Enhancing AI Optimization with Chaotic Maps: The Oscillating Chaotic Sunflower Optimization Algorithm

M.Bheemalingaiah[1*], G.Sreenivasulu[2], L.Venkateswa Reddy[3], Khaja Mahabubullah[4], A.Ramesh Babu[5], D.Himagiri[6]

[1,2,5,6] Département of CSE, J.B. Institute of Engineering and Technology, Hyderabad, India,
Email : bheemasiva2019@gmail.com[1], g.sreenivasulu@jbiet.edu.in[2], askarbabu@gmail.com[5], himagiri.danapana@gmail.com[6]
[3]Department of CSE, Joginapally B.R. Engineering College, Hyderabad, India,
Email : lakkireddy.v@gmail.com
[4]Department of MCA, Deccan College of Engineering and Technology, Hyderabad, India,
Email : hod_mca@deccancollege.ac.in
*Corresponding Author

Received: 11.04.2024 Revised : 18.05.2024 Accepted: 24.05.2024

ABSTRACT
Metaheuristic algorithms have been at the forefront of optimization research for many years, with continuous advancements and the development of new algorithms. Among these, the recently proposed Sunflower Optimization Algorithm (SFO) has emerged as a notable search algorithm due to its simplicity and effectiveness. However, as a relatively new algorithm, it presents opportunities for further enhancement and flexibility in its methodology. This study introduces the Oscillating Chaotic Sunflower Optimization Algorithm (OCSFO), an innovative variant of the SFO that incorporates a novel exploration technique utilizing chaotic maps. Specifically, the OCSFO algorithm employs Chebyshev, Circle, Logistic, Sine, and Tent chaotic maps to guide individual production and algorithm execution. The novelty of this research lies in the integration of chaotic dynamics into the SFO framework, enhancing its exploratory capabilities and potentially improving convergence rates and solution quality. The OCSFO algorithm was applied to solve various optimization problems, including benchmark test functions and practical applications such as parameter tuning in machine learning models and optimizing design parameters in engineering systems. To evaluate the performance of the proposed OCSFO, both restricted and unrestricted test functions were utilized, providing a comprehensive assessment of its effectiveness. Comparative results demonstrate that the OCSFO achieves competitive outcomes compared to the classical SFO, underscoring its potential as a robust optimization tool. This work is highly relevant as it contributes to the ongoing evolution of metaheuristic algorithms, offering a new approach to optimization that leverages the strengths of chaotic systems. The findings of this study provide valuable insights and pave the way for further research and development in the field of metaheuristic optimization and its applications in artificial intelligence and engineering.

Keywords: Machine Learning Models, Metaheuristic Algorithms, Sunflower Optimization Algorithm, Chaotic Maps, Optimization.

1. INTRODUCTION

Plant intelligence-based heuristic algorithms have established their effectiveness in solving combinatorial and nonlinear problems. Extensive research over the years has revealed that plants exhibit intelligent behaviours and possess complex systems that resemble a nervous system. For instance, plant roots transmit data about light and toxins to their growth centres, and plants interact with their environment through electric currents, as seen in their defence mechanisms against aphids or caterpillars [1].

The SFO Algorithm, inspired by the movement of sunflowers towards the sun, has found diverse applications. Qais et al. [2] used the SFO Algorithm for parameter selection in photovoltaic module modelling and simulation, achieving notable results with specific pollination and elimination rates. Shaha Al-Otaibi et al. [3] applied the SFO Algorithm, a novel approach called SFO-CORP, which addresses the issue of short-lived sensor nodes in wireless sensor networks. Hussein et al. [4] used the SFO Algorithm for PI controller parameter selection, where it outperformed the Particle Swarm Optimization (PSO)

algorithm. Tabibi et al. [5] introduced a self-adaptive SFO Algorithm to optimize parameters for proton exchange membrane fuel cells, minimizing error values significantly. Shaheen et al. [6] employed the SFO Algorithm to solve optimal power flow problems in power systems, achieving better results than both PSO and genetic algorithms. Alshammar [7] developed a chaotic SFO Algorithm using Logistic chaotic mapping for optimal tuning of power system stabilizers, which outperformed several other algorithms across multiple test functions.

Chaotic maps have been extensively integrated into metaheuristic algorithms to enhance their performance by escaping local optima and improving convergence rates. Author in literature [8] combined chaotic maps with the golden section search method, producing a highly effective optimization algorithm for nonlinear problems.

Literature [9] demonstrated that chaotic maps help algorithms avoid local minima more efficiently, leading to superior optimization performance. Demir et al. [10] proposed a hybrid chaotic swarm optimization method using the Logistic-sine map, which showed excellent results on benchmark functions and practical design problems. Author in literature [11] enhanced PSO with the Piecewise Linear Chaotic Map (PWLCM), achieving better global optimization performance. Tian [12] integrated chaotic maps and Gaussian mutation into PSO, avoiding local optima and ensuring thorough exploration of the solution space. Literature [13] utilized various chaotic maps to improve the Whale Optimization Algorithm, achieving high effectiveness across multiple benchmark functions. Pluhacek et al. [14,15] employed chaotic maps to enhance the inertia-weighted PSO algorithm, resulting in significant performance improvements.

Building on these insights, this study proposes the Oscillating Chaotic Sunflower Algorithm (OCSFO), a novel variant that enhances the exploration process through oscillation-based positioning and the use of chaotic maps. This innovative approach allows for flexible and robust search capabilities, achieving competitive results compared to the classical SFO Algorithm. The details of the classical SFO Algorithm and the OCSFO's methodology are presented in the next section. Performance tests, experimental parameters, and results are discussed in the third section, followed by a comprehensive evaluation in the final section.

2. Related Work

The Sunflower Optimization (SFO) Algorithm is part of a broader class of plant intelligence-based heuristic algorithms that have demonstrated their capability in solving complex combinatorial and nonlinear problems. As researchers have delved into plant behaviours, such as heliotropism in sunflowers, these insights have been translated into optimization algorithms with significant success.

2.1 Sunflower Optimization Algorithm and Oscillating Chaotic Sunflower Optimization Algorithm

Mathematical methods used in solving optimization problems can be disadvantageous due to the high computational cost or the difficulty in formulating a mathematical model for every problem. Alternatively, metaheuristics offer a versatile approach that can be applied to various problems without requiring a specific model. The lower computational cost, ease of adaptability, and strong performance of metaheuristics make them a popular choice for many applications. These advantages have led to their application across diverse fields, from engineering to social sciences.

