Kurdistan Journal of Applied Research (KJAR)
Print-ISSN: 2411-7684 | Electronic-ISSN: 2411-7706

Original Article

CLFOX: A Cubic–Lévy Flight Enhanced FOX Optimizer for Global Numerical Optimization and Engineering Design Problems

Sima A. Salih [a] *, **Hardi M. Mohammed** [a], **Zrar Kh. Abdul** [a]

[a] Department of Computer Science, College of Science, Charmo University, 46023 Chamchamal, Sulaimani, Kurdistan Region, Iraq.

Submitted: 04 May 2026
Revised: 22 August 2026
Accepted: 25 August 2026

*** Corresponding Author:**
sema.abdulla@chu.edu.iq

Keywords: Engineering design optimization, Chaotic maps, Exploration enhancement, Lévy flight, Metaheuristic optimization, Three-bar truss.

How to cite this paper: S. A. Salih, H. M. Mohammed, Z. K. Abdul, "CLFOX: A Cubic–Lévy Flight Enhanced FOX Optimizer for Global Numerical Optimization and Engineering Design Problems", KJAR, vol. 11, no. 2, pp. 51-87, December 2026, doi: 10.24017/science.2026.2.3

Abstract: FOX is a recent powerful metaheuristic optimization algorithm, motivated by a red fox's behavior inside snow for catching prey. Although FOX is one of the more powerful algorithms, it has some limitations, including weak exploration, getting trapped inside local optima, and having unbalanced exploration and exploitation. This study proposes a Cubic–Lévy flight enhanced FOX optimizer (CLFOX), which is an improved version of FOX, to overcome its limitations. Four new techniques have been added to CLFOX to improve FOX's performance. The first technique is initializing the population using a chaotic map. Second is introducing Lévy flight as another exploration strategy. Third is increasing the main condition of the algorithm to keep the balance between each strategy that happens inside the algorithm. Finally, an interval-based operator is added that runs once every 10 iterations. To show the importance of CLFOX, three sets of benchmark test functions are used: the classical benchmark function set, the CEC2019 set, and the CEC2022 set. The results show that CLFOX can outperform FOX in 42 out of 45 benchmark testing functions. CLFOX was also compared to nine competitive algorithms, including FOX. The results show that CLFOX outperforms almost all the other algorithms and takes the first rank among them. The statistical results show that 88% of the improved values are significant. CLFOX is also used to solve real-world engineering challenges, including pressure vessel design, tension/string design, three-bar truss design, gear train design, cantilever beam design, and welded beam design, to show its performance in comparison to the other algorithms.

1. Introduction

Nowadays, meta-heuristic (MH) optimization techniques are frequently employed to deal with challenging real-world problems [1]. As these problems become more sophisticated, the demand for novel and more efficient MH algorithms is becoming increasingly critical [2]. Optimization techniques known as MH algorithms seek to identify the optimal solution to a given issue. Stated differently, MHs are a collection of clever techniques able to increase the effectiveness of heuristic processes and they have demonstrated strong effectiveness in addressing large-scale optimization challenges [3].

To put it straightforwardly, optimization is the art of perfectionism that focuses on creating things in the best possible way. The challenge of how to identify the ideal solution to a problem while imposing a set of constraints is addressed by optimization [4]. The majority of MH algorithms adhere to a common framework: they initiate by producing a random population of potential solutions and subsequently evolve these candidates iteratively through processes such as recombination, movement, or

mutation. There are two main parts to the process: exploration, which looks for new places in the search space, and exploitation, which improves the solutions in places that look promising [5].

MH algorithms have been successfully implemented across different domains [6], comprising engineering problems [7, 8], vehicle routing problems [9], finance tasks [10], medical applications [11], healthcare scheduling cases [12], economic problems [13], computer science problems [14, 15], and machine learning problems [16, 17].

The FOX optimization algorithm is among the most significant and recent developments, inspired by how the red fox hunts in the snow [1]. This algorithm mimics how foxes hear the sound of prey and then jump in a planned way to catch it. The exploration phase is the fox's global search for food, and the exploitation phase is the fox figuring out how far away its target is and jumping toward it, which becomes the two important parts of the algorithm [1, 18].

The FOX algorithm is great at finding a solution for hard optimization problems and often surpasses other algorithms, but it does have some flaws. Finding these flaws is important for making it better in the future. One of the biggest problems with the FOX algorithm is that it can't achieve a proper balance between exploring and using the information [18]. The algorithm gives each phase a 50% chance by using a random control variable, but the way exploration is done in practice can make things less diverse. During the algorithm's exploration stage, search agents tend to group together in the same area because they use the best solution as a reference point. This makes exploration less random, which stops the agents from thoroughly searching the larger search space. Another major problem is that agents tend to get stuck in local optima [19, 20]. This occurs when the algorithm recognizes a suboptimal solution and inaccurately considers it to be the global optimum. This issue frequently arises from the inadequate global search coverage already noted, which limits the agents' capacity to bypass local minima.

To address these limitations, this work presents an enhanced version of the FOX algorithm, termed the Cubic–Lévy Flight FOX (CLFOX) optimizer, which combines chaotic dynamics with Lévy flight movement to increase randomness and drive the agents toward more distant regions of the search space, thereby strengthening exploration and helping the algorithm avoid local optima. The main contributions of this paper are summarized as follows:

- Proposing CLFOX, which enhances the FOX optimizer through chaotic-map initialization, where thirteen maps are examined and the most effective one is selected, together with a Lévy flight mechanism to strengthen exploration.
- Improving the search behavior by reformulating the main condition logic to rebalance the exploitation and exploration phases, and incorporating an interval-based update strategy to prevent stagnation.
- Evaluating CLFOX on three benchmark function sets, namely the classical, CEC2019, and CEC2022 sets, and comparing it against the original FOX algorithm and eight other recent high-performing algorithms.
- Validating CLFOX on six engineering design problems, supported by statistical analysis to confirm the effectiveness of the proposed improvements.

The following sections of this work are organized as follows. Section two shows some of the related works done on the FOX optimization algorithm. Section three looks at the material and methods including the original FOX optimization algorithm and proposed CLFOX algorithm, and identifies each strategy that was used to improve the algorithm. Section four shows the proposed algorithm's results and compares it to other algorithms. The discussion and future work explanation are in section five. Finally, section six contains the conclusion.

2. Related Works

The FOX optimization method has obtained significant interest because of its simplicity and robust efficacy in addressing diverse optimization challenges. This algorithm has become a notable title in optimization topics among researchers. Several studies have attempted to enhance the algorithm, to hybridize it with other algorithms or techniques, or to apply it to real-world problems across various fields. Some of these previous works are reviewed in this section.

