sand Swarm Optimization (Kiani et al., 2023)	Twelve maps	Internal parameters
Improved ABC (Gao et al., 2011)	Henon Map	Initial population, search strategy
Chaotic Bee Colony (Alatas, 2010a)	Seven maps	Random numbers and search strategy
Improved Salp Swarm Algorithm (Tawhid & Ibrahim, 2022)	Tent map	Search Equation
Chaos based Grey Wolf Optimizer (Lu et al., 2020)	Eleven maps	Internal parameters A and C
Firefly algorithm with chaos (Gandomi et al., 2013)	Twelve maps	Light absorption and attractiveness coefficients
Chaotic Harmony Search (Alatas, 2010b)	Seven maps	Random parameters
Modified Dandelion Optimizer (Aribowo et al., 2023)	Logistic map	Random parameter

3. Sailfish Optimizer Algorithm

The Sailfish Optimizer (SFO) is a recent nature-inspired optimization algorithm that draws inspiration from the hunting behavior of sailfish, which are among the fastest marine predators (Shadravan et al., 2019). This algorithm, like other metaheuristic algorithms, is designed to solve optimization problems. The algorithm models the interactions between prey (sardines) and predators (sailfish) in the ocean. Sardines represent potential solutions or candidates in the search space, while sailfish represent the optimization process. Sailfish in the natural world often use group hunting strategies to capture prey. Similarly, the SFO employs a group-based approach, where a population of sailfish collaboratively searches for optimal solutions. Sailfish are known for their incredible speed and agility. In the algorithm, this speed and direction are used to guide the search. Sailfish adjust their positions and directions based on the quality of solutions (sardines) they encounter.

3.1. Mathematical Model of SFO

In the SFO algorithm, the following equation is used to update sailfish position:

$$X_{\text{new_SF}}^i = X_{\text{elite_SF}}^i - \lambda_i \times \left(\text{rand}(0,1) \times \left(\frac{X_{\text{elite_SF}}^i + X_{\text{injuredS}}^i}{i} \right) - X_{\text{old_SF}}^i \right) \quad (1)$$

Where $X_{\text{elite_SF}}^i$ is the best position of sailfish, X_{injuredS}^i is the best position of sardine, $X_{\text{old_SF}}^i$ is the current position of sailfish, rand is the random number from 0 to 1, λ_i is the coefficient at the i^{th} iteration as follows:

$$\lambda_i = 2 \times \text{rand}(0,1) \times PD - PD \quad (2)$$

Where PD represents prey density showing the number of sardines in each iteration. Note that sardines are decreasing in each iteration. Prey density is calculated as follows:

$$PD = 1 - \left(\frac{N_{\text{SF}}}{N_{\text{SF}} + N_{\text{S}}} \right) \quad (3)$$

N_{SF} and N_{S} are the numbers of sailfish and sardines, respectively. Also, N_{SF} is defined as $N_{\text{S}} \times \text{PP}$, where PP is the percentage of sardine population that initially forms the sailfish population. Sardine position is calculated by the following equation:

$$X_{\text{new_S}}^i = r \times (X_{\text{elite_SF}}^i - X_{\text{old_S}}^i + AP) \quad (4)$$

Where $X_{\text{old_S}}^i$ is the current position of the sardine and $X_{\text{elite_SF}}^i$ is the best sailfish position, r is the random number between 0 to 1 and AP is the attack power of sailfish in each iteration, calculated as follows:

$$AP = A \times (1 - (2 \times \text{Itr} \times \epsilon)) \quad (5)$$

Where Itr is the iteration number, and A and ϵ are the coefficients for decreasing attack power linearly from A to 0. In the final stages of the hunt, when a sardine becomes fitter than sailfish it is inserted in the sailfish population by the following equation:

$$X_{SF}^i = X_s^i \quad \text{if } f(S_i) < f(SF_i) \quad (6)$$

The complete SFO algorithm is provided in Algorithm 1.

Algorithm 1. Sailfish Optimizer Algorithm (Shadravan et al., 2019)

```

Initialize the population of sailfish and sardines randomly, based on PP = 0.3
Initialize parameters ( $A = 4$  and  $\epsilon = 0.001$ ).
Compute the fitness of sailfish and sardines.
Find the best sailfish and injured sardine
While the termination conditions are not satisfied
  for each sailfish
    Calculate  $\lambda_i$  using (2)
    Update the position of sailfish using (1).
  end for

  Calculate AttackPower using (5)
  If AttackPower < 0.5
    Calculate  $\alpha = N_s \times AP$ 
    Calculate  $\beta = d_i \times AP$ 

    Select a set of sardines based on the value of  $\alpha$  and  $\beta$ 
    Update the position of selected sardine by (4)
  else
    Update the position of all sardines by (4)
  end if
  Calculate the fitness of all sardines.
  If there is a better solution in sardine population
    Replace a sailfish with injured Sardine using (6)
    Remove the hunted sardine from the population.
    Update the best sailfish and the best sardine.
  end if
end while
Return best sailfish

```

ipt for review they also accept an obligation to maintain confidentiality of the manuscript's contents; this obligation ends only when the manuscript becomes lawfully available to them through another channel without an obligation of confidentiality.

4. Proposed Chaotic SFO

A chaotic system is a nonlinear dynamic system known for its inherent unpredictability, lack of periodicity, wide-ranging spectral characteristics, ergodic behavior, and extreme sensitivity to initial conditions. Researchers widely use chaotic maps to improve convergence and global optimization capabilities. Using a chaotic map instead of pseudo random numbers for generating an initial population in optimization offers several advantages that contribute to improved efficiency and effectiveness in finding optimal solutions, including enhanced exploration, faster convergence, and deterministic and reproducible initialization. Ten popular chaotic maps with mathematical representation are listed in Table 3, where k is the iteration number and z_k is the k th chaotic number. The range of values generated by these maps is shown graphically in Figure 1 (Saremi et al., 2014).

Table 3. Ten chaotic Maps with mathematical formulation

No	Name	Mathematical representation
1	Chebyshev	$z_{k+1} = \cos(a \cdot \cos^{-1}(z_k)), a=4$
2	Circle map	$z_{k+1} = z_k + b - (a/2\pi) \sin(2\pi k) \bmod(1), a=0.5, b=0.2$
3	Gaussian map	$z_{k+1} = \begin{cases} 0, z_k = 0 \\ 1/z_k \bmod(1), \text{otherwise} \end{cases}$
4	Iterative map	$z_{k+1} = \sin\left(\frac{a\pi}{z_k}\right), a=0.7$
5	Logistic map	$z_{k+1} = az_k(1-z_k), a=4$
6	Piecewise map	$x_{k+1} = \begin{cases} \frac{z_k}{P} & 0 \leq z_k < P \\ \frac{z_k - P}{0.5 - P} & P \leq z_k < 0.5 \\ \frac{1 - P - z_k}{0.5 - P} & 0.5 \leq z_k < 1 - P \\ \frac{1 - z_k}{P} & 1 - P \leq z_k < 1 \end{cases}$
7	Sine map	$z_{k+1} = \frac{a}{4} \sin(\pi z_k), 0 < a \leq 4$
8	Singer	$z_{k+1} = \mu(7.86Z_k - 23.31z_k^2 + 28.75z_k^3 - 13.30287z_k^4)$
9	Sinusoidal	$z_{k+1} = az_k^2 \sin(\pi z_k)$
10	Tent	$z_{k+1} = \begin{cases} z_k/0.7, z_k < 0.7 \\ 10(1-z_k)/3, z_k \geq 0.7 \end{cases}$

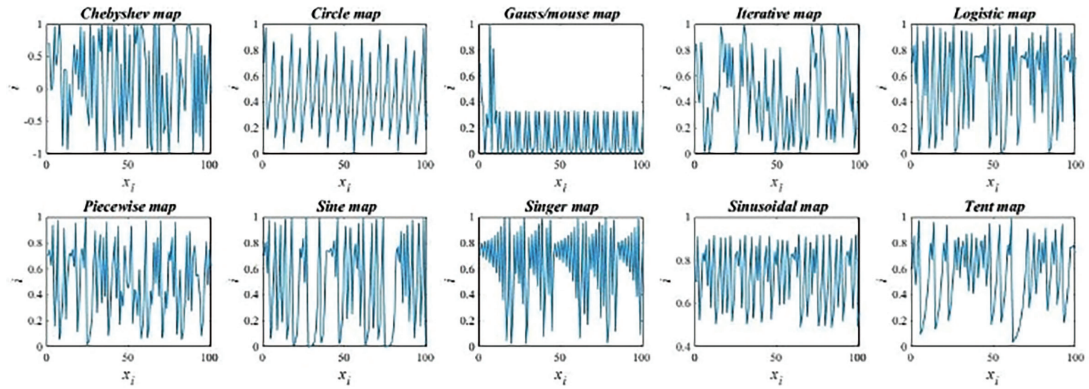


Figure 1. Range of random values generated by ten chaotic maps illustrating the search space coverage (Saremi et al., 2014)

The initial population in an optimization process plays a crucial role in determining the performance and behavior of the optimization algorithm. For this study, the initial population of SFO is generated with the given chaotic maps to give better coverage of the search space. The pseudocode of the algorithm to generate chaotic population is given in Table 5, where Dim are the dimensions, NP is number of sailfish/ sardines in the population, Min and Max define the scope of values, and C is the chaotic number.