One of the successful metaheuristic optimization algorithms proposed in recent years is the Sunflower Optimization Algorithm (SFO). Inspired by the heliotropic movements of sunflowers, the SFO algorithm mimics the natural behaviour of sunflowers as they follow the sun's rays. Each day, sunflowers reorient themselves to maximize their exposure to sunlight, moving eastward in the morning and westward in the evening. The SFO algorithm uses this behaviour as a metaphor for its optimization process, where each sunflower represents a potential solution.

In the SFO algorithm, individuals (sunflowers) adjust their positions based on their proximity to an optimal solution, referred to as the "sun." Sunflowers close to the optimal solution make finer adjustments, while those farther away make larger moves to get closer. Neighbouring sunflowers also engage in pollination, creating new individuals. Despite the potential for millions of pollinations in the real world, the algorithm simplifies this by assuming each sunflower performs one pollination. The individual closest to the optimal solution becomes the new sun, and a proportion of the population is eliminated to maintain a manageable search space, a process termed the elimination rate. This cycle repeats until a termination criterion is met.

The effectiveness of the SFO algorithm stems from its consideration of the distance from the sun, governed by the inverse square radiation law. According to this law, radiation intensity is inversely proportional to the square of the distance, meaning that sunflowers farther from the sun receive more radiation. This effect is represented by a parameter, (Q_i), which modulates the influence of distance on

the algorithm's operations. The mathematical representation of this relationship involves the power of the source (P) and the distance (r_i) between the (i^{th}) individual and the immediate best solution:

$$Q_i = \frac{P}{4\pi r_i^2} - - - (1)$$

The basic principle in the algorithm is to represent the orientation of sunflowers to the sun. For this, the sun (X^*) in the population is the reference for other sunflowers (X_i). Its orientation, which takes into account the criterion of distance relative to the reference, is expressed by:

$$\vec{S}_I = \frac{X^* - X_i}{||X^* - X_i||}, \qquad i = 1,2,3, \dots, n_p - - - (2)$$

The orientation steps of the population individuals towards the sun are among the critical parameters of the sunflower algorithm. Individuals (i). $(i-1)$ in the orientation velocities of the population individuals. It pollinates with the individual and forms the new individual with random position. This random position is proportional to the distance between individuals. The general expression of the steps towards the sun is given by:

$$d_i = \lambda P_i\left(\left|\left|X_i - X_{i-1}\right|\right|\right)\left|\left|X_i - X_{i-1}\right|\right| - - - (3)$$

One of the main criteria for determining step values is the maximum stride length (d_{max}). This parameter is directly proportional to the Euclidean distance of the default upper bound (X_{max}) and lower bound (X_{max}) according to the problem definition, and inversely proportional to the number of individuals in the population (N_{pop}):

$$d_{max} = \frac{||X_{max} - X_{min}||}{2N_{pop}} - - - (4)$$

Based on these basic parameters, the new individual is calculated as:

$$\vec{X}_{i+1} = \vec{X}_i + d_i\vec{s}_i - - - (5)$$

In the classical SFO Algorithm, individuals that are far from the sun in each iteration are removed from the search process according to the mortality rate, and new individuals are produced in their place. Given the randomness, it is possible that new individuals behave similarly to older individuals.

In this study, the Oscillating Chaotic Sunflower Optimization Algorithm (OCSFO) is proposed to allow the exploration process to be carried out more flexibly. In this version of the algorithm, the search space for a given portion of newly added individuals throughout the iterations is determined based on the population best and the search space maximum. This process involves two main stages. The first stage determines a location between the iteration best and the search space maximum for a specific proportion (Y) of individuals to be added. This location is based on the value obtained from the Gaussian Distribution Function, which uses the midpoint of the two boundaries as the average value:

$$P_f = \frac{X_f^* + X_f^{max}}{2} - - - (6)$$

$$L_f = N\left(P_f, \sigma^2\right) - - - (7)$$

A trigonometric approach is used for generating candidate values around the specified location centre, allowing a more flexible discovery process. New candidate values introduced into the population in iteration (i) are identified using:

$$X_f^{new} = L_f\left(1 + \eta Cos\left(\frac{2\pi i}{\beta}\right)\right) - - - (8)$$

Here, (η) refers to the value returned from the chaotic map function used, and (β) is a predefined application constant. The identification of other individuals who did not use this function and who recently joined the population was carried out according to the classical SFO. However, chaotic map functions were used in all candidate productions (including the initial population). The study used well-known chaotic map functions to evaluate the performance of different chaotic variations of the OCSFO. Algorithm 1 uses the pseudocode of the OCSFO Algorithm, with the chaotic map selection indicated by the (k) parameter. In the algorithm, Equations 6-8 are executed in the function of subtracting individuals away from the sun. This function uses the parameters i, Y, σ, and β.

Algorithm 1. OCSFO Algorithm pseudocode

1. X_{max}, X_{min} ← Problem-defined upper limit and lower limits
2. Itr_{max} ← Maximum number of iterations
3. N_{pop} ← population size.
4. k ←Chaotic generator function type
5. O ← population removal rate (%)
6. **for** $i = 1$' then N_{pop}
7. X_i= IndividualGeneratorFunction (k, X_{max}, X_{min})
8. add to population (X_i)
9. **end for**
10. Calculate objective()
11. X^* ← findsun()
12. RedirectYour Sunflowers to the Sun()
13. **while** ($i < Itr_{max}$)
14. **for** j=1'then $Npop$
15. CalculateVector of Individuals()
16. SubtractIndividualsAway from the Sun$(\%O, \%Y, i, \sigma\ \beta)$.
17. EvaluateNewIndividuals(X^*)
18. **end for**
19. calculateobjective.
20. X^* ←findsun()
21. **End while**
22. ShowBestSolution()

Chaotic maps are the randomness of a mathematically simple deterministic dynamic system, and the chaotic system can be considered a source of randomness. In this study, it was examined whether more efficient results could be obtained from SFO application by using chaotic maps. These chaotic maps have been used in individual production and movement behaviours. Table 1 shows the chaotic maps used in the study. All chaotic maps used are added to the OCSFO Algorithm and enabled by a function selection parameter. Thus, similar to the studies in [16,17] the OCSFO Algorithm being run is named with the chaotic map used (such as tentOCSFO).