Jumaah *et al.* [18] proposed an enhancement of FOX called Improved FOX Optimization Algorithm (IFOX), which was developed to rectify the significant shortcomings in the original FOX algorithm, notably that it relies on a fixed exploration and exploitation ratio, thereby limiting adaptability throughout various optimization challenges. To address this, IFOX implements an adaptive mechanism that dynamically chooses between exploration and exploitation according to the fitness values of potential solutions at each iteration, instead of relying on preset probabilities. It simplifies and clarifies the fundamental equations by substituting unnecessarily complex or unclear formulas and minimizing the number of hyperparameters to enhance parameter tuning efficiency. Notably, adaptive parameters are used to increase solution diversity and enhance search robustness, while the process of computing candidate solutions becomes more stable and focused. These changes enable IFOX to respond more effectively to the particular landscape of the optimization problem, improving convergence speed and solution quality. This adaptivity entails a moderate increase in computational expense, since the fitness function needs to be assessed twice per agent per iteration.

In another study, Wang *et al.* [19] proposed the improved FOX optimizer, opposition-based learning 4 FOX (OBL4FOX), to improve the global search ability and the convergence performance of the original FOX algorithm. The proposed method uses an opposition-based learning mechanism for generating opposite solutions to enlarge the search space and enhance the global exploration. Four nonlinear dynamic update strategies are proposed to enhance the core control variable of FOX, which provide more adaptability to explore complex search spaces. The performance of OBL4FOX was tested on the CEC2022 benchmark functions and the results demonstrated that OBL4FOX performed better than the original FOX and other optimization algorithms. The same algorithm was also applied to engineering optimization problems like pressure vessel, three-bar truss and welded beam design problems where it resulted in competitive values for the objective functions within the respective engineering constraints.

Subsequently, Zhang *et al.* [20] created another modified version of FOX called adaptive spiral flight FOX optimization algorithm ASFFOX, that was proposed to overcome some of FOX's limitations comprising: the weak ergodicity of agents, restricted population diversity and a propensity to enter local optima during complicated, high-dimensional optimization issues. To overcome these issues, ASFFOX incorporates multi-strategy enhancements. Population is initialized by a tent chaotic map with more diverse and well-distributed individuals, improving exploration from the outset, adaptively adjusting inertia weights to strike a harmony among global search and local exploitation, enhancing accuracy and convergence speed. It introduces the Lévy flight technique to aid in the population's departure from local optima by performing random, long-distance movements, and adopts a technique for updating the variable spiral position to add flexibility and richness to the search paths. Collectively, these improvements lead to a better performance and faster, more stable, convergence compared to the original FOX.

Recently, Jumaah *et al.* [21] introduced a new variant of the FOX optimization algorithm, called improved FOX (IFOX), to the literature, which tries to balance exploitation and exploration but not through static probabilities, which becomes a significant limitation in FOX. Furthermore, the original FOX required complex calculations, such as distance estimations and probability-based directional jumps, which added unnecessary computational complexity. IFOX addresses these limitations by introducing a dynamic, epsilon greedy approach to balance exploration and exploitation, where the balance shifts adaptively throughout training rather than remaining static. IFOX also simplifies the update mechanisms, removing calculations for sound-based distance and directionality, and instead directly using the best-known solution with random perturbations guided by dynamically adjusted parameters. These modifications enabled the IFOX to deliver enhanced and resilient convergence, a decreased computational cost, and enhanced generalization and training periods compared to the original FOX.

BAŞ [22] introduced notable research called BinFOX, a binary variant of FOX that utilizes U-shaped transfer functions to change ongoing search updates into binary answers. The work aims to address one of FOX's weaknesses, which is its failure to address binary optimization and discrete problems. Four variants of BinFOX are presented, each employing a distinct U-shaped transfer function to

stabilize exploration and exploitation inside a binary search space. This method illustrates that FOX may be effectively modified for binary problem domains by suitable discretization.

Sharma *et al.* [23] created another extension of the FOX optimizer named Binary FOX, which was proposed to overcome the inability of the original FOX algorithm to solve binary optimization problems, particularly feature selection tasks. Since the original FOX operates in a continuous search space, the authors transformed it into eight binary variants using S-shaped and V-shaped transfer functions to convert continuous position updates into binary values. This modification enables FOX to search effectively in binary solution spaces while preserving its exploration and exploitation capabilities. The proposed variants were evaluated on thyroid cancer image datasets for feature selection, where they reduced the number of selected features while maintaining high classification accuracy. Among the proposed variants, the V1 transfer function achieved the best overall performance, demonstrating the effectiveness of Binary FOX for binary optimization problems.

Aula and Rashid [24] proposed the hybridization of the Tree-Seed Algorithm (T-SA) and the FOX optimization algorithm, aimed at improving optimization methodologies through the collaborative fusion of these algorithms. Research outlines the creation of the Hybrid FOX-T-SA algorithm for traversing intricate search spaces and attaining enhanced effectiveness in both benchmark function tests and real optimization challenges. This method integrates FOX's exploratory powers with T-SA's exploitative strength, enabling the hybrid algorithm to adeptly manage intricate, multimodal, and limited optimization challenges by harmonizing global exploration with local refining, resulting in accelerated convergence and superior quality solutions. The results demonstrate that FOX-T-SA hybridization is proficient in improving performance across many applications by efficiently balancing exploration and exploitation mechanisms.

Aula and Rashid [25] also examined the FOX-T-SA hybrid algorithm, focusing on its utilization in optimizing Multi-Layer Perceptron (MLP) representations for analytical estimating in the tourism sector. This study highlights how the collaboration between T-SA's exploitation and FOX's exploration capacities substantially enhances the predicted accuracy and convergence speed for MLPs. The results show that FOX-T-SA hybridization serves as a potent and adaptable optimization tool, and can be used to solve different problems.

Jumaah *et al.* [26] proposed the hybridization of the FOX optimization method with Q-learning, or Q-FOX, which aims to enhance reinforcement learning efficacy by adaptively adjusting the hyperparameters, including the discount factor and learning rate. The Q-FOX approach employs a new objective function which optimizes the average Q-values while minimizing learning error, therefore improving the learning process and resulting in more cumulative rewards than conventional Q-learning. These adaptive tuning mechanisms mitigate the constraints of static hyperparameters, enabling the Q-learning agents to attain more stable and efficacious learning results.

Ahmed *et al.* [27] proposed a further hybridization of FOX by combining it with the differential evolution (DE) algorithm. In that work, the FOX optimization process is designated the red fox (RF). The authors developed the RF-DE technique to enhance convergence and address the RF algorithm's drawbacks, including premature and delayed convergence. This hybridization is accomplished by integrating the advantages of both RF and DE methodologies, yielding an improved convergence factor and a more resilient feature selection technique. The procedure entails an initial population setup phase, succeeded by consecutive RF and DE phases, wherein chaotic mapping is employed for initializing the population to increase the initial population's fitness and expand the search space. The RF phase mimics fox hunting strategies, but the DE phase utilizes mutation, crossover, and selection to enhance the solutions. This integration promotes information exchange among the group members and improves utilization.