Algorithm 2. Pseudocode to generate chaotic population
Initialize Dim and NP, Range = Min, Max. Generate chaotic numbers as C(Dim*NSF) For i = 1 to NP For j = 1 to Dim $SF_i.Position_j = C(((i-1) * Dim) + j) * (Max - Min) + Min$ End for End For

The rest of the process of the SFO algorithm is the same as mentioned in the previous section. For the upcoming experiments section, a naming convention is followed. SFO with pseudo random numbers is referred to as SFO while chaotic versions are SFO_M1, SFO_M2, SFO_M3, SFO_M4, SFO_M5, SFO_M6, SFO_M7, SFO_M8, SFO_M9, SFO_M10, where M1, M2 and so on reflect the map numbers already mentioned in Table 4. For example, SFO_M5 refers to the SFO initialized with logistic map. This way we have eleven algorithms (techniques).

5. Experimental Simulation

This section describes the experiments, discusses the overall results, provides a ranking of the techniques and outlines statistical testing.

5.1. Experimental Setup

For programming and performing experiments, MATLAB 2019b was used. The operating system was Microsoft Windows 10. System configuration was Intel(R) Core (TM) i5-5200U CPU @ 2.20 GHz with 8 GB RAM. Experiments were performed using twenty-three classical benchmark functions. These functions are used for testing exploration and exploitation capabilities and are widely used by researchers (Molga & Smutnicki, 2005; Yao et al., 1999; Digalakis & Margaritis, 2001). These can be divided into three groups: unimodal, multimodal, and fixed dimension multimodal, and are reported in Table 4, 5 and 6, where Dim is the dimension, Range indicates the boundary of the search space and $f_{\min}$ is the optimal value. Each algorithm was run 25 times with 500 iterations in each run. SFO parameter settings were the same for all eleven techniques, where the initial attack value A was 4.0, attack decline factor ϵ was 0.001, and the percentage of sardine population (PP) that formed the initial sailfish population was 0.3.

Table 4. Unimodal benchmark problems

Name	Mathematical Model	Dim	Range	$F_{\min}$
Sphere	$F_1(x) = \sum_{i=1}^n x_i^2$	30	[-100,100]	0
Schwefel's problem 2.22	$F_2(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	30	[-10,10]	0
Schwefel's problem 1.2	$F_3(x) = \sum_{i=1}^n \left(\sum_{j=1}^n x_j \right)^2$	30	[-100,100]	0
Schwefel's problem 2.21	$F_4(x) = \max\{ x_i , 1 \leq i \leq n\}$	30	[-100,100]	0
Rosenbrock	$F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30,30]	0
Step	$F_6(x) = \sum_{i=1}^n (x_i + 0.5)^2$	30	[-100,100]	0
Noise	$F_7(x) = \sum_{i=1}^n ix_i^4 + \text{random}(0,1)$	30	[-1.28,1.28]	0

Table 5. High Dimensional Multimodal Test Functions

Name	Mathematical Model	Dim	Range	$F_{\min}$
Generalized Schwefel	$F_8 = -\sum_{i=1}^n \left(x_i \sin(\sqrt{ x_i }) \right)$	30	[-500, 500]	$-418.9829 \times \text{Dim}$
Rastrigin	$F_9(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$	30	[-5.12, 5.12]	0

Name	Mathematical Model	Dim	Range	$F_{\min}$
Ackley	$F_{10}(x)$ $= -20 \exp \left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} \right)$ $- \exp \left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i) \right) + 20 + e$	30	$[-32, 32]$	0
Griewank	$F_{11}(x) = \frac{1}{400} \sum_i^n x_i^2 - \prod_{i=1}^n \cos \left(\frac{x_i}{\sqrt{i}} \right) + 1$	30	$[-600, 600]$	0
Generalized Penalized function 1	$F_{12}(x)$ $= \frac{\pi}{n} \{ 10 \sin(\pi y_1)$ $+ \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + \sin^2(3\pi y_{i+1})]$ $+ (y_n - 1)^2 \} + \sum_{i=1}^n u(x_i, 5, 100, 4)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \leq x_i \leq a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30	$[-50, 50]$	0
Generalized Penalized function 2	$F_{13}(x)$ $= 0.1 \{ \sin^2(3\pi x_1)$ $+ \sum_{i=1}^n (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)]$ $+ (x_n - 1)^2 [1 + \sin^2(2\pi x_n)]$ $+ \sum_{i=1}^n u(x_i, 5, 100, 4) \}$	30	$[-50, 50]$	0

Table 6. Fixed Dimension Multimodal Functions

Name	Test Functions	Dim	Range	$F_{\min}$
Shekel's Foxholes function	$F_{14}(x)$ $= \left(\frac{1}{500} + \sum_{i=1}^{25} \frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})} \right)$	2	$[-65, 65]$	0.998004
Kowalik's function	$F_{15}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	4	$[-5, 5]$	0.00030
Six-hump camel back	$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2$ $- 4x_2^2 + 4x_2^4$	2	$[-5, 5]$	-1.0316

(continued)

Table 6. Fixed Dimension Multimodal Functions (continued)

Name	Test Functions	Dim	Range	$F_{\min}$
Branin	$F_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10$	2	$[-5, 5]$	0.39788
Goldstein-Price function	$F_{18}(x) = [1 + (1 + x_1 + x_2)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$[-2, 2]$	3
Hartman 1	$F_{19}(x) = -\sum_{i=1}^4 c_i \exp \left[-\sum_{j=1}^3 a_{ij} (x_j - p_{ij})^2 \right]$	3	$[0, 1]$	-3.86
Hartman 2	$F_{20}(x) = -\sum_{i=1}^4 c_i \exp \left[-\sum_{j=1}^6 a_{ij} (x_j - p_{ij})^2 \right]$	6	$[0, 1]$	-3.32
Shekel 1	$F_{21}(x) = -\sum_{i=1}^5 \left[(x - a_i)(x - a_i)^T + c_i \right]^{-1}$	4	$[0, 10]$	-10.1532
Shekel 2	$F_{22}(x) = -\sum_{i=1}^7 \left[(x - a_i)(x - a_i)^T + c_i \right]^{-1}$	4	$[0, 10]$	-10.4028
Shekel 3	$F_{23}(x) = -\sum_{i=1}^{10} \left[(x - a_i)(x - a_i)^T + c_i \right]^{-1}$	4	$[0, 10]$	-10.5363

5.2. Results and Discussion

As discussed in the previous sections, results for basic Sailfish Optimizer (SFO) with random population and SFO with chaotic population (for ten chaotic maps) have been generated on twenty-three classical benchmark functions. Statistical results (mean, best, median, standard deviation) of all eleven methods on benchmark functions are listed in Tables 9, 10, 11 and 12. A complete ranking of each algorithm in all twenty-three functions, based on the mean and best values, is given in Tables 13 and 14, respectively. Then, in accordance with the average rank of each algorithm, as shown in Tables 13 and 14, a final rank is assigned in Table 15 to each technique.

5.2.1. Analysis of Unimodal Benchmark Functions

Unimodal benchmark functions (F1-F7) have a single global minimum, making them mathematically simpler to define and analyze. They also provide a baseline for evaluating the efficiency and effectiveness of optimization algorithms and measure how close an algorithm can get to the optimal solution. In other words, they are used to test the exploitation capabilities of an algorithm; how well an optimization converges to the optimum point. In all unimodal functions except F3, the chaotic population version shows best results for mean values, however in each function winning map is different (see Table 7). If we look at best values for unimodal functions then it is evident that SFO_M8 has best

results for F1, F2, while SFO_M1 achieved best results for F3, F5, F6, and SFO_M6 and SFO_M3 performed best in F4 and F7, respectively. It means that the convergence of SFO is improved by using chaotic population instead of the random one. However, it is important to find out the significance of the improved results, for this purpose statistical tests have been performed and are discussed in the upcoming subsections.