3.Simulation and Analysis

In this section, we present the performance results of the chaotic-based Sunflower Optimization Algorithms (SFOs), specifically focusing on the Oscillating Chaotic Sunflower Optimization (OCSFO) variants. The performance of these chaotic-based algorithms is compared with that of the classical Sunflower Algorithm (SFO). Various well-known test functions, as listed in Table 2, were used to evaluate the performance. The test functions include both unrestricted functions (Sphere, Rastrigin, Nonlinear) and restricted functions (Rosenbrock Disk, Rosenbrock Cubic/Line). For each test function, the problem sizes were as follows: five for the Sphere function, three for the Rastrigin function, and two for the other functions. The individual size and the number of internal iterations were set to 20, with the number of solution iterations chosen as 5, resulting in a total of 95 iterations. The pollination rate (p) was set to 0.05 and the elimination rate (O) to 0.1. The performance of the algorithms was evaluated across 20 independent experiments. The results are analyzed in terms of the minimum values obtained over iterations, with a focus on comparing the classical SFO and the chaotic-based SFOs.

Table 1. Chaotic Maps used in OCSFO variants.

Name	Function	Parameter
Sine	$X_{n+1} = \frac{a}{4}\sin(\pi X_n)$	$0 < a \leq 4$
Logistic	$X_{n+1} = \mu X_n(1 - X_n)$	$0 < \mu \leq 4$
Chebyshev	$X_{n+1} = \cos(k\cos^{-1}X_n)$	$k \geq 2$
Circle	$X_{n+1} = X_n + 0.2 - \left(\frac{1}{4\pi}\right)\sin(2\pi X_n) mod(1)$	
Tent	$X_{n+1} = \begin{cases} \mu X_n, & X_n < \frac{1}{2} \\ \mu(1 - X_n), & \frac{1}{2} \leq X_n \end{cases}$	

Table 2. Test Functions used for Performance Evaluation.

Name	Function Equation	Parameter	Min
Sphere	$f(x) = \sum_{i=1}^{N} x_i^2$	$-100 < x_i < 100$	0
Rastrigin	$f(x) = 10N + \sum_{i=1}^{N} [x_i^2 - 10\cos(2\pi x_i)]$	$-5.12 < x_i < 5.12$	0
Camel	$f(x) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	$-200 < x_i < 200$	-1.031
Nonlineer	$f(x) = x_1^2 - 3x_1x_2 + 4x_2^2 + x_1 - x_2$	$-2 < x_i < 2$	-0.28
Rosenbrock Cubic/Line	$f(x,y) = (1-x)^2 + 100(y-x^2)^2$	$(x-1)^3 - y + 1 \le 0$ $ve\ x + y - 2 \le 0$	0
Rosenbrock Disk	$f(x,y) = (1-x)^2 + 100(y-x^2)^2$	$x^2 + y^2 \le 2$	0

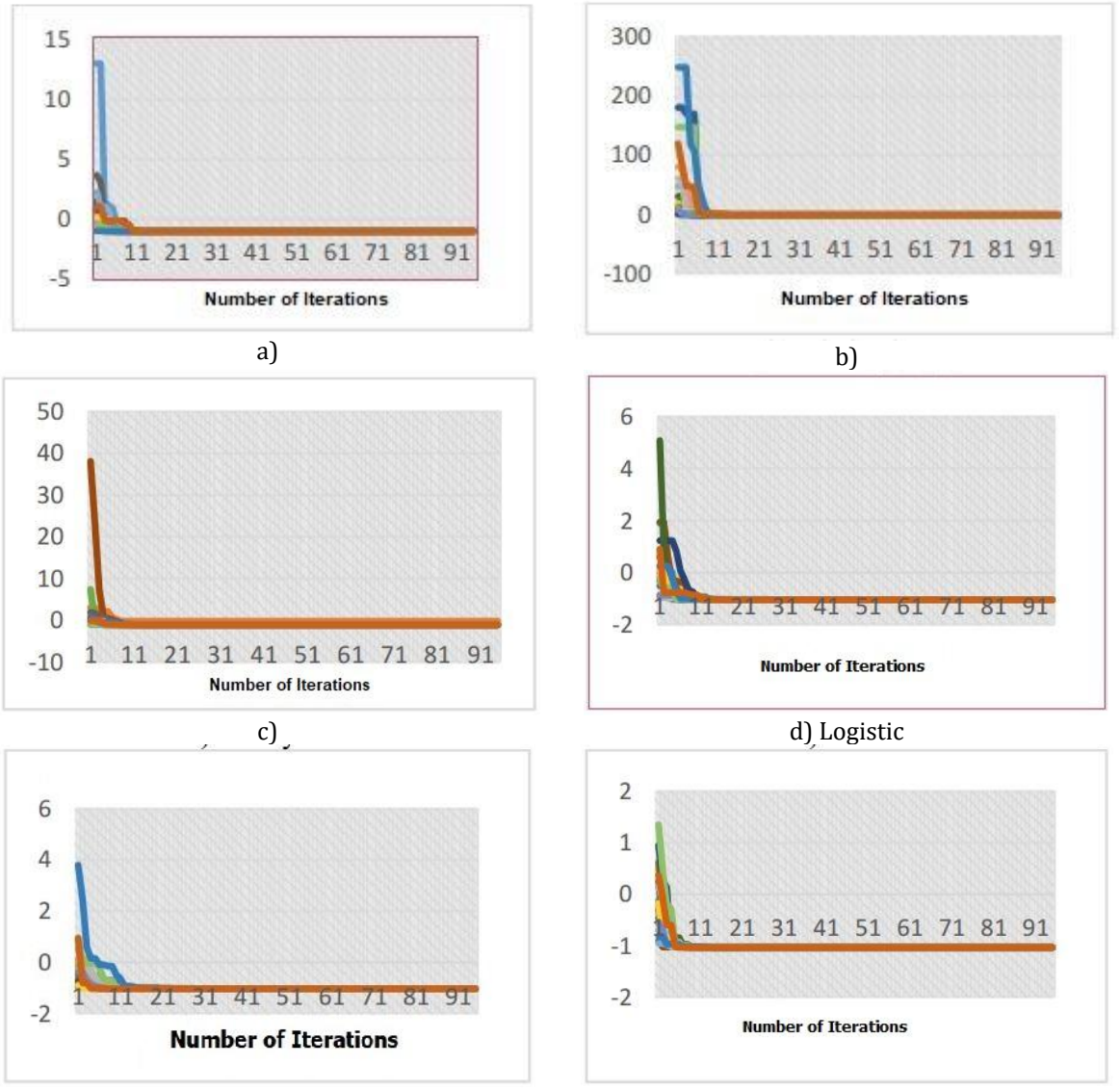

Figure 1. Minimum variations of the conformity value according to iterations for the Camel Test Function, a) Classic Sunflower, b) Chebyshev, c) Circle, d) Logistic, e) Sine, f) Tent.