Masbakhah *et al.* [28] proposed a separate study that combines FOX MH with the random forest (RF) model to improve cervical cancer prediction, resulting in a hybrid RF-FOX technique. The FOX algorithm is crucial in the feature selection step of the technique. Following the data preparation, the FOX algorithm is employed on the training data to determine and select the most pertinent and impactful variables for cervical cancer prediction. Upon acquiring an ideal feature set using FOX, these features are utilized to train the RF classifier, which gains from the more concentrated, information-dense

input. This hybridization enhances the classification results, as shown by the superior area under the receiver operating characteristic curve, recall, accuracy, and precision across the diagnostic categories in comparison to the use of RF alone.

Feda *et al.* [29] recently hybridized FOX with the grey wolf optimizer (GWO), creating the FOX-GWO method for feature selection tasks. This approach mitigates FOX's shortcomings, including insufficient exploitation and trapping in local optima, by incorporating GWO's leadership structure and information-sharing techniques. The hybrid employs FOX operators for exploration and GWO operators to improve exploitation, facilitating the algorithm's effective navigation of high-dimensional feature spaces. An S-shaped function for transfers is incorporated to transform continuous locations to binary, enhancing the selection of pertinent features. The procedure entails initializing the population, iteratively applying FOX and GWO methods to enhance solutions, and transforming the outcomes into a binary format for feature assessment. Figure 1 and table 1 provide an overview of fox enhancements.

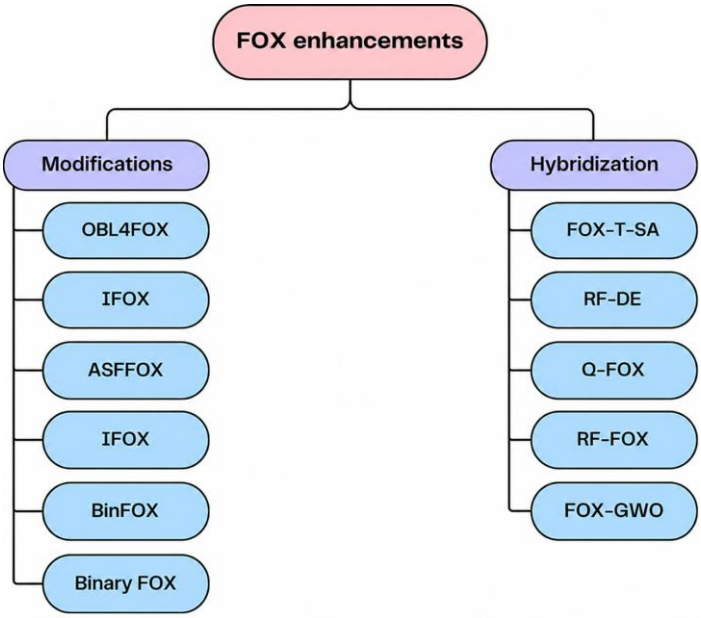

Figure 1: FOX modifications and hybridization with other algorithms.

Table 1: Enhanced FOX methods.

Reference	Method	Techniques used	Result
[18]	IFOX	An adaptable approach to exploitation and exploration. Also updates to the exploitation and exploration methods.	Demonstrated superior performance by outperforming the original FOX and other optimizers in 51 out of 52 benchmark test cases and two engineering problems.
[19]	OBL4FOX	Opposition-based learning (OBL) and four nonlinear dynamic control-variable strategies.	Outperformed FOX and other algorithms on the CEC2022 benchmark functions and achieved better objective values for three engineering optimization problems.
[20]	ASFFOX	Tent chaotic map for initialization, adaptive inertia weight, Lévy flight mechanism and variable spiral search strategy.	Surpasses the original FOX and six other algorithms in convergence speed, accuracy, and stability.
[21]	IFOX	Dynamic exploration exploitation trade-off, direct position updating, elimination of auxiliary variables and simplified jump and movement calculations.	Introduces Artificial Liver Classifier that demonstrated superior performance, outperforming or matching conventional classifiers across five benchmark datasets.
[22]	BinFOX	Four variations of BinFOX, each utilizing a different U-shaped transfer function.	Tested on 25 knapsack problems, the BinFOX1 variant proved to be the most effective, based on calculated gap values.

Table 1: continue.

[23]	Binary FOX	Introduce eight FOXs based on S and V transfer functions.	Provides an effective resolution for feature selection in machine learning.
[24]	FOX-T-SA	Merge FOX's exploratory capabilities with T-SA's exploitative power across CEC benchmark suites and real-world engineering problems.	Consistently outperforms PSO, GWO, FOX, and T-SA in convergence speed, solution quality, and computational efficiency.
[25]	FOX-T-SA	Merge FOX's exploratory capabilities with T-SA's exploitative power.	The hybrid method is ideal for analyzing vast and complicated datasets since it was very effective at optimizing MLP models
[26]	Q-FOX	Hybridize Q-learning with FOX.	The method improved Q-learning, achieving the maximum cumulative reward in Cart Pole and Frozen Lake tasks.
[27]	RF-DE	Using a chaotic map for initialization following by RF then DE steps.	The RF-DE feature selection algorithm demonstrated superior performance, outperforming six other contemporary optimization methods.
[28]	RF-FOX	Using FOX for selecting best features, then using it to train an RF classifier.	The hybrid model improved cervical cancer prediction, achieving over 95% accuracy by selecting key features.
[29]	FOX-GWO	The hybrid method uses FOX operators for exploration and GWO operators to boost exploitation.	The new algorithm showed superior performance across 18 datasets, outperforming other optimizers in most cases for accuracy.

It's worth mentioning that the FOX optimization algorithm, as an effective algorithm, has been used by researchers as a powerful metaheuristic instrument for dealing with intricate optimization, prediction, and model-tuning problems. In machine learning, FOX is used to optimize artificial neural networks by replacing traditional weight optimization methods to improve classification performance and avoid local optima traps [30]. FOX is able to solve nonlinear systems of equations and to tune support v regression models to accurately predict the thermal properties of advanced nanofluids [31, 32]. Also, FOX has been employed to address hyperparameter tuning within the context of reinforcement learning [33].

In medical imaging and diagnostics, FOX is applied for feature selection and weighted collective learning to enhance the precision of models in thyroid cancer detection and to optimize the deep learning models used for brain tumor as well as skin cancer diagnosis from magnetic resonance imaging (MRI) or dermatoscopic images [34-37]. FOX is used for solving control systems as well, as FOX optimizes controller gains such as in trajectory tracking and manipulator vibration reduction, and multi-resolution proportional integral derivative controllers for robust frequency regulation in hybrid power systems [38, 39].