Table 7. Mean, median, standard deviation and best values in multimodal functions (F1- F7) for all techniques

	F1				F2			
	Mean	Best	Median	Std. Dev	Mean	Best	Median	Std. Dev
SFO	1.18E-15	5.37665E-20	6.16E-16	1.63E-15	1.28E-07	3.70256E-09	6.93E-08	1.37E-07
SFO_M1	1.12E-15	1.32181E-19	1.66E-16	1.85E-15	1.05E-07	6.52022E-09	7.68E-08	8.26E-08
SFO_M2	6.71E-16	5.40648E-20	1.67E-16	1.22E-15	9.4E-08	2.83617E-09	6.04E-08	1.1E-07
SFO_M3	7.26E-16	1.055E-17	2.33E-16	1.45E-15	1.24E-07	6.21222E-09	8.99E-08	1.12E-07
SFO_M4	7.99E-16	5.33155E-20	6.3E-17	1.88E-15	1.31E-07	2.9813E-09	5.21E-08	1.79E-07
SFO_M5	1.28E-15	1.32231E-19	3.94E-16	2.05E-15	1.41E-07	1.15783E-09	8.15E-08	1.55E-07
SFO_M6	2.01E-15	4.53023E-18	3.93E-16	5.98E-15	1.6E-07	1.11686E-08	1.14E-07	1.43E-07
SFO_M7	1.91E-15	1.82984E-18	3.77E-16	3.99E-15	1.57E-07	1.48719E-09	1.01E-07	1.63E-07
SFO_M8	2.15E-15	2.90708E-21	5.74E-16	2.79E-15	1.07E-07	4.79213E-10	5.49E-08	1.19E-07
SFO_M9	2.2E-15	1.35757E-18	4.2E-16	3.32E-15	7.68E-08	5.33519E-09	6.46E-08	6.04E-08
SFO_M10	1.25E-15	6.93643E-20	3.14E-16	1.95E-15	1.26E-07	1.63021E-08	1.26E-07	8.66E-08
	F3				F4			
	Mean	Best	Median	Std. Dev	Mean	Best	Median	Std. Dev
SFO	1.77E-13	3.6067E-16	2.84E-14	2.98E-13	5.57E-09	3.86173E-10	4.65E-09	4.69E-09
SFO_M1	3.21E-13	2.16503E-19	4.98E-14	6.34E-13	9.03E-09	6.16628E-11	7.74E-09	1.04E-08
SFO_M2	4.66E-13	1.38792E-15	7.48E-14	9.54E-13	6.29E-09	2.8483E-10	4.47E-09	6E-09
SFO_M3	2.28E-13	1.13843E-15	8.73E-14	3.05E-13	6.93E-09	1.06361E-10	2.36E-09	1.74E-08
SFO_M4	6.92E-13	7.73717E-16	7.93E-14	1.65E-12	7.04E-09	3.92124E-10	4.67E-09	7.96E-09
SFO_M5	3.64E-13	5.60338E-16	1.77E-13	5.92E-13	7.57E-09	1.57262E-12	3.52E-09	9.36E-09
SFO_M6	2.27E-13	2.72601E-15	1.41E-13	2.49E-13	5.19E-09	9.05888E-13	3.87E-09	4.97E-09
SFO_M7	4.47E-13	2.36107E-15	1.25E-13	6.82E-13	5.64E-09	1.90414E-10	4.22E-09	5.59E-09
SFO_M8	2.7E-13	8.18907E-17	9.45E-14	4.9E-13	4.8E-09	1.41594E-10	3.63E-09	4.77E-09
SFO_M9	1.89E-13	5.34299E-17	4.99E-14	3.57E-13	1.1E-08	9.81355E-11	7.24E-09	1.4E-08
SFO_M10	6.01E-13	7.94342E-16	9.38E-14	1.64E-12	5.37E-09	9.71627E-10	4.03E-09	3.76E-09
	F5				F6			
	Mean	Best	Median	Std. Dev	Mean	Best	Median	Std. Dev
SFO	0.002065	8.48521E-08	0.001592	0.001943	1.707286	0.047265876	0.331217	2.246986
SFO_M1	0.002279	1.93592E-08	0.00119	0.002758	0.199692	0.005038043	0.161012	0.142088

(continued)

Table 7. Mean, median, standard deviation and best values in multimodal functions (F1- F7) for all techniques (continued)

F5					F6			
	Mean	Best	Median	Std. Dev	Mean	Best	Median	Std. Dev
SFO_M2	0.003127	1.9397E-05	0.000901	0.006149	0.201037	0.012562147	0.098267	0.262822
SFO_M3	0.002979	2.66513E-05	0.000876	0.005135	0.177861	0.020194202	0.088759	0.208825
SFO_M4	0.001603	2.88972E-08	0.000764	0.002064	1.555321	0.015998155	0.851989	1.939228
SFO_M5	0.001665	6.77579E-07	0.000671	0.002005	0.876181	0.025100904	0.220485	1.259514
SFO_M6	0.003497	4.60483E-06	0.001469	0.006641	1.56821	0.028260543	0.526884	2.000048
SFO_M7	0.005012	1.84142E-05	0.002438	0.006152	1.42255	0.070939192	0.502469	1.767784
SFO_M8	0.002451	1.58854E-05	0.001538	0.002389	0.316152	0.047793237	0.2673	0.285682
SFO_M9	0.0022	3.02215E-06	0.000949	0.004389	0.325643	0.041677588	0.154314	0.349044
SFO_M10	0.002396	3.59715E-05	0.000805	0.003042	0.981522	0.087499115	0.50128	1.335444
F7								
	Mean	Best	Median	Std. Dev				
SFO	0.000342	1.39876E-05	0.000183	0.000383				
SFO_M1	0.000294	7.4195E-06	0.00015	0.000414				
SFO_M2	0.000338	6.30231E-06	0.000301	0.000272				
SFO_M3	0.000433	2.42023E-06	0.000272	0.000543				
SFO_M4	0.000277	1.08965E-05	9.81E-05	0.00045				
SFO_M5	0.000281	1.20352E-05	0.000141	0.000318				
SFO_M6	0.000274	5.95496E-05	0.000155	0.000274				
SFO_M7	0.000386	1.33029E-05	0.000211	0.000513				
SFO_M8	0.000434	5.82007E-05	0.000197	0.000531				
SFO_M9	0.000296	3.79843E-05	0.00016	0.000437				
SFO_M10	0.000346	4.1153E-06	0.000145	0.000665				

5.2.2. Analysis of High-Dimensional Multimodal Benchmark Functions

Multimodal benchmark functions (F8 – F13) have a massive number of local optima and hence they can be better used to test local optima avoidance and exploration capability of the algorithm. In other words, high dimensional multimodal functions are suitable for testing how well an algorithm explores the open search space. This group is considered to be the most difficult to optimize as the number of local optima increases with the increasing dimensions. Now, if we look closely at the mean results generated by high dimensional multimodal functions (F8-F13) in Table 8, then it is interesting to note that SFO had below average performance (< 5 rank), whereas SFO_M3 was the winner in three functions (F8, F10, F12), SFO_M5 in F9, SFO_M9 in F13 and there was a tie between SFO_M4 and SFO_M5 in F11 for being best option. For the best values generated by each high dimensional multimodal function, all the techniques performed similarly for F9 and F11. SFO_M2 showed the best results for F8, SFO_M5 for F10, and SFO_M1 for F13. SFO with random population performed best for F12. Overall results again endorse the fact that the performance of SFO is improved when the initial population is chaotic.