Camel Test Function

Figure 1 illustrates the iteration-wise minimum values obtained for the Camel test function. The classical SFO and various chaotic-based SFOs were evaluated, including Chebyshev, Circle, Logistic, Sine, and Tent. The results show that the average number of iterations to achieve 99% of the minimum value was 17. Specifically, TentOCSFO achieved this in an average of 13 iterations, making it the most effective among the chaotic-based methods, followed by CircleOCSFO and LogisticOCSFO with 16 iterations each. ChebyshevOCSFO and SineOCSFO required more iterations, with averages of 19 and 25, respectively.

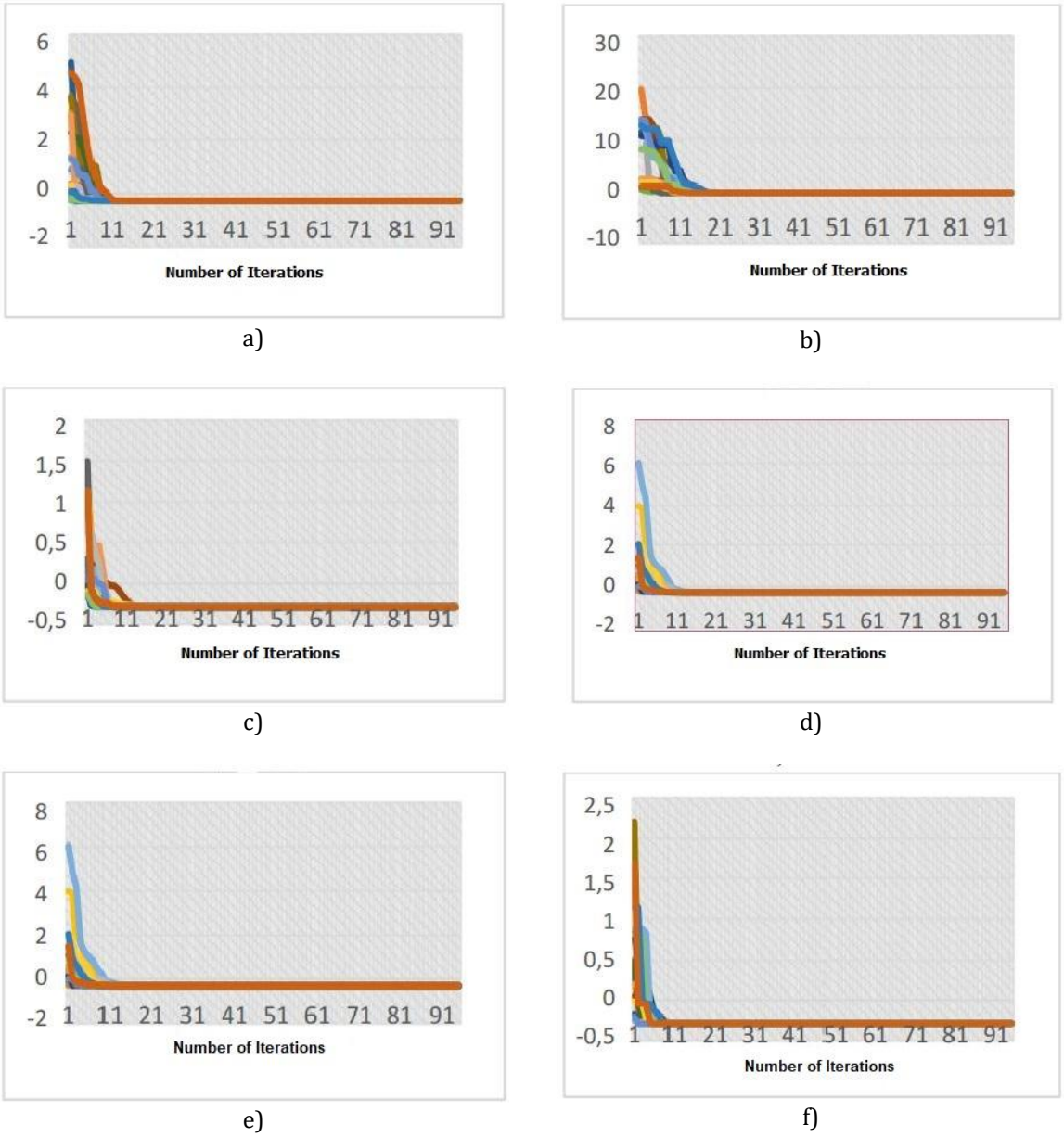

Figure 2. Minimum variations of the conformity value according to iterations for the Nonlinear Test Function, a) Classic Sunflower, b) Chebyshev, c) Circle, d) Logistic, e) Sine, f) Tent.

Nonlinear Test Function

Performance results for the Nonlinear test function are shown in Figure 2. The classical SFO reached the minimum value in an average of 8 iterations. Among the chaotic-based SFOs, TentOCSFO and CircleOCSFO performed the best, reaching the minimum in 6 and 7 iterations, respectively. ChebyshevOCSFO and LogisticOCSFO took longer, achieving the minimum in 11 and 10 iterations, respectively. The overall performance highlights TentOCSFO as the most efficient chaotic-based method for this function.