In engineering design, FOX is used for cost minimization and parameter selection, demonstrated in optimal reinforced concrete column design and distributed generation placement in power networks, balancing technical and economic criteria [40, 41]. FOX also demonstrates strong performance in signal decomposition and time-series forecasting applications, where its enhanced variants have been used to optimize decomposition parameters under complex conditions, resulting in improved convergence accuracy and stability [20]. Additionally, FOX is used in tuning the multimodal deep learning frameworks used for the analysis of medical wounds, metaheuristic benchmarking, and image segmentation model optimization—in each, it adapts and tunes essential model hyperparameters or structural parameters to yield superior results compared to traditional methods [42, 43]. Figure 2's categories show the use of the FOX optimization algorithm in different fields.

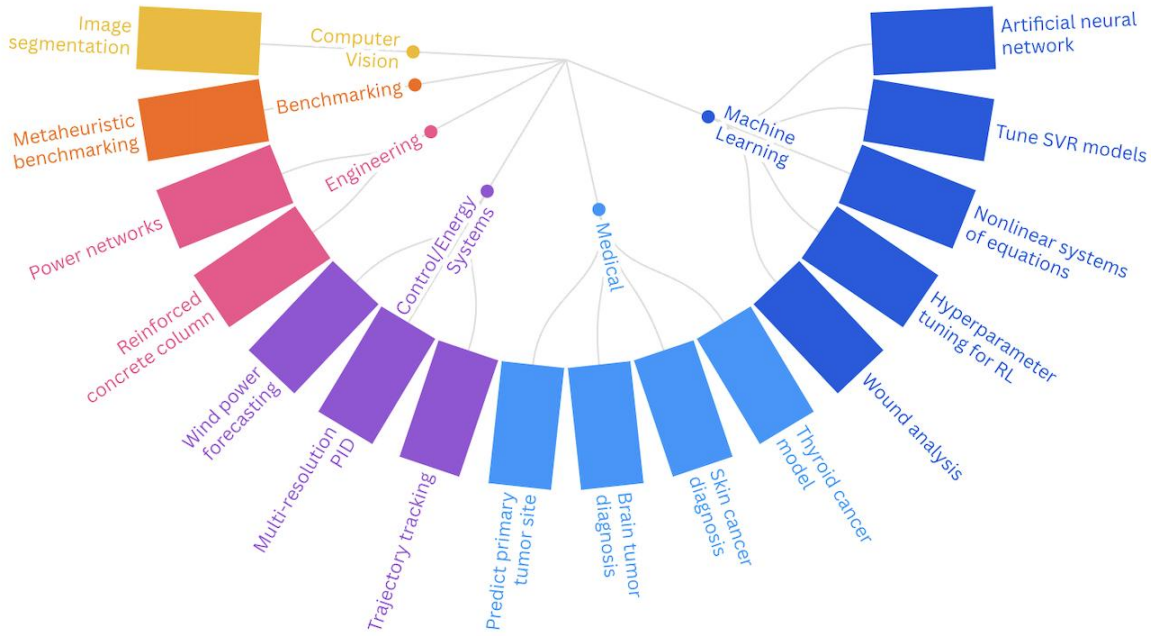

Figure 2: Representative applications of FOX in machine learning, computer vision, engineering, energy systems, medical diagnosis, and metaheuristic optimization.

Overall, the existing studies demonstrate that the performance of FOX can be enhanced by some modifications, especially in exploration, convergence and solution quality. However, most of the reported improvements focus on one or few modifications, and there is still room to investigate how different exploration strategies can cooperate to overcome the limitations of the original FOX. In particular, the problems of keeping the population diverse and avoiding premature convergence remain important challenges in complex optimization problems. These limitations show the need for a more complete improvement of FOX, improving its exploring capabilities while still being able to strike a good balance between exploring and exploiting.

Additionally, this study presents CLFOX, a new and enhanced version of FOX. The suggested version differs from every other FOX enhancement found in the literature. This version improves algorithm exploration by simultaneously introducing the Lévy flying technique and the Cubic chaotic map. Additionally, raising the main condition causes the algorithm to function very differently than it did in the original FOX. By introducing a random agent into the population via an interval-based technique, the algorithm is able to increase variety and decrease the chance of getting trapped in local targets.

3. Material and Methods

3.1.1. Fox Optimization Algorithm

The FOX algorithm is an entirely novel MH motivated by nature that was developed in 2022 [1]. It is driven by the distinctive chasing method of the red fox in snowy environments. FOX has been benchmarked against a range of established MH algorithms, demonstrating competitive performance in solving both benchmark numerical functions and real-world optimization scenarios.

The hunting activity of the red fox in snow includes three essential stages: searching, detecting ultrasonic signals, and jumping to grab prey. Initially, the fox engages on a random route to locate prey, because snow cover significantly limits visibility. In its next phase, it identifies prey by detecting ultrasonic signals beneath the snow's surface. The fox evaluates the sound delay to determine the prey's distance and performs an accurate jump to grab it. Nearly all MH optimization algorithms are comprised of two phases: exploration and exploitation. The FOX algorithm integrates each of these behavioral phases to simulate its exploration and exploitation mechanisms. Significantly, in contrast to nu-

merous other algorithms, FOX clearly separates these two phases, allocating specific processes for exploitation and exploration. This division improves the algorithm's stability and overall efficacy. The subsequent sections clarify the mathematical modeling of these natural processes and their integration into the algorithmic design.

3.1.2. FOX Initialization

Like many other nature-inspired algorithms, the FOX algorithm starts by initializing a population of search agents, often referred to as the initial matrix or initial population. As shown in Equation (1), the variable X represents a matrix containing n agents (rows) and m dimensions (columns), where each element $x_{i,j}$ denotes the $j-th$ dimension of the $i-th$ agent:

$$X = \begin{matrix} x_{1.1} & x_{1.2} & \dots \\ x_{2.1} & x_{2.2} & \dots \\ \vdots & \vdots & \dots \end{matrix} \quad (1)$$

The algorithm evaluates all agents at the beginning of every iteration by applying the objective (fitness) function. Recognized and kept as $BestPosition$ is the agent displaying the lowest or optimal fitness value; its corresponding fitness is recorded as $BestFitness$.

To ascertain whether the algorithm should engage in exploration or exploitation throughout each iteration, a random variable r_1 between $[0, 1]$ is produced. If $r_1 \geq 0.5$, is more than or equal to 0.5, the exploitation stage is initiated. Otherwise, exploration is employed. This probabilistic method guarantees that the algorithm allocates roughly 50% of its iterations to each phase, preserving the equilibrium between global and local search.