Table 8. Mean, median, standard deviation and best values in multimodal functions (F8-F13) for all techniques

	F8				F9			
	Mean	Best	Median	Std. Dev	Mean	Best	Median	Std. Dev
SFO	-3790.227	-5489.43162	-3.81E+03	772.5454	2.52E-13	0	0	6.22E-13
SFO_M1	-6099.292	-6099.29223	-6099.292	0	6.14E-13	0	1.14E-13	8.74E-13
SFO_M2	-5309.962	-8998.20927	-4727.855	2196.549	1.71E-13	0	0	3.74E-13
SFO_M3	-6156.794	-6809.88179	-6257.159	365.1468	3.12E-13	0	1.14E-13	3.74E-13
SFO_M4	-4929.293	-6442.42712	-4936.9	606.4734	3.41E-13	0	0	6.94E-13
SFO_M5	-3696.408	-5288.96485	-3691.828	537.9836	1.09E-13	0	0	1.83E-13
SFO_M6	-3587.817	-4901.89522	-3590.719	457.4052	1.68E-13	0	0	4.6E-13
SFO_M7	-4364.167	-6316.07836	-4330.463	706.4712	1.32E-13	0	0	2.65E-13
SFO_M8	-4949.593	-6638.94492	-4955.544	860.4344	2.73E-13	0	0	6.21E-13
SFO_M9	-4581.722	-4870.26592	-4571.602	83.31105	2.41E-13	0	5.68E-14	4.47E-13
SFO_M10	-3442.116	-5009.15645	-3371.879	445.1623	1.73E-13	0	0	2.98E-13
	F10				F11			
	Mean	Best	Median	Std. Dev	Mean	Best	Median	Std. Dev
SFO	2.22E-08	3.88996E-11	1.42E-08	2.32E-08	5.33E-17	0	0	1.64E-16
SFO_M1	1.62E-08	7.25685E-10	7.27E-09	2.1E-08	4.88E-17	0	0	1.57E-16
SFO_M2	1.88E-08	8.49568E-10	1.22E-08	1.86E-08	3.55E-17	0	0	6.82E-17
SFO_M3	1.34E-08	5.25064E-10	7.38E-09	2.07E-08	2.09E-16	0	0	6.1E-16
SFO_M4	2.17E-08	6.06495E-10	1.05E-08	2.58E-08	8.88E-18	0	0	3.01E-17
SFO_M5	1.67E-08	6.22524E-12	8.41E-09	1.91E-08	8.88E-18	0	0	3.01E-17
SFO_M6	1.4E-08	9.01243E-10	1.3E-08	1.17E-08	4.88E-17	0	0	1.63E-16
SFO_M7	1.35E-08	6.99878E-10	1.07E-08	1.09E-08	1.78E-17	0	0	4.07E-17
SFO_M8	1.39E-08	9.10608E-10	1.14E-08	1.15E-08	3.11E-17	0	0	1.31E-16
SFO_M9	2.56E-08	1.92668E-10	1.33E-08	3.11E-08	1.33E-17	0	0	4.78E-17
SFO_M10	1.82E-08	5.51933E-10	7.37E-09	2.39E-08	7.55E-17	0	0	2.84E-16
	F12				F13			
	Mean	Best	Median	Std. Dev	Mean	Best	Median	Std. Dev
SFO	0.038453	1.85031E-08	0.000278	0.05479	7.78E-06	1.29965E-07	1.69E-06	1.36E-05
SFO_M1	0.005945	1.11418E-06	0.000271	0.016705	4.56E-06	2.62226E-08	2.22E-06	5.12E-06
SFO_M2	0.048005	3.11391E-08	0.000234	0.136502	6.38E-06	6.14887E-12	2.76E-06	1.02E-05
SFO_M3	0.003388	1.40092E-06	0.000127	0.011627	7.19E-06	4.04891E-08	2.91E-06	1.01E-05
SFO_M4	0.092758	2.52617E-07	0.00054	0.1849	8.01E-06	1.18282E-08	5.37E-06	1.13E-05
SFO_M5	0.070279	3.20721E-06	0.001071	0.121811	7.41E-06	1.34816E-08	2.51E-06	9.23E-06
SFO_M6	0.078838	4.3865E-07	0.010648	0.145297	8.57E-06	1.80231E-07	5.9E-06	1.13E-05
SFO_M7	0.037242	6.91409E-07	0.000145	0.079862	7.19E-06	9.13896E-09	3.16E-06	9.77E-06
SFO_M8	0.005179	2.50284E-07	0.002517	0.007067	4.75E-06	1.42413E-08	2.64E-06	5.21E-06
SFO_M9	0.01635	7.77265E-08	0.000149	0.038321	4.5E-06	4.71322E-09	9.51E-07	7.64E-06
SFO_M10	0.046912	3.65158E-07	0.009534	0.057862	9.08E-06	5.20297E-08	4.39E-06	1.18E-05

5.2.3. Analysis of Fixed-Dimensional Multimodal Benchmark Functions

The third group is fixed dimension multimodal benchmark functions (F14-F23) that are like the second group, however they have fewer local optima and have low dimensions (< 6). For fixed dimensional multimodal functions, results are once again in favor of chaotic maps, except for F19, where both mean and best values were generated by SFO (See Tables 9 and 10). In the overall mean results, SFO_M2, SFO_M3 and SFO_M6 provided promising values. It is important to recall here that this study investigates the impact of chaotic maps on initial population and since chaotic maps start with the representative solution, therefore, they performed better in multimodal functions and avoided local optima better than the basic SFO. The next section discusses and verifies the significance of the overall results of this study.

Table 9. Mean, median, standard deviation and best values in multimodal functions (F14-F19) for all techniques

	F14				F15			
	Mean	Best	Median	Std. Dev	Mean	Best	Median	Std. Dev
SFO	9.006231	0.99800384	8.840836	3.681908	0.00224	0.000308808	0.000391	0.005787
SFO_M1	8.189574	0.998003843	6.903336	4.026676	0.023213	0.000309074	0.000663	0.030098
SFO_M2	5.338558	1.9920309	5.928845	1.363346	0.000414	0.000308426	0.000422	9.04E-05
SFO_M3	7.804219	2.982105157	7.873993	3.905195	0.001194	0.000309159	0.000576	0.000931
SFO_M4	8.688608	0.998003838	10.76318	4.242894	0.000474	0.000308767	0.000416	0.000165
SFO_M5	9.26148	0.998003838	12.67051	4.562065	0.020515	0.000309855	0.000349	0.029374
SFO_M6	10.71188	2.982105209	11.7187	2.550613	0.000423	0.000310913	0.000339	0.00022
SFO_M7	8.065527	0.998003838	8.840836	4.662701	0.045693	0.000309173	0.063337	0.028294
SFO_M8	11.08625	0.998240396	12.67051	3.251919	0.000445	0.000307591	0.000407	0.000118
SFO_M9	11.12684	0.998507089	12.67051	2.820946	0.000497	0.000308703	0.000341	0.000319
SFO_M10	5.65195	0.998004088	5.928845	1.291178	0.000526	0.000308741	0.000511	0.000191
	F16				F17			
	Mean	Best	Median	Std. Dev	Mean	Best	Median	Std. Dev
SFO	-1.031606	-1.03162845	-1.031628	5.18E-05	0.39789	0.397887358	0.397887	1.23E-05
SFO_M1	-1.03162	-1.03162845	-1.031628	2.38E-05	0.397887	0.397887358	0.397887	1.12E-08
SFO_M2	-1.031627	-1.03162845	-1.031628	5.51E-06	0.397887	0.397887358	0.397887	8.53E-09
SFO_M3	-1.006133	-1.03162845	-1.031627	0.059887	0.398001	0.397887358	0.397887	0.000314
SFO_M4	-1.031623	-1.03162845	-1.031628	2.02E-05	0.398021	0.397887358	0.397887	0.000432
SFO_M5	-1.031625	-1.03162845	-1.031628	9.65E-06	0.397966	0.397887358	0.397887	0.000385
SFO_M6	-1.031586	-1.03162845	-1.031624	8.91E-05	0.397887	0.397887358	0.397887	1.65E-09
SFO_M7	-1.031585	-1.03162845	-1.031628	0.000129	0.397887	0.397887358	0.397887	4.2E-09
SFO_M8	-1.030605	-1.03162845	-1.031628	0.00497	0.397964	0.397887358	0.397887	0.000373
SFO_M9	-1.031628	-1.03162845	-1.031628	8.61E-09	0.397888	0.397887358	0.397887	4.9E-06
SFO_M10	-1.031628	-1.03162845	-1.031628	1.62E-08	0.397887	0.397887358	0.397887	5.24E-09
	F18				F19			
	Mean	Best	Median	Std. Dev	Mean	Best	Median	Std. Dev
SFO	6.24001	3.000000026	3.000001	8.773969	-3.788603	-3.86278211	-3.861934	0.208429
SFO_M1	5.160159	3.000000152	3.000053	7.325127	-3.765188	-3.86277127	-3.861112	0.250238
SFO_M2	15.96003	3.000003013	3.000264	13.48917	-3.835396	-3.86272934	-3.831955	0.016864