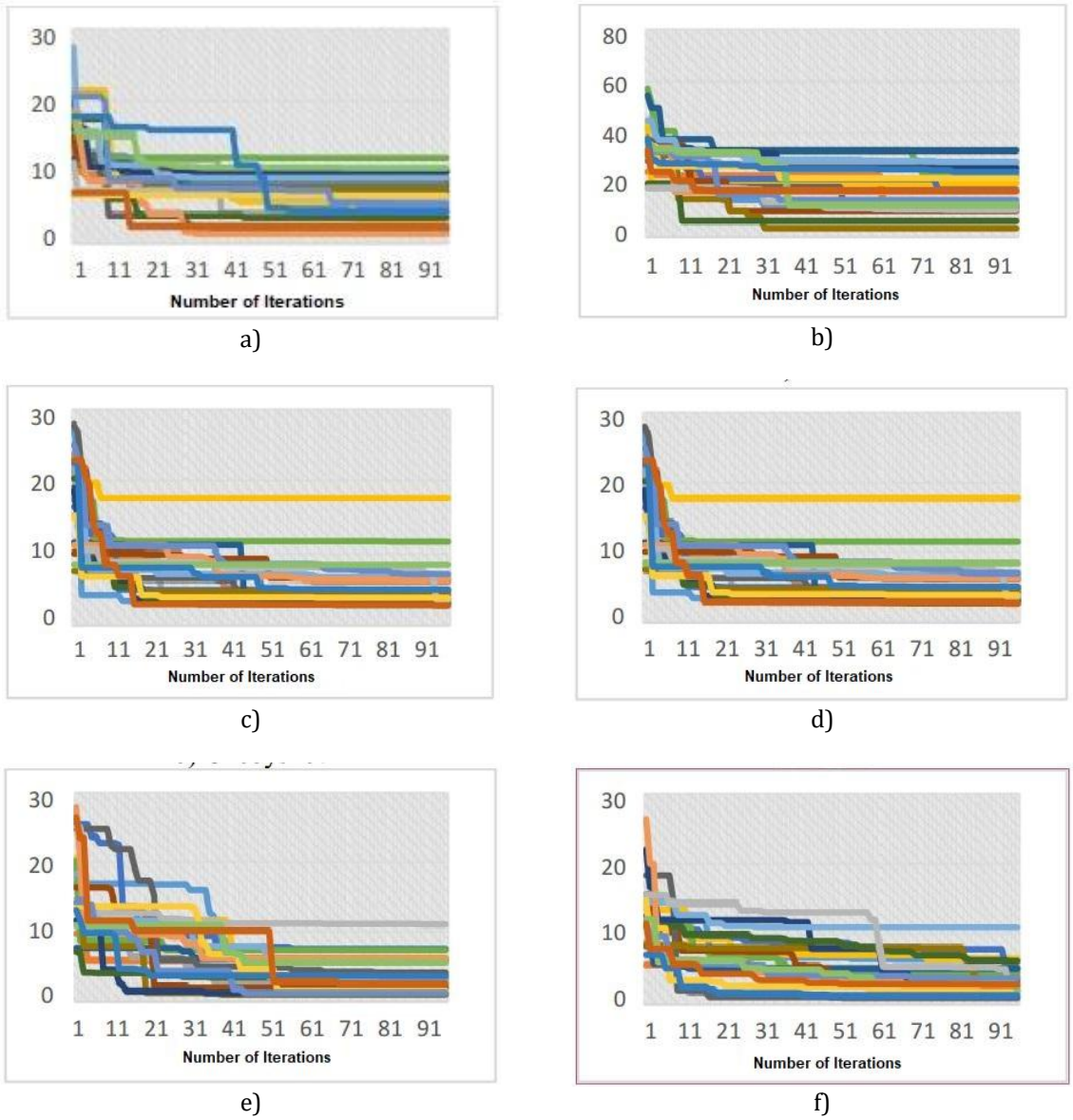

Figure 3. Minimum variations of the conformity value according to iterations for the Rastrigin Test Function, a) Classic Sunflower, b) Chebyshev, c) Circle, d) Logistic, e) Sine, f) Tent.

Rastrigin Test Function

Figure 3 displays the iteration-minimum results for the Rastrigin test function. This function, known for its multiple local minima, showed that the classical SFO achieved a minimum value of 1.471 at the 32nd iteration. TentOCSFO performed well, achieving a minimum of 1.012 by the 17th iteration, making it one of the top performers. LogisticOCSFO achieved a minimum of 0.885 but required more iterations (65). CircleOCSFO and ChebyshevOCSFO also performed competently but fell short compared to TentOCSFO.

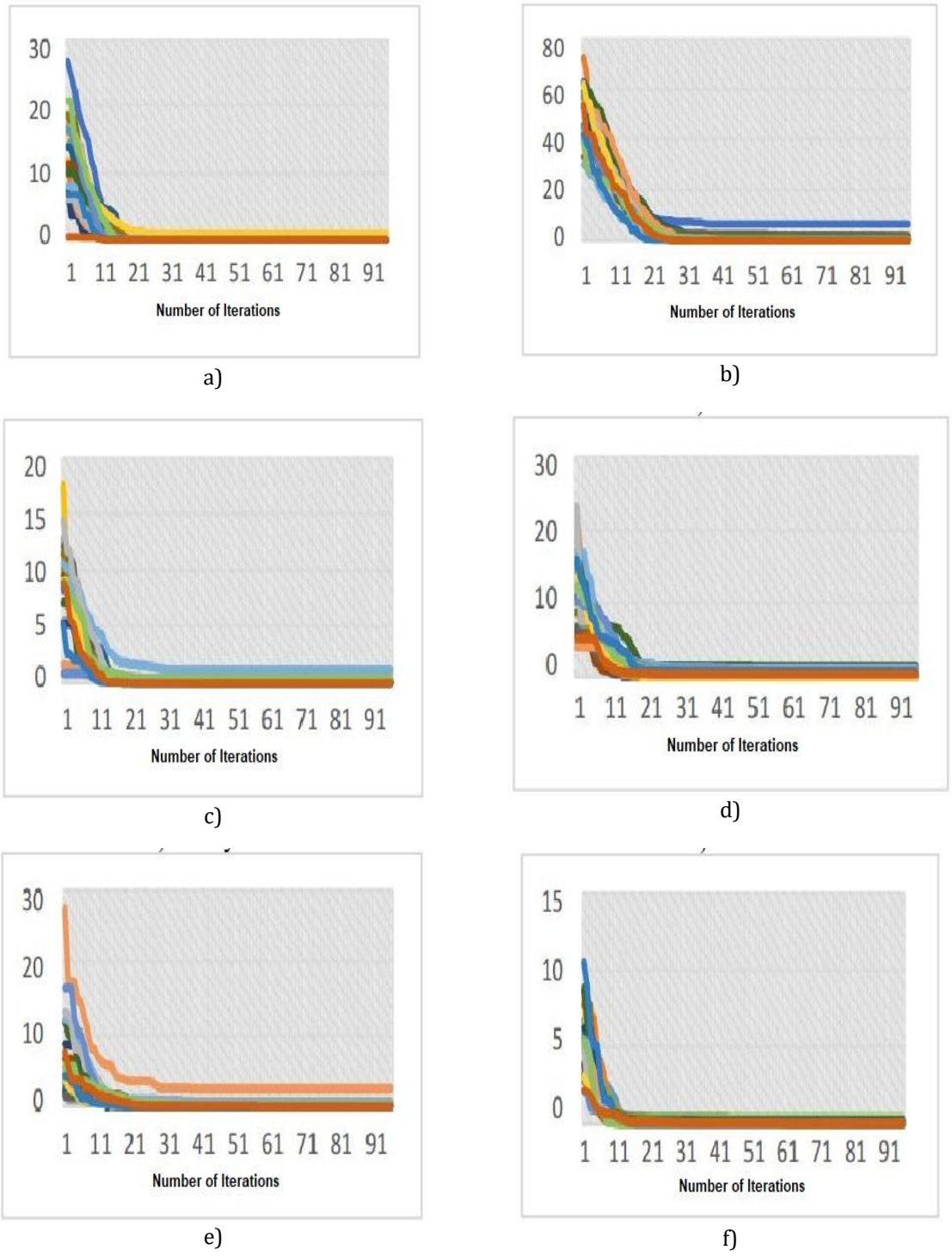

Figure 4. Minimum variations of the conformity value according to iterations for the Sphere Test Function, a) Classic Sunflower, b) Chebyshev, c) Circle, d) Logistic, e) Sine, f) Tent.