3.1.3. Exploitation Phase

The FOX algorithm's exploitation phase emulates two essential elements in the hunting behavior of the red fox: identifying prey using ultrasonic signals and performing a strategic jump to catch it. During this phase, the algorithm evaluates the prey's location and directs each agent to move strategically toward that place. To begin with, the algorithm calculates the speed of sound for the current iteration using the best-known result and a randomly generated time value. Let $SpeedOfSound_i$ be a random number in the extent $[0, 1]$ and $BestPosition_i$ reflect the best place discovered thus far. The speed of sound is then computed as:

$$SpeedOfSound_i = \frac{BestPosition_i}{TimeSoundTravel_i} \quad (2)$$

Next, the total distance the sound travels from the fox to the prey and back is calculated using:

$$DistanceSoundTravel_i = SpeedOfSound_i * TimeSoundTravel_i \quad (3)$$

It is crucial to recognize that both equations use the same value for $TimeSoundTravel_i$. Since the total travel distance includes both the forward and return paths, the actual distance to the prey is taken as half:

$$PreyDistance_i = \frac{DistanceSoundTravel_i}{2} \quad (4)$$

This logic follows a basic principle from physics: when using sound or ultrasonic waves to detect an object, the measured distance must be divided by two to reflect the one-way travel.

After estimating the prey's distance, the algorithm calculates the jump height of the fox using a basic kinematic equation:

$$Jump_i = \frac{9.81 * t^2}{2} \tag{5}$$

Here, 9.81 shows the acceleration impacted on by gravity, and t is the mean time that the sound takes to travel. To determine this average time, the algorithm first computes the mean across all dimensions:

$$SumTimePerDim = \frac{\sum TimeSoundTravel_i\ (i:1)}{Dimension} \tag{6}$$

$$t = \frac{SumTimePerDim}{2} \tag{7}$$

Because the jump involves both upward and downward motion, the total time is divided by 2 to capture the duration of a single direction.

Finally, a new position is generated using the calculated jump and prey distance. The algorithm uses one of two coefficients, c_1 or c_2, based on the value of another random variable r_2. If $r_2 \leq 0.18$, the fox is modeled as jumping in a less favorable direction, and c_1 in the rang $[0, 0.18]$ is used.

Otherwise, if $r_2 > 0.18$, the fox leaps towards the northeast, which research shows has an 82% success rate. In this case, a larger random value c_2 in the interval $[0.19, 1]$ is used:

$$X_{(i+1)} = \begin{cases} PreyDistance_i * Jump_i * C1, & r_2 > 0.18 \\ PreyDistance_i * Jump_i * C2, & r_2 \leq 0.18 \end{cases} \tag{8}$$

This directional preference is inspired by real red fox behavior, which tends to favor the northeast when diving for prey due to magnetic alignment. By applying this rule, the FOX algorithm increases its chances of finding the global optimum during exploitation.

3.1.4. Exploration Phase

The initial stage of the red fox's hunting strategy, which involves seeking out prey beneath the snow, is represented in the algorithm as the exploration phase. In metaheuristic optimization, exploration denotes the worldwide and random investigation of the search space to find appealing regions that may provide optimal solutions. In the FOX algorithm, each agent (fox) searches randomly across the solution space in an effort to locate the prey. This randomized search permits the algorithm to diversify its exploration and avoid early local optimum convergence. The strategy followed in this phase is given by the following equation:

$$X_{(i+1)} = BestPosition_i + rand(1, Dim) * MinSumTimePerDim * a \tag{9}$$

Where $BestPosition_i$ is the best place, i is the current iteration and $rand(1, Dim)$ produces a random matrix to give the phase more randomness and ensure the fox moves stochastically.

The value of $MinSumTimePerDim$ is decided by selecting the minimum value among all previously calculated average times, as shown below:

$$MinSumTimePerDim = Min(SumTimePerDim) \tag{10}$$

The variable a is introduced to shrink the exploration range as the algorithm progresses. This helps to focus the search more precisely in later iterations. Its value starts at 2 and decreases linearly with each iteration, calculated as:

$$a = 2 * \left(i - \left(\frac{1}{Max_i} \right) \right) \tag{11}$$

Here, Max_i represents the maximum iterations. By using this strategy, FOX ensures a balanced transition from broad exploration in initial iterations to a more sophisticated, localized search in later stages.

Figure 3 and Algorithm 1 provide the FOX optimization algorithm's flow chart and pseudo-code.