	F18				F19			
	Mean	Best	Median	Std. Dev	Mean	Best	Median	Std. Dev
SFO_M3	3.000028	3.000000079	3.000012	4.94E-05	-3.670053	-3.86185931	-3.724063	0.212668
SFO_M4	4.080001	3.000000002	3	5.290898	-3.770575	-3.86278208	-3.862613	0.122349
SFO_M5	32.16002	3.000001502	3.000028	38.88	-3.860252	-3.86277623	-3.861217	0.002925
SFO_M6	3.000002	3.000000051	3.000001	4.02E-06	-3.855982	-3.86278184	-3.859001	0.007826
SFO_M7	15.96002	3.000000212	3.00001	29.69509	-3.86089	-3.86274617	-3.861096	0.001225
SFO_M8	68.88005	3.000003019	84	30.62324	-3.860038	-3.8625343	-3.86089	0.002519
SFO_M9	84.00002	84.00000027	84.00001	1.92E-05	-3.830177	-3.86277999	-3.823685	0.024843
SFO_M10	20.28177	3.00000022	30.00011	12.96132	-3.838152	-3.86278078	-3.828897	0.017192

Table 10. Mean, Median, Standard Deviation and Best values in multimodal functions (F20-F23) for all techniques

	F20				F21			
	Mean	Best	Median	Std. Dev	Mean	Best	Median	Std. Dev
SFO	-2.981951	-3.30794818	-3.064445	0.305237	-10.1262	-10.1528771	-10.14428	0.046345
SFO_M1	-2.964347	-3.27010773	-3.022736	0.188488	-10.13304	-10.1531054	-10.14297	0.024699
SFO_M2	-2.775056	-3.2819005	-2.845905	0.349242	-10.12517	-10.1531523	-10.14359	0.044107
SFO_M3	-2.535158	-3.23874902	-2.7384	0.518037	-10.13962	-10.1530375	-10.14791	0.017083
SFO_M4	-2.838025	-3.12176873	-2.803186	0.144466	-10.10967	-10.1525112	-10.12636	0.046876
SFO_M5	-2.96115	-3.10519461	-2.927867	0.065568	-10.11131	-10.149704	-10.12456	0.035671
SFO_M6	-2.996723	-3.31193297	-3.002989	0.233633	-10.13039	-10.1525056	-10.13379	0.020591
SFO_M7	-2.953756	-3.07334691	-2.936767	0.044759	-10.10901	-10.152857	-10.12088	0.044522
SFO_M8	-2.847962	-3.11745561	-2.834594	0.121156	-10.11785	-10.1531985	-10.13077	0.035775
SFO_M9	-2.784176	-3.28629327	-2.823611	0.389538	-10.12626	-10.1528175	-10.13336	0.022931
SFO_M10	-2.980367	-3.30246284	-3.005761	0.196323	-10.09825	-10.1528089	-10.11697	0.063703
	F22				F23			
	Mean	Best	Median	Std. Dev	Mean	Best	Median	Std. Dev
SFO	-10.37129	-10.4027164	-10.38752	0.044726	-10.51278	-10.5356957	-10.52649	0.034133
SFO_M1	-10.37751	-10.4028303	-10.39291	0.031	-10.26014	-10.5361767	-10.53138	1.246733
SFO_M2	-10.37475	-10.402749	-10.3912	0.04174	-10.51744	-10.5361053	-10.52834	0.022086
SFO_M3	-10.38922	-10.4029275	-10.39513	0.019771	-10.5169	-10.5363314	-10.52538	0.024081
SFO_M4	-10.37088	-10.4029332	-10.37667	0.029877	-10.51477	-10.5362894	-10.52871	0.02922
SFO_M5	-10.37394	-10.4028849	-10.3825	0.032225	-10.49848	-10.5319518	-10.50331	0.024988
SFO_M6	-8.846035	-10.4027768	-10.38967	3.090603	-10.49199	-10.5362221	-10.50758	0.042137
SFO_M7	-10.3742	-10.4028741	-10.38587	0.03796	-10.49863	-10.5359015	-10.51324	0.036889
SFO_M8	-10.08307	-10.402825	-10.39849	1.507783	-10.20741	-10.5361827	-10.52791	1.546578
SFO_M9	-10.37006	-10.40279	-10.38739	0.042799	-10.50027	-10.5359724	-10.50497	0.033799
SFO_M10	-10.37607	-10.4029	-10.38817	0.026948	-10.49628	-10.5349336	-10.51256	0.039632

5.2.4. Analysis of Results Based on Ranking

First, the overall rank of each technique (for mean and best results) on all benchmark functions was calculated and shown in Tables 11 and 12, respectively. Then final rank (see Table 13) was calculated

Table 11. Average rank of each technique based on mean values

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23
SFO	5	7	1	4	3	11	7	8	7	10	9	6	8	7	8	7	7	5	8	2	5	7	4
SFO_M1	4	3	6	10	5	2	4	2	11	5	7.5	3	2	5	10	6	5	4	10	4	2	2	10
SFO_M2	1	2	9	6	9	3	6	3	4	8	6	8	4	1	1	3	4	7	6	10	6	4	1
SFO_M3	2	5	4	7	8	1	10	1	9	1	11	1	5	3	7	11	10	2	11	11	1	1	2
SFO_M4	3	8	11	8	1	9	2	5	10	9	1.5	11	9	6	4	5	11	3	9	8	9	8	3
SFO_M5	7	9	7	9	2	6	3	9	1	6	1.5	9	7	8	9	4	9	9	2	5	8	6	7
SFO_M6	9	11	3	2	10	10	1	10	3	4	7.5	10	10	9	2	8	1	1	4	1	3	11	9
SFO_M7	8	10	8	5	11	8	9	7	2	2	4	5	6	4	11	9	2	6	1	6	10	5	6
SFO_M8	10	4	5	1	7	4	11	4	8	3	5	2	3	10	3	10	8	10	3	7	7	10	11
SFO_M9	11	1	2	11	4	5	5	6	6	11	3	4	1	11	5	1	6	11	7	9	4	9	5
SFO_M10	6	6	10	3	6	7	8	11	5	7	10	7	11	2	6	2	3	8	5	3	11	3	8

Table 12. Average rank of each technique based on best value

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23
SFO	3	6	4	9	3	8	8	7	6	2	6	1	10	4	6	6	5.5	2	1	2	5	11	9
SFO_M1	6	9	1	3	1	1	4	6	6	8	6	9	7	5	7	4	7	5	7	6	3	6	5
SFO_M2	4	4	9	8	9	2	3	1	6	9	6	2	1	9	2	2	4	9	9	5	2	10	6
SFO_M3	11	8	8	5	10	4	1	2	6	4	6	10	8	10	8	8	11	4	11	7	4	2	1
SFO_M4	2	5	6	10	2	3	5	4	6	6	6	5	4	3	5	1	8	1	2	8	9	1	2
SFO_M5	7	2	5	2	4	5	6	8	6	1	6	11	5	2	10	7	3	8	6	10	11	4	11
SFO_M6	10	10	11	1	6	6	11	10	6	10	6	7	11	11	11	11	5.5	3	3	1	10	9	3
SFO_M7	9	3	10	7	8	10	7	5	6	7	6	8	3	1	9	10	2	6	8	11	6	5	8
SFO_M8	1	1	3	6	7	9	10	3	6	11	6	4	6	7	1	9	9	10	10	9	1	7	4
SFO_M9	8	7	2	4	5	7	9	11	6	3	6	3	2	8	3	5	10	11	5	4	7	8	7
SFO_M10	5	11	7	11	11	11	11	2	9	6	5	6	9	6	4	3	1	7	4	3	8	3	10

Table 13. Final ranking of all techniques based on average and best ranks

	SFO	SFO_M1	SFO_M2	SFO_M3	SFO_M4	SFO_M5	SFO_M6	SFO_M7	SFO_M8	SFO_M9	SFO_M10
Avg Mean Rank	8.5	2	1	3	11	6	5	7	8.5	4	10
Avg Best Rank	4	2.5	2.5	9	1	5.5	11	10	5.5	7	8
Final Rank	7	2	1	5.5	5.5	4	9	10	8	3	11

based on the mean and best ranks. For average mean ranking, SFO and SFO_M8 have the same rank. The lowest ranked technique is the Tent map, whereas circle map is the best method for population initialization of SFO. The Gaussian map and the iterative map perform similarly to each other in terms of rank.

The superior performance of certain chaotic maps, such as the circle, Gaussian or iterative maps, can be attributed to their stronger ergodicity and ability to generate more uniformly distributed values across the search space. These properties ensure better coverage of the search space hence reducing the likelihood of premature convergence. For example, the circle map produces quasi-periodic sequences that enhance global exploration. On the other hand, maps such as the tent or sine exhibit patterns that may lead to clustered sequences, limiting their effectiveness in maintaining population diversity. This explains why not all chaotic maps consistently outperform random initialization, but specific maps align better with the dynamics of the Sailfish Optimizer when tackling functions with a high density of local optima.