Sphere Test Function

Figure 4 presents the results for the Sphere test function, where the actual minimum value is 0. Here, TentOCSFO and SineOCSFO achieved the closest values to the minimum, with averages of 19 and 43 iterations, respectively. The classical SFO showed similar performance, reaching the minimum in an average of 22 iterations. While TentOCSFO performed closely to the classical SFO, ChebyshevOCSFO and LogisticOCSFO required more iterations.

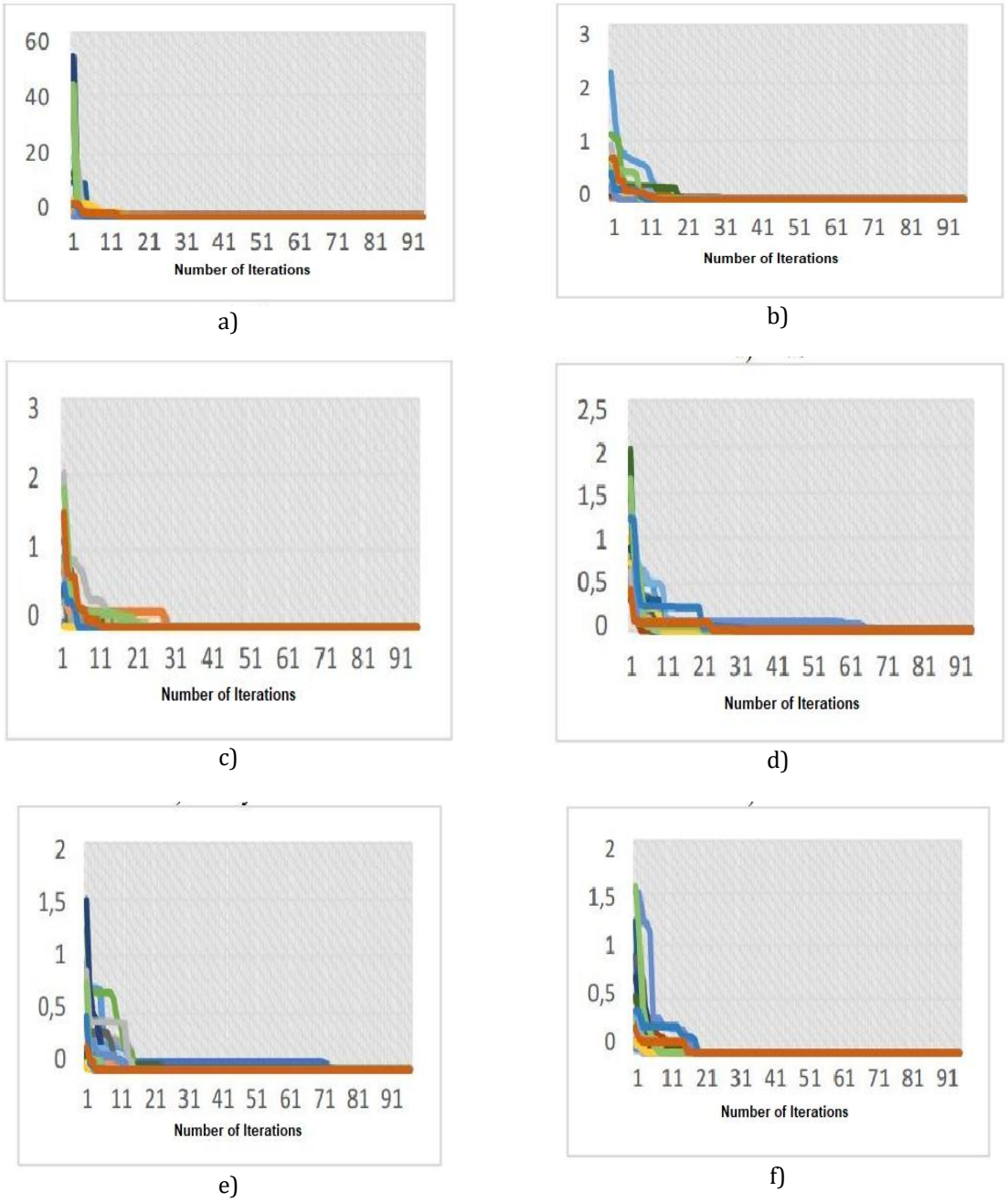

Figure 5. Minimum variations of the conformity value according to iterations for the Rosenbrock Disk Test Function, a) Classic Sunflower, b) Chebyshev, c) Circle, d) Logistic, e) Sine, f) Tent.

Rosenbrock Disk Test Function

The results for the Rosenbrock Disk test function are shown in Figure 5. The classical SFO reached the minimum in an average of 9 iterations. The chaotic-based methods showed similar performance, with TentOCSFO and ChebyshevOCSFO achieving the minimum in 10 and 10 iterations, respectively. The other methods also performed well, though TentOCSFO and ChebyshevOCSFO were slightly less efficient in comparison to the classical SFO.