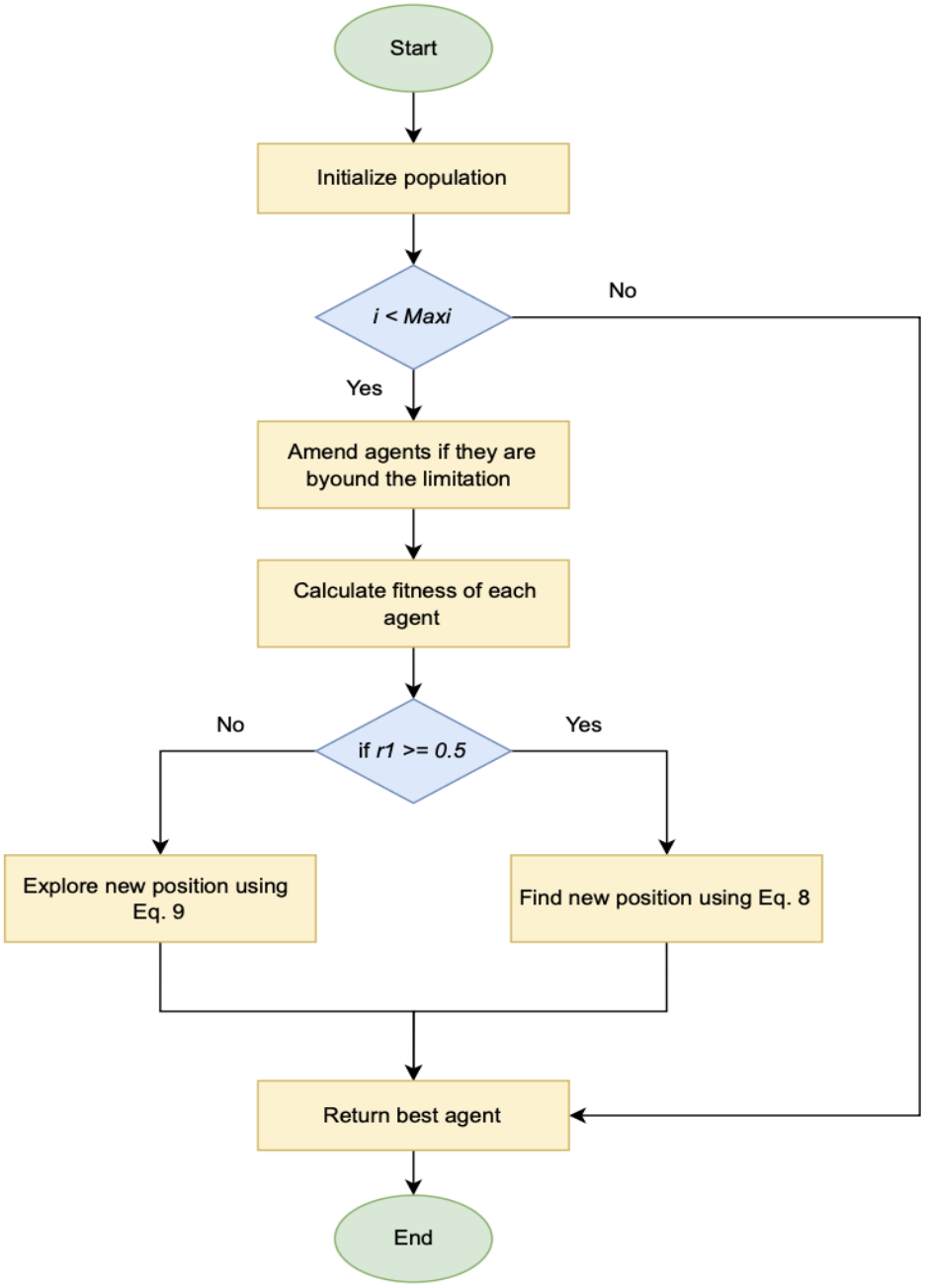

Figure 3: FOX optimization algorithm flowchart, illustrating population initialization, fitness evaluation, position updates, and output of the best agent..

Algorithm 1: FOX optimization algorithm.

Initialize the red fox population X, $BestPosition$, $BestFitness$, $MinSumTimePerDim$, c_1, c_2
Input: red fox population X
While $i < Max_i$
  for each agent j **in** X **do**
    return the search agents who cross the boundaries
    calculate $Fitness$ for each agent
    if $Fitness_j < BestFitness$
      $BestFitness = Fitness_j$
      $BestPosition = X(i:1)$
    endIf
  endFor
  for each agent j **in** X **do**
    if $r_1 \geq 0.5$
      initialize $TimeSoundTravel$
      calculate $SpeedOfSound$ using equation (2)
      compute $DistanceSoundTravel$ using equation (3)
      compute $PreyDistance$ using equation (4)
      compute $SumTimePerDim$ and t using equations (6 and 7)
      calculate $Jump$ using equation (5)
      if $r_2 > 0.18$
        find new position using equation (8), first condition
      elseif $r_2 \leq 0.18$
        find new position using equation (8), second condition
      endIf
    elseif $r_1 < 0.5$
      calculate $MinSumTimePerDim$ using equation (10)
      calculate a using equation (11)
      explore new position using equation (9)
    endIf
  endFor
  $i = i + 1$
endWhile
Output: $BestPosition$ and $BestFitness$

3.2. *Proposed CLFOX*

As previously mentioned, metaheuristic algorithms often depend on two primary strategies: exploration and exploitation. Attaining an efficient mix of these strategies is crucial for the algorithm's overall efficacy. Exploitation means focusing the search on the most promising answers identified so far, also known as the local search. An algorithm may insufficiently converge and show irregular search behavior without adequate exploitation. Conversely, exploration permits the algorithm to analyze the wide search space in search of various solutions. A lack of exploration could lead to early convergence and trapping in local optima because of the reduced population variation.

The original FOX algorithm's exploitation mechanism operates effectively; nevertheless, its exploration capacity is significantly constrained [1, 18]. This difference limits the algorithm's performance when it comes to steering clear of local search results and identifying global solutions. Empirical evidence corroborates this claim, since FOX exhibits superior performance in the unimodal functions relative to multimodal ones, further highlighting its weak exploratory behavior.

3.2.1. *Chaotic Initialization of Population*

The standard FOX approach initializes the population with uniformly distributed random numbers. This strategy, although relatively simple, frequently leads to restricted variety in the population and heightens the possibility of early convergence to the local optima. In an algorithm like FOX, which demonstrates a restricted exploratory capability, the initialization of a stochastic and heterogeneous population is crucial. An effectively distributed and stochastic population initialization greatly enhances the search space coverage and optimization efficacy.

The CLFOX algorithm incorporates chaotic maps into the initialization process to overcome this issue. Chaotic maps are nonlinear systems distinguished by their dynamic nature. Chaos is a restricted,

unstable dynamic process marked by a sensitive dependence on beginning conditions and involves an unlimited number of unstable episodic motions [44, 45]. Chaos is described by three essential belongings: ergodicity, randomness, and sympathy to beginning conditions. The incorporation of these elements ensures the variety of the generated solutions [46-48]. Consequently, they are adept at enhancing population initialization in metaheuristic algorithms.

The use of chaotic systems in population-based algorithms has been extensively studied, with notable algorithms such as the fitness dependent optimizer [45], Aquila Optimization (AO) [46], grey wolf optimizer (GWO) [49], ant colony optimization (ACO) [50], particle swarm optimization (PSO) [51], salp swarm algorithm [52], firefly algorithm [53], coati optimization algorithm [54], artificial hummingbird algorithm [55] and kepler optimization algorithm [56]. In these algorithms, chaotic maps are typically used to generate random numbers or initialize populations, leading to improved exploration and convergence characteristics.

In this study, thirteen chaotic maps were examined for their effectiveness in enhancing the FOX algorithm's performance. These include the Chebyshev, Gauss, Circle, Logistic, Iterative, Piecewise, Sine, Sinusoidal, Singer, Tent, Bernoulli, Piecewise, and Cubic maps. Each map was evaluated using three different sets of benchmark functions, specifically classical benchmark methods, the CEC2019 benchmark suite, and the CEC2022 benchmark suite. Table 2 shows the win value, win rate, and significance rate of each map regarding each of the three sets of functions. The win value shows how many functions the suggested algorithm outperforms the FOX optimization algorithm for per map, with a total of 45 functions. The win rate is the percentage representation of the win value. The significance rate refers to the p-value obtained from each test conducted to assess its significance.

Table 2: Chaotic map results for the proposed algorithm in comparison with the FOX optimization algorithm.

Map		Classical	CEC2019	CEC2022	Total
Chebyshev	Win value	22	7	9	38
	Win rate	95.65	70	75	80.22
	Significance rate	86.67	85.71	66.67	79.68
Circle	Win value	20	7	10	37
	Win rate	86.96	70	83.33	80.1
	Significance rate	84.62	85.71	50	73.44
Gauss	Win value	21	8	9	38
	Win rate	91.3	80	75	82.1
	Significance rate	85.71	75	66.67	75.79
Iterative	Win value	21	8	8	37
	Win rate	91.3	80	66.67	79.32
	Significance rate	92.86	75	75	80.95
Logistic	Win value	22	7	9	38
	Win rate	95.65	70	75	80.22
	Significance rate	80	71.43	77.78	76.4
Piecewise	Win value	21	8	7	36
	Win rate	91.3	80	58.33	76.54
	Significance rate	85.71	87.5	85.71	86.31
Sine	Win value	21	6	9	36
	Win rate	91.3	60	75	75.43
	Significance rate	92.86	100	66.67	86.51
Singer	Win value	22	8	10	**40**
	Win rate	95.65	80	83.33	**86.33**
	Significance rate	80	87.5	70	79.17
Sinusoidal	Win value	22	7	10	39
	Win rate	95.65	70	83.33	82.99
	Significance rate	86.67	85.71	60	77.46
Tent	Win value	21	9	9	39
	Win rate	91.3	90	75	85.43
	Significance rate	92.86	66.67	66.67	75.4
Bernoulli	Win value	21	6	10	37
	Win rate	91.3	60	83.33	78.21
	Significance rate	85.71	83.33	70	79.68

Table 2: continue.