5.2.5. Statistical Analysis Based on Mann Whitney U Test

To further verify the significance of the results, the Mann Whitney U test with 95 % significance level ($\alpha = 0.05$) was performed for two selected maps (the best performing in terms of mean and best results) so that the impact of changing initial population is observed with statistical proof. For example, F1 SFO_M2 and SFO_M8 were compared to SFO to generate a p-value, and SFO_M8 and SFO_M9 were compared to SFO for F2 and so on. F3 was omitted because SFO itself shows the best mean results for F3. Algorithm pairs and obtained p-values are reported in Table 14. If p-value is less

Table 14. Statistical analysis of selected maps with SFO

Functions	Algorithm Pairs	P Value	Functions	Algorithm Pairs	P Value
F1	SFO_M2 Vs SFO	0.1121	F13	SFO_M2 Vs SFO	0.8026
	SFO_M8 Vs SFO	0.8325		SFO_M9 Vs SFO	0.3067
F2	SFO_M9 Vs SFO	0.4074	F14	SFO_M2 Vs SFO	0.001
	SFO_M8 Vs SFO	0.5768		SFO_M10 Vs SFO	0.002
F3	Not Applicable	NA	F15	SFO_M2 Vs SFO	0.4643
				SFO_M6 Vs SFO	0.0429
F4	SFO_M8 Vs SFO	0.5253	F16	SFO_M9 Vs SFO	0.0009
	SFO_M6 Vs SFO	0.7005		SFO_M10 Vs SFO	0.0012

(continued)

Table 14. Statistical analysis of selected maps with SFO (continued)

Functions	Algorithm Pairs	P Value	Functions	Algorithm Pairs	P Value
F5	SFO_M4 Vs SFO	0.2164	F17	SFO_M6 Vs SFO	0.0006
	SFO_M1 Vs SFO	0.8475		SFO_M10 Vs SFO	0.0499
F6	SFO_M3 Vs SFO	0.0003	F18	SFO_M6 Vs SFO	0.3271
	SFO_M1 Vs SFO	0.0152		SFO_M3 Vs SFO	0.0055
F7	SFO_M6 Vs SFO	0.8325	F19	SFO_M7 Vs SFO	0.6862
	SFO_M3 Vs SFO	0.5004		SFO_M8 Vs SFO	0.5637
F8	SFO_M3 Vs SFO	< 0.0001	F20	SFO_M6 Vs SFO	0.954
	SFO_M2 Vs SFO	0.0108		SFO_M10 Vs SFO	0.4297
F9	SFO_M5 Vs SFO	0.9567	F21	SFO_M1 Vs SFO	0.7292
	SFO_M7 Vs SFO	0.8556		SFO_M3 Vs SFO	0.5768
F10	SFO_M3 Vs SFO	0.0842	F22	SFO_M1 Vs SFO	0.7005
	SFO_M5 Vs SFO	0.441		SFO_M3 Vs SFO	0.1467
F11	SFO_M4 Vs SFO	0.353	F23	SFO_M6 Vs SFO	0.7005
	SFO_M5 Vs SFO	0.353		SFO_M3 Vs SFO	0.7437
F12	SFO_M3 Vs SFO	0.3548			
	SFO_M8 Vs SFO	0.8026			

than 0.05, it means that there is a significant difference between compared techniques and if $p > 0.05$ than both techniques are similar. P value indicates that SFO initialized with chaotic maps is slightly better than the random values in one unimodal function (F6), and six multimodal functions (F8, F14-F18). It means that the performance is improved by applying chaotic maps when exploration is tested, however little improvement is shown in the exploitation phase. It can be said that chaotic maps can be positively used to improve the initial population of other SI based algorithms. The convergence curves of the selected functions are also presented to better understand convergence improvement in Figure 2.

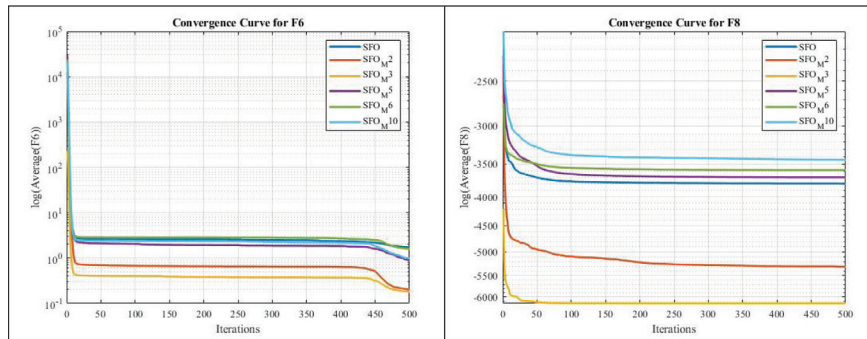


Figure 2. Comprehensive Convergence analysis of SFO Vs chaotic SFO variants on unimodal, high-dimensional multimodal and fixed-dimensional multimodal functions

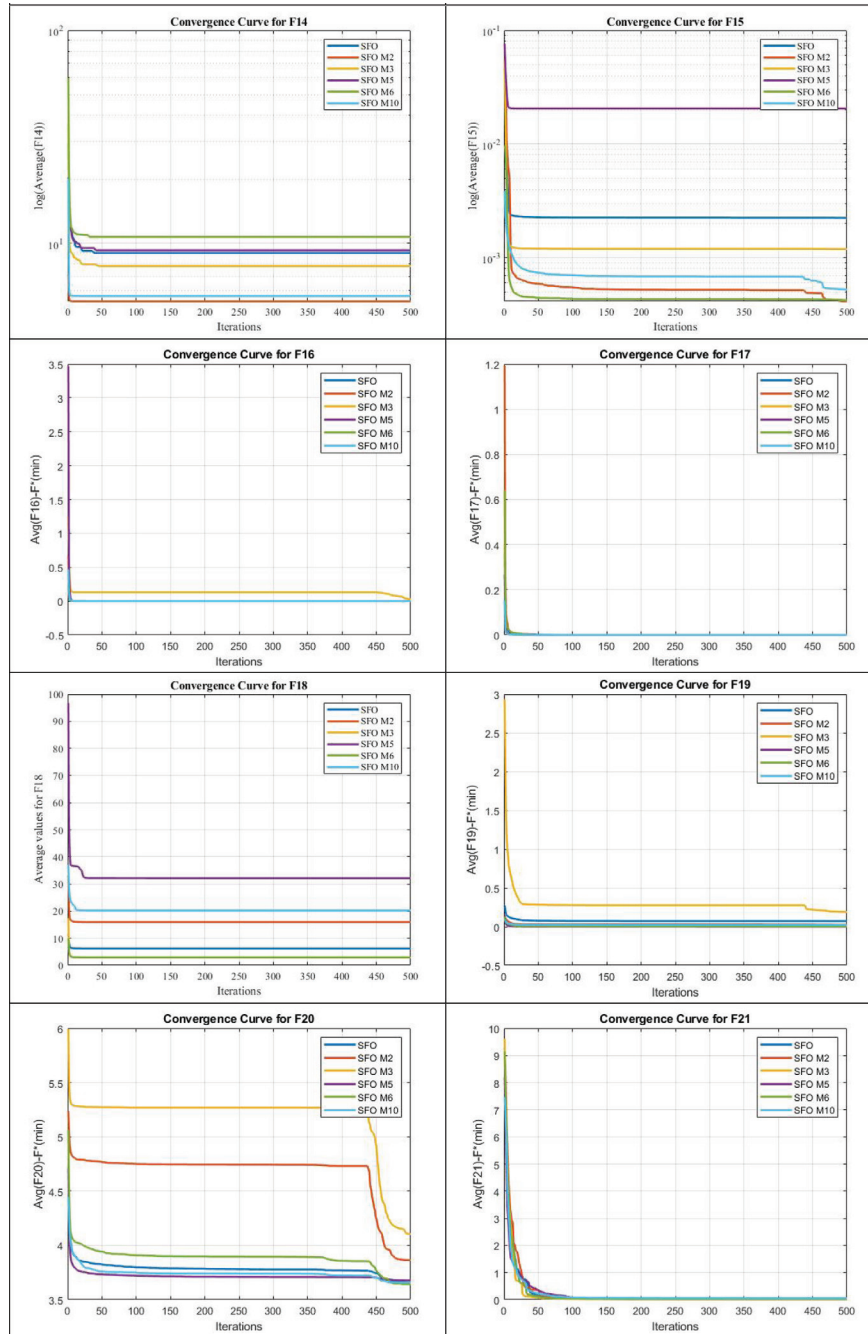


Figure 2. Comprehensive Convergence analysis of SFO Vs chaotic SFO variants on unimodal, high-dimensional multimodal and fixed-dimensional multimodal functions (continued)