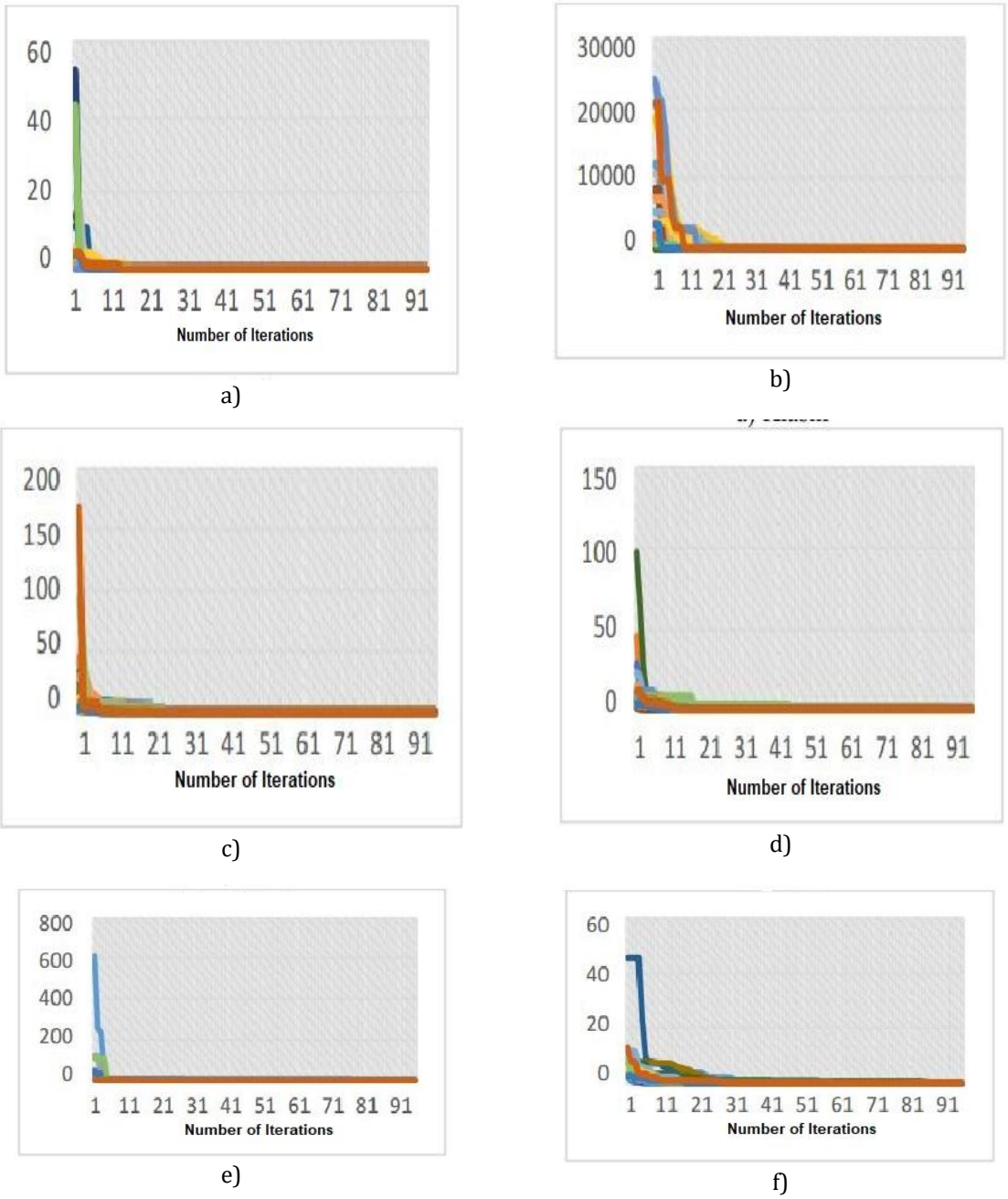

Figure 6. Minimum variations of the conformity value according to iterations for the Rosenbrock Cubic/Line Test Function, a) Classic Sunflower, b) Chebyshev, c) Circle, d) Logistic, e) Sine, f) Tent.

Rosenbrock Cubic/Line Test Function

Figure 6 presents the results for the Rosenbrock Cubic/Line test function. The classical SFO demonstrated superior performance, reaching the minimum in an average of 10 iterations. Chaotic-based SFOs generally required more iterations, with ChebyshevOCSFO and TentOCSFO needing 53 and 47 iterations, respectively. LogisticOCSFO and CircleOCSFO also showed good results but were less effective compared to the classical SFO.

Table 3. Statistical results for all experiments

		Camel	Nonlineer	Rastrigin	Sphere	R.Disk	R. Cubic/Line
	Min	-1.031	-0.285	1.471	2.812E-05	2.938E-07	3.420E-08
	Max	-1.027	-0.285	12.046	1.029	0.028	5.369E-07
Classical	Mean	-1.031	-0.285	6.416	0.094	0.001	9.042E-05
	Median	-1.031	-0.285	6.479	0.019	5.437E-06	1.858E-05
	Std.Deviation	1.000E-03	3.519E-06	3.014	0.226	0.006	8.745E-07
	Min	-1.031	-0.285	3.554	0.014	3.635E-08	5.997E-07
	Max	-1.031	-0.285	33.335	6.490	6.193E-05	6.776
Chebyshev	Mean	-1.031	-0.285	17.778	0.890	1.001E-05	0.762
	Median	-1.031	-0.285	17.572	0.399	5.116E-06	6.036E-07
	Std.Deviation	7.43E-05	8.59E-06	7.966	1.408	1.523E-05	1.857
	Min	-1.031	-0.285	2.666	3.951E-05	3.611E-08	5.484E-07
	Max	-0.215	-0.285	17.727	1.142	4.778E-05	0.001
Circle	Mean	-0.988	-0.285	5.684	0.142	8.096E-06	9.958E-05
	Median	-1.031	-0.285	4.793	0.067	3.477E-06	5.888E-06
	Std.Deviation	1.820E-01	1.869E-06	3.502	0.249	1.299E-05	1.541E-07
	Min	-1.031	-0.285	0.885	0.001	6.099E-08	4.551E-07
	Max	-1.031	-0.285	11.396	1.326	1.19E-07	0.952
Logistic	Mean	-1.031	-0.285	4.704	0.333	2.791E-05	0.070
	Median	-1.031	-0.285	4.028	0.124	3.688E-06	1.608E-05
	Std.Deviation	1.889E-05	2.154E-06	3.303	0.390	6.675E-05	0.224
	Min	-1.031	-0.285	1.026	1.14E-07	4.312E-08	5.800E-07
	Max	-1.031	-0.285	11.0422	2.441	0.008	3.211
Sine	Mean	-1.031	-0.285	4.258	0.236	2.325E-07	0.160
	Median	-1.031	-0.285	3.662	0.032	4.254E-06	5.806E-06
	Std.Deviation	3.884E-05	6.959E-06	2.590	0.537	0.001	0.699
	Min	-1.031	-0.285	1.012	1.02E-07	9.126E-08	5.518E-08
	Max	-1.031	-0.285	10.909	0.291	0.002	0.110
Tent	Mean	-1.031	-0.285	3.909	0.042	3.902E-07	0.008
	Median	-1.031	-0.285	3.492	0.012	2.793E-06	2.361E-05
	Std.Deviation	2.574E-05	3.693E-06	2.356	0.068	5.632E-07	0.025

4.Statistical Summary

Table 3 summarizes the statistical results for all experiments. For the Camel function, chaotic-based SFOs, particularly ChebyshevOCSFO, LogisticOCSFO, SineOCSFO, and TentOCSFO, showed better performance compared to the classical SFO, although CircleOCSFO and classical SFO were slightly behind. For the Nonlinear function, all methods performed well, with minimal differences between them. For the Rastrigin function, LogisticOCSFO and TentOCSFO outperformed others, with TentOCSFO being the best. In the Sphere function, SineOCSFO and TentOCSFO were closest to the minimum value. For the Rosenbrock Disk, all methods performed well, but classical SFO and TentOCSFO were slightly less efficient. Finally, for the Rosenbrock Cubic/Line, classical SFO was the most effective, with chaotic-based methods trailing behind.