Plcm	Win value	22	8	9	39
	Win rate	95.65	80	75	83.55
	Significance rate	100	87.5	66.67	84.72
Cubic	Win value	22	8	10	**40**
	Win rate	95.65	80	83.33	**86.33**
	Significance rate	93.33	87.5	70	**83.61**
No map	Win value	21	8	9	38
	Win rate	91.3	80	75	82.1
	Significance rate	92.86	75	66.67	78.18

The Cubic map is defined mathematically as:

$$x_{n+1} = ax_n^3 + bx_n^2 + (1 - a - b)x_n \tag{12}$$

Where x_n is a random value in rang $[0, 1]$, x_{n+1} is the next generated value, and a and b are control parameters commonly set to $a = -1.5, b = 1.5$.

3.2.2. Lévy Flight-Based Exploration Strategy

Lévy flights are named for Paul Lévy, a French mathematician, and are a category of non-Gaussian random processes characterized by stationary increments that follow a Lévy stable distribution [57]. These probability distributions are referred to as Lévy distributions or steady distributions. The Lévy flight distribution exhibits a significant likelihood of producing huge steps, which occur with greater frequency than in a normal distribution [58, 59]. In another words, the steps that are generated using Lévy flight gives more randomness and keep the diversity more than using a random or decreased step size.

Two of the core challenges identified in the original FOX algorithm is its weak exploratory behavior and the imbalance between exploration and exploitation. This imbalance often causes the algorithm to converge too early, getting stuck in local optima due to its stronger focus on exploitation. To address these limitations, the CLFOX algorithm encompasses the Lévy flight mechanism to improve its exploration capabilities.

Lévy flights have proven successful at improving the performance of various metaheuristic algorithms, including GWO [60], ACO [61], WOA [62], CGO [63] and AOA [64]. By replacing the random movement component in the FOX update equation with a Lévy flight-based step, the algorithm gains the capacity to perform both global and local searches more efficiently. The following is the modified version of the exploration phase (Equation 9):

$$X_{(i+1)} = BestPosition_i + L * MinSumTimePerDim * a \tag{13}$$

Here, L is the step generated with Lévy flight, calculated using the following equations:

$$L = \frac{u}{|v|^{1/\beta}} \tag{14}$$

$$u \sim \mathcal{N}(0,\ \sigma^2 I_{Dim}),\quad v \sim \mathcal{N}(0,\ I_{Dim}) \tag{15}$$

$$\sigma = \left(\frac{\Gamma(1+\beta)\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right)\beta 2^{\frac{\beta-1}{2}}} \right)^{1/\beta} \tag{16}$$

In these equations, the gamma function is $\Gamma(\cdot)$, u, v are Gaussian-distributed random vectors of dimension Dim and σ is a scaling for u to produce the correct Lévy behavior. Finally, β is a constant

known as the Lévy index, which governs the distribution's tail behavior. A lower value of β indicates that the distribution's tail exhibits significant data accumulation, meaning that the values of the random variable are more likely to deviate substantially from the distribution's mean. A higher value of β signifies that the values of the random variable are closely clustered around the mean of the distribution [57].

3.2.3. Main Condition Increment

In the primary FOX algorithm, the split between exploitation and exploration is dictated by a stochastic condition, assigning an equal 50% probability to each phase. Despite this superficial balance, the exploration phase increasingly resembles exploitation in later iterations due to relying on the optimal agent within the population [19]. This design mistakenly reduces the algorithm's exploratory capability and heightens the likelihood of early convergence.

An additional condition has been included in the enhanced version of the algorithm to help alleviate this limitation. The original exploration and exploitation components are each allocated a 30% probability, reducing the overall effect to 60% of the whole decision-making process. The remaining 40% is allocated to a novel condition predicated on the Lévy flight mechanism, specifically engineered to improve exploration by facilitating broader and more varied search steps.

This modification enhances the overall probability of exploratory behavior via either the initial exploration equation (Equation 9) or the Lévy flight-based update rule (Equation 13). Despite the increased focus on exploration, the algorithm preserves a balanced dynamic between exploitation and exploration. This is due to the continued use of the best agent in both exploration strategies, which ensures convergence while still expanding the search capability of the algorithm.

3.2.4. Interval-Based Operator

An interval-based technique in optimization algorithms involves doing a certain action at regular intervals, usually following a certain number of iterations. This method is very effective for preserving population diversity, reducing stagnation, and improving overall algorithmic efficacy. The CLFOX algorithm includes an interval-based operator to prevent premature convergence by infusing variation into the population every 10 iterations.

The technique involves the random selection of an agent from the population and uniformly reinitializing its position within the specified search space. This periodic reset introduces novel genetic material into the population, promoting the discovery of uncharted areas within the solution space. The following formula determines the selected agent's updated position:

$$X_r = L_b + (U_b - L_b)\ .\ *\ rand(1, Dim) \tag{17}$$

Where r indicates the index of the randomly chosen agent, X_r is its updated position, L_b and U_b are the search space's lower and upper boundaries. This method guarantees the periodic introduction of novel diversity into the population, facilitating the exploration of new areas and diminishing the likelihood of stagnation.

Ultimately, CLFOX employs four key techniques (chaotic maps, interval-based updates, Lévy flight, and sigmoid-based movement) to improve the algorithm's overall performance. These techniques collectively serve to maintain population diversity, increase the algorithm's exploratory capabilities, and ensure a fair compromise between exploitation and exploration. By periodically introducing variation and non-linear search dynamics, CLFOX significantly reduces the likelihood of early convergence and entrapment in local optima. Figure 4 and algorithm 2 present the CLFOX flow chart and pseudo-code.