6. Conclusion

This study investigated the impact of chaotic maps on the initial population of a swarm intelligence-based algorithm, namely, sailfish optimizer. Ten popular chaotic maps were utilized to initialize sailfish and sardine population and applied on twenty-three well known benchmark functions. Results show significant improvement in the exploration of the algorithm and slight enhancement in the exploitation phase, where all techniques performed equally well. All the techniques were assigned ranks according to their performance in terms of mean and best results. A final ranking was then calculated based on mean and best ranks. The final ranking showed interesting insights into the research. SFO stood seventh, it means that SFO had below average performance and at least six chaotic maps had better performance than SFO. Mean performance of SFO was also near the bottom line. At the end, statistical testing was performed using Mann Whitney U test and convergence was visualized on graphs. In a few problems (F12, F19), SFO with random population performed better, in nine problems SFO performed similar to the other techniques. However, in twelve problems, SFO showed below average performance. Therefore, it is concluded that applying chaotic maps on initial population is useful when exploration is concerned, and it can also slightly improve exploitation. In the future, sailfish optimizer algorithm's internal parameters can be tuned and adjusted with chaotic maps to further explore the effectiveness of the technique. Additionally, the impact of chaotic maps on other recent metaheuristic algorithms could be explored in the future.

References

- Abdel-Salam, M., Hu, G., Çelik, E., Gharehchopogh, F. S., & EL-Hasnony, I. M. (2024). Chaotic RIME optimization algorithm with adaptive mutualism for feature selection problems. *Computers in Biology and Medicine*, 179, 108803. <https://doi.org/10.1016/j.combiomed.2024.108803>
- Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., & Gandomi, A. H. (2021). The Arithmetic Optimization Algorithm. *Computer Methods in Applied Mechanics and Engineering*, 376, 113609. <https://doi.org/10.1016/j.cma.2020.113609>
- Adamuthe, A. C., & Nitave, T. R. (2018). Adaptive harmony search for optimizing constrained Resource Allocation problem. *International Journal of Computing*, 17(4). <https://doi.org/10.47839/ijc.17.4.1148>
- Agushaka, J. O., & Ezugwu, A. E. (2020). Influence of Initializing Krill Herd Algorithm with Low-Discrepancy Sequences. *IEEE Access*, 8. <https://doi.org/10.1109/ACCESS.2020.3039602>
- Agushaka, J. O., & Ezugwu, A. E. (2021). Evaluation of several initialization methods on arithmetic optimization algorithm performance. *Journal of Intelligent Systems*, 31(1). <https://doi.org/10.1515/jisys-2021-0164>
- Agushaka, J. O., & Ezugwu, A. E. (2022). Initialisation Approaches for Population-Based Metaheuristic Algorithms: A Comprehensive Review. *Applied Sciences*, 12(2). <https://doi.org/10.3390/app12020896>
- Alatas, B. (2010a). Chaotic bee colony algorithms for global numerical optimization. *Expert Systems with Applications*, 37(8). <https://doi.org/10.1016/j.eswa.2010.02.042>
- Alatas, B. (2010b). Chaotic harmony search algorithms. *Applied Mathematics and Computation*, 216(9). <https://doi.org/10.1016/j.amc.2010.03.114>

- Alkafaween, E., Hassanat, A. B. A., & Tarawneh, S. (2021). Improving initial population for genetic algorithm using the multi linear regression based technique (mlrbt). *Communications - Scientific Letters of the University of Žilina*, 23(1). <https://doi.org/10.26552/COM.C.2021.1.E1-E10>
- Aribowo, W., Suprianto, B., & Prapanca, A. (2023). A novel modified dandelion optimizer with application in power system stabilizer. *IAES International Journal of Artificial Intelligence*, 12(4). <https://doi.org/10.11591/ijai.v12.i4.pp2033-2041>
- Ashraf, A., Pervaiz, S., Haider Bangyal, W., Nisar, K., Ag. Ibrahim, A. A., Rodrigues, J. J. P. C., & Rawat, D. B. (2021). Studying the impact of initialization for population-based algorithms with low-discrepancy sequences. *Applied Sciences (Switzerland)*, 11(17). <https://doi.org/10.3390/app11178190>
- Bangyal, W. H., Nisar, K., Ibrahim, A. A. B. A., Haque, M. R., Rodrigues, J. J. P. C., & Rawat, D. B. (2021). Comparative analysis of low discrepancy sequence-based initialization approaches using population-based algorithms for solving the global optimization problems. *Applied Sciences (Switzerland)*, 11(16). <https://doi.org/10.3390/app11167591>
- Boussaid, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, 237, 82–117. <https://doi.org/10.1016/j.ins.2013.02.041>
- Chen, H., Li, W., & Yang, X. (2020). A whale optimization algorithm with chaos mechanism based on quasi-opposition for global optimization problems. *Expert Systems with Applications*, 158, 113612. <https://doi.org/10.1016/j.eswa.2020.113612>
- Chen, X. (2020). Research on New Adaptive Whale Algorithm. *IEEE Access*, 8, 90165–90201. <https://doi.org/10.1109/ACCESS.2020.2993580>
- Deng, Y., Liu, Y., & Zhou, D. (2015). An Improved Genetic Algorithm with Initial Population Strategy for Symmetric TSP. *Mathematical Problems in Engineering*. <https://doi.org/10.1155/2015/212794>
- Digalakis, J. G., & Margaritis, K. G. (2001). On benchmarking functions for genetic algorithms. *International Journal of Computer Mathematics*, 77(4), 481–506.
- Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: A new meta-heuristic. *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, 2, 1470–1477. <https://doi.org/10.1109/CEC.1999.782657>
- El Ghouate, N., Bencherqui, A., Mansouri, H., Maloufy, A. El, Tahiri, M. A., Karmouni, H., Sayyouri, M., Askar, S. S., & Abouhawwash, M. (2024). Improving the Kepler optimization algorithm with chaotic maps: Comprehensive performance evaluation and engineering applications. *Artificial Intelligence Review*, 57(11), 313.
- Elaziz, M. A., & Mirjalili, S. (2019). A hyper-heuristic for improving the initial population of whale optimization algorithm. *Knowledge-Based Systems*, 172, 42–63. <https://doi.org/10.1016/j.knosys.2019.02.010>
- Fan, Q., Chen, Z., Li, Z., Xia, Z., Yu, J., & Wang, D. (2021). A new improved whale optimization algorithm with joint search mechanisms for high-dimensional global optimization problems. *Engineering with Computers*, 37(3), 1851–1878. <https://doi.org/10.1007/s00366-019-00917-8>
- Gandomi, A. H., & Alavi, A. H. (2012). Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation*, 17(12). <https://doi.org/10.1016/j.cnsns.2012.05.010>
- Gandomi, A. H., Yang, X. S., Talatahari, S., & Alavi, A. H. (2013). Firefly algorithm with chaos. *Communications in Nonlinear Science and Numerical Simulation*, 18(1). <https://doi.org/10.1016/j.cnsns.2012.06.009>