Overall, the results demonstrate that the chaotic-based SFOs, particularly TentOCSFO, generally provide competitive and often superior performance compared to the classical SFO. ChebyshevOCSFO, while effective, showed slightly lower performance in comparison to other chaotic variants.

5.CONCLUSIONS

In this paper, the Oscillating Chaotic Sunflower Optimization (OCSFO) Algorithm is proposed, introducing a more flexible exploration process to the traditional Sunflower Optimization Algorithm (SFO) by integrating chaotic and trigonometric approaches. The performance of the proposed OCSFO algorithm

was evaluated using various chaotic maps—Circle, Logistic, Chebyshev, Tent, and Sine—and tested on a set of well-known benchmark functions, including both unrestricted (Nonlinear, Camel, Rastrigin, Sphere) and restricted (Rosenbrock Disk, Rosenbrock Cubic/Line) test functions.

The experimental results demonstrate that the OCSFO algorithm produces highly competitive results across different test functions. In particular, the Tent-based OCSFO variant showed superior performance in terms of reaching minimum values more quickly and effectively for several test functions, including the Camel, Nonlinear, and Sphere functions. The Logistic-based OCSFO also performed well, especially for the Rastrigin function. Although the Chebyshev-based OCSFO showed slightly lower performance compared to other chaotic variants, it still demonstrated competitive results.

Statistical analyses confirmed that the chaotic-based SFOs, particularly the TentOCSFO, offer a significant improvement over the classical SFO, making them a viable alternative for optimization tasks. The chaotic maps introduced an additional layer of randomness and flexibility, enhancing the exploration capabilities of the algorithm and preventing premature convergence.

Future work will focus on further enhancing the OCSFO algorithm by exploring rule inference-based optimization and sentiment analysis applications. Additionally, there is potential for investigating other chaotic maps and trigonometric approaches to further improve the performance and applicability of the OCSFO algorithm in various optimization problems.

REFERENCE

[1] Alshammari, B.M., Guesmi, T. New Chaotic Sunflower Optimization Algorithm for Optimal Tuning of Power System Stabilizers. J. Electr. Eng. Technol. 15, 1985–1997 (2020). https://doi.org/10.1007/s42835-020-00470-1.

[2] Qais MH, Hasanien HM, Alghuwainem S. Identification of electrical parameters for three-diode photovoltaic model using analytical and sunflower optimization algorithm. Applied Energy 2019; 250: 109-117. Doi:10.1016/j.apenergy.2019.05.013.

[3] Al-Otaibi, S.; Cherappa, V.; Thangarajan, T.; Shanmugam, R.; Ananth, P.; Arulswamy, S. Hybrid K-Medoids with Energy-Efficient Sunflower Optimization Algorithm for Wireless Sensor Networks. Sustainability 2023, 15, 5759. https://doi.org/10.3390/su15075759.

[4] AM. Hussien, HM. Hasanien, SF. Mekhamer, Sunflower optimization algorithm-based optimal PI control for enhancing the performance of an autonomous operation of a micro-grid. Ain Shams Engineering Journal 2021; Doi: 10.1016/j.asej.2020.10.020.

[5] Tabibi, S.; Ghaffari, A. Energy-Efficient Routing Mechanism for Mobile Sink in Wireless Sensor Networks Using Particle Swarm Optimization Algorithm. Wirel. Pers. Commun. 2019, 104, 199–216.

[6] Shaheen MAM, Hasanien HM, Mekhamer SF, Talaat HEA. Optimal Power of Power Systems Including Distributed Generation Units Using Sunflower Optimization Algorithm. IEEE Access 2019; 7: 109289-109300. Doi:10.1109/Access.2019.2933489.

[7] Alshammari, B. M., & Guesmi, T. (2020). New Chaotic sunflower optimization algorithm for optimal tuning of power system stabilizers. Journal of Electrical Engineering and Technology. https://doi.org/10.1007/s42835-020-00470-1

[8] Tawhid, M.A., Ibrahim, A.M. Improved salp swarm algorithm combined with chaos. Mathematics and Computers in Simulation, Volume 202, December 2022, Pages 113-148. https://doi.org/10.1016/j.matcom.2022.05.029.

[9] Liu, Q.; Li, N.; Jia, H.; Qi, Q.; Abualigah, L.; Liu, Y. A hybrid arithmetic optimization and golden sine algorithm for solving industrial engineering design problems. Mathematics 2022, 10, 1567.

[10] Demir FB, Tuncer T, Kocamaz AF. A chaotic optimization method based on logistic-sine map for numerical function optimization. Neural Computing and Applications 2020; 32: 14227-14239. Doi: 10.1007/s00521-020-04815-9.

[11] Wen, C.; Jia, H.; Wu, D.; Rao, H.; Li, S.; Liu, Q.; Abualigah, L. Modified remora optimization algorithm with multistrategies for global optimization problem. Mathematics 2022, 10, 3604.

[12] Xiao, Y.; Sun, X.; Guo, Y.; Cui, H.; Wang, Y.; Li, J.; Li, S. An enhanced honey badger algorithm based on Lévy flight and refraction opposition-based learning for engineering design problems. J. Intell. Fuzzy Syst. 2022, 43, 4517–4540.

[13] Yang, J.; Liu, Z.; Zhang, X.; Hu, G. Elite chaotic manta ray algorithm integrated with chaotic initialization and opposition-based learning. Mathematics 2022, 10, 2960.

[14] Pluhacek M, Senkerik R, Zelinka I. Impact of Various Chaotic Maps on the Performance of Chaos Enhanced PSO Algorithm with Inertia Weight-An Initial Study. Nostradamus: Modern Methods of Prediction, Modeling and Analysis of Nonlinear Systems, Advances in Intelligent Systems and Computing; 2013; Berlin. 153-166. Doi: 10.1007/978-3-642-33227-2_18

[15] Jia, H.; Sun, K.; Zhang, W.; Leng, X. An enhanced chimp optimization algorithm for continuous optimization domains. Complex Intell. Syst. 2021, 8, 65–82.
[16] Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The arithmetic optimization algorithm. Comput. Meth. Appl. Mech. Eng. 2021, 376, 113609.
[17] Xiao, Y.; Guo, Y.; Cui, H.; Wang, Y.; Li, J.; Zhang, Y. IHAOAVOA: An improved hybrid aquila optimizer and African vultures optimization algorithm for global optimization problems. Math. Biosci. Eng. 2022, 19, 10963–11017.