- Gandomi, A. H., & Yang, X. S. (2014). Chaotic bat algorithm. *Journal of Computational Science*, 5(2). <https://doi.org/10.1016/j.jocs.2013.10.002>
- Gao, W. F., Liu, S. Y., & Jiang, F. (2011). An improved artificial bee colony algorithm for directing orbits of chaotic systems. *Applied Mathematics and Computation*, 218(7). <https://doi.org/10.1016/j.amc.2011.09.034>
- Gupta, A., Tiwari, D., Kumar, V., Rana, K. P. S., & Mirjalili, S. (2022). A Chaos-Infused Moth-Flame Optimizer. *Arabian Journal for Science and Engineering*, 47(8). <https://doi.org/10.1007/s13369-022-06689-6>
- Hefny, H. A., & Azab, S. S. (2010). Chaotic particle swarm optimization. *2010 The 7th International Conference on Informatics and Systems (INFOS)*, 1–8.
- Huang, X., Zeng, X., Han, R., & Wang, X. (2019). An enhanced hybridized artificial bee colony algorithm for optimization problems. *IAES International Journal of Artificial Intelligence*, 8(1). <https://doi.org/10.11591/ijai.v8.i1.pp87-94>
- Javidi, M., & Hosseinpourfard, R. (2015). Chaos genetic algorithm instead genetic algorithm. *International Arab Journal of Information Technology*, 12(2).
- Jianhao, W., Long, W., Lijie, C., & Tian, G. (2021). Enhanced whale optimization algorithm for large-scale global optimization problems. *2021 International Conference on Computer Communication and Artificial Intelligence, CCAI 2021*, 180–187. <https://doi.org/10.1109/CCAI50917.2021.9447541>
- Jin, Q., Xu, Z., & Cai, W. (2021). An improved whale optimization algorithm with random evolution and special reinforcement dual-operation strategy collaboration. *Symmetry*, 13(2), 1–24. <https://doi.org/10.3390/sym13020238>
- Karaboga, D., & Basturk, B. (2007). Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems. *Lecture Notes in Computer Science*, 4529 LNAI, 789–798. https://doi.org/10.1007/978-3-540-72950-1_77
- Kaur, G., & Arora, S. (2018). Chaotic whale optimization algorithm. *Journal of Computational Design and Engineering*, 5(3), 275–284. <https://doi.org/10.1016/j.jcde.2017.12.006>
- Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, 4, 1942–1948. <https://doi.org/10.1109/ICNN.1995.488968>
- Kiani, F., Nematzadeh, S., Anka, F. A., & Findikli, M. A. (2023). Chaotic Sand Cat Swarm Optimization. *Mathematics*, 11(10). <https://doi.org/10.3390/math11102340>
- Li, Q., Liu, S. Y., & Yang, X. S. (2020). Influence of initialization on the performance of metaheuristic optimizers. *Applied Soft Computing Journal*, 91. <https://doi.org/10.1016/j.asoc.2020.106193>
- Lu, C., Gao, L., Li, X., Hu, C., Yan, X., & Gong, W. (2020). Chaotic-based grey wolf optimizer for numerical and engineering optimization problems. *Memetic Computing*, 12(4). <https://doi.org/10.1007/s12293-020-00313-6>
- Luo, T., Xie, J., Zhang, B., Zhang, Y., Li, C., & Zhou, J. (2024). An improved levy chaotic particle swarm optimization algorithm for energy-efficient cluster routing scheme in industrial wireless sensor networks. *Expert Systems with Applications*, 241, 122780. <https://doi.org/10.1016/j.eswa.2023.122780>
- Ma, Z. S. (2012). Chaotic populations in genetic algorithms. *Applied Soft Computing*, 12(8), 2409–2424.
- Maaranen, H., Miettinen, K., & Mäkelä, M. M. (2004). Quasi-random initial population for genetic algorithms. *Computers & Mathematics with Applications*, 47(12). <https://doi.org/10.1016/j.camwa.2003.07.011>

- Maaranen, H., Miettinen, K., & Penttinen, A. (2007). On initial populations of a genetic algorithm for continuous optimization problems. *Journal of Global Optimization*, 37(3). <https://doi.org/10.1007/s10898-006-9056-6>
- Mahmood, S., Bawany, N. Z., & Tanweer, M. R. (n.d.). A comprehensive survey of Whale Optimization Algorithm: Modifications and classification. *Indonesian Journal of Electrical Engineering and Computer Science*.
- Mirjalili, S., & Lewis, A. (2016). Advances in Engineering Software The Whale Optimization Algorithm. *Advances in Engineering Software*, 95, 51–67. <https://doi.org/10.1016/j.advengsoft.2016.01.008>
- Mohamed, A. A., Kamel, S., Hassan, M. H., & Zeinoddini-Meymand, H. (2024). CAVOA: A chaotic optimization algorithm for optimal power flow with facts devices and stochastic wind power generation. *IET Generation, Transmission and Distribution*, 18(1), 121–144. <https://doi.org/10.1049/gtd2.13076>
- Molga, M., & Smutnicki, C. (2005). Test functions for optimization needs. *Test Functions for Optimization Needs*, 101, 48.
- Ouertani, M. W., Manita, G., & Korbaa, O. (2021). Chaotic lightning search algorithm. *Soft Computing*, 25(3). <https://doi.org/10.1007/s00500-020-05273-0>
- Ruiye, J., Tao, C., Songyan, W., & Ming, Y. (2018). Order whale optimization algorithm in rendezvous orbit design. *Proceedings - 2018 10th International Conference on Advanced Computational Intelligence, ICACI 2018*, 97–102. <https://doi.org/10.1109/ICACI.2018.8377588>
- Saremi, S., Mirjalili, S., & Lewis, A. (2014). Biogeography-based optimisation with chaos. *Neural Computing and Applications*, 25(5). <https://doi.org/10.1007/s00521-014-1597-x>
- Sayed, G. I., Darwish, A., & Hassanien, A. E. (2018). A New Chaotic Whale Optimization Algorithm for Features Selection. *Journal of Classification*, 35(2), 300–344. <https://doi.org/10.1007/s00357-018-9261-2>
- Shadravan, S., Naji, H. R., & Bardsiri, V. K. (2019). The Sailfish Optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. *Engineering Applications of Artificial Intelligence*, 80, 20–34. <https://doi.org/10.1016/j.engappai.2019.01.001>
- Slezkin, A., & Hodashinsky, I. (2021). Population Initialization Methods for the Swallow Swarm Algorithm in Solving the Problem of Fuzzy Classifier Parameter Optimization. *CEUR Workshop Proceedings*, 3047. <https://doi.org/10.47813/sibdata-2-2021-19>
- Storn, R., & Price, K. (1997). Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4), 341–359. <https://doi.org/10.1023/A:1008202821328>
- Tang, R., Fong, S., & Dey, N. (2018). Metaheuristics and Chaos Theory. In *Chaos Theory*. <https://doi.org/10.5772/intechopen.72103>
- Tawhid, M. A., & Ibrahim, A. M. (2022). Improved salp swarm algorithm combined with chaos. *Mathematics and Computers in Simulation*, 202. <https://doi.org/10.1016/j.matcom.2022.05.029>
- Tizhoosh, H. R. (2005). Opposition-based learning: A new scheme for machine intelligence. *Proceedings - International Conference on Computational Intelligence for Modelling, Control and Automation, CIMCA 2005 and International Conference on Intelligent Agents, Web Technologies and Internet, I*, 695–701. <https://doi.org/10.1109/cimca.2005.1631345>
- Wang, G. G., Deb, S., Gandomi, A. H., Zhang, Z., & Alavi, A. H. (2016). Chaotic cuckoo search. *Soft Computing*, 20(9). <https://doi.org/10.1007/s00500-015-1726-1>

- Wang, L., & Zhong, Y. (2015). Cuckoo Search Algorithm with Chaotic Maps. *Mathematical Problems in Engineering*. <https://doi.org/10.1155/2015/715635>
- Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82. <https://doi.org/10.1109/4235.585893>
- Yang, X. S. (2009). Firefly algorithms for multimodal optimization. *Lecture Notes in Computer Science* (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5792 LNCS, 169–178. https://doi.org/10.1007/978-3-642-04944-6_14
- Yang, X. S., & Deb, S. (2009). Cuckoo search via Lévy flights. *2009 World Congress on Nature and Biologically Inspired Computing*, NABIC 2009 - Proceedings, 210–214. <https://doi.org/10.1109/NABIC.2009.5393690>
- Yang, X. S., Deb, S., Zhao, Y. X., Fong, S., & He, X. (2018). Swarm intelligence: past, present and future. *Soft Computing*, 22(18). <https://doi.org/10.1007/s00500-017-2810-5>
- Yang, X. S., & Gandomi, A. H. (2012). Bat algorithm: A novel approach for global engineering optimization. *Engineering Computations (Swansea, Wales)*, 29(5), 464–483. <https://doi.org/10.1108/026444401211235834>
- Yao, X., Liu, Y., & Lin, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2), 82–102.
- Yin, B., Wang, C., & Abza, F. (2020). New brain tumor classification method based on an improved version of whale optimization algorithm. *Biomedical Signal Processing and Control*, 56, 101728. <https://doi.org/10.1016/j.bspc.2019.101728>
- Zhang, Q., & Liu, L. (2019). Whale optimization algorithm based on lamarckian learning for global optimization problems. *IEEE Access*, 7, 36642–36666. <https://doi.org/10.1109/ACCESS.2019.2905009>
- Zhang, Y., & Mo, Y. (2022). Chaotic adaptive sailfish optimizer with genetic characteristics for global optimization. *The Journal of Supercomputing*, 78(8), 10950–10996.