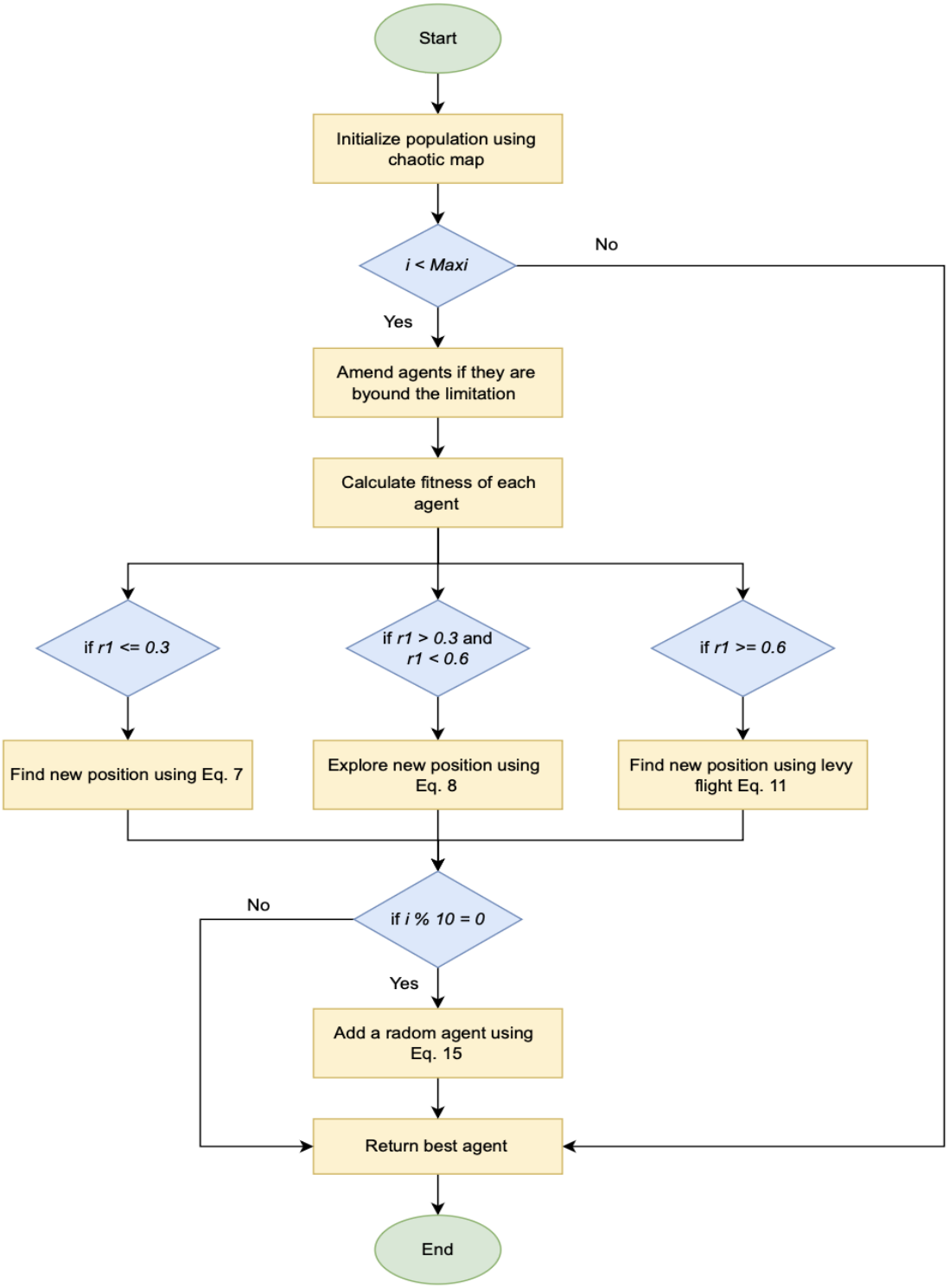

Figure 4: CLFOX optimization algorithm flowchart, showing population initialization, fitness evaluation, position updates, periodic random agent injection and output of the best agent.

Algorithm 2. CLFOX optimization algorithm.

Initialize the red fox population X using Cubic chaotic map
Initialize $BestPosition$, $BestFitness$, $MinSumTimePerDim$, c_1, c_2
Input: the red fox population X
While $\boldsymbol{i < Max_i}$
  for each agent $\boldsymbol{j}$ **in** $\boldsymbol{X}$ **do**
    return the search agents who cross the boundaries
    calculate $Fitness$ for each agent
    if $\boldsymbol{Fitness_j < BestFitness}$
      $BestFitness = Fitness_j$
      $BestPosition = X(i: 1)$
    endIf
  endFor
  for each agent $\boldsymbol{j}$ **in** $\boldsymbol{X}$ **do**
    if $\boldsymbol{r_1 \leq 0.3}$
      initialize $TimeSoundTravel$
      calculate $SpeedOfSound$ using equation (2)
      compute $DistanceSoundTravel$ using equation (3)
      compute $PreyDistance$ using equation (4)
      compute $SumTimePerDim$ and t using equations (6 and 7)
      calculate $Jump$ using equation (5)
      if $\boldsymbol{r_2 > 0.18}$
        find new position using equation (8), first condition
      elseif $\boldsymbol{r_2 \leq 0.18}$
        find new position using equation (8), second condition
      endIf
    elseif $\boldsymbol{r_1 > 0.3}$ && $\boldsymbol{r_1 < 0.6}$
      calculate $MinSumTimePerDim$ using equation (10)
      calculate a using equation (11)
      explore new position using equation (9)
    elseif $\boldsymbol{r_1 \geq 0.6}$
      calculate L using equation (14)
      calculate σ using equation (16)
      explore new position using equation (13)
    endIf
    if $\boldsymbol{i \ mod \ 10 = 0}$
      replace a random agent in X using equation (17)
    endIf
  endFor
  $i = i + 1$
endWhile
Output: $BestPosition$ and $BestFitness$

4. Results

CLFOX was tested on three sets of benchmark functions along with five real world applications. The sets included classical benchmark methods, CEC2019 benchmark suite and CEC2022 benchmark suite, which overall make up 45 benchmark functions. The proposed algorithm was compared to nine recent and powerful algorithms such as the original FOX optimization algorithm, AO, Dingo Optimization Algorithm (DOA), Human Evolutionary Optimization Algorithm (HEOA), Rat Swarm Optimizer (RSO), Tunicate Swarm Algorithm (TSA), Hiking Optimization Algorithm (HOA), Greylag Goose Optimization (GGO) and Pufferfish Optimization Algorithm (POA). Each algorithm was tested 30 times with a maximum of 500 iterations and 30 search agents. The results of all algorithms are available in the benchmark functions' result tables. With every trial, the p-value was calculated to demonstrate the significance of CLFOX's better results in contrast to the other algorithms. The Wilcoxon rank-sum test was employed to get the p-value through the paper. All p-values presented in this study were derived using this trial with a weight threshold of 0.05. A list of all hyperparameters of CLFOX and all algorithms compared is shown in table 3. The findings prove that CLFOX stands out from almost all of them in fixing the benchmark functions. Compared to FOX, CLFOX gives a better result in 42 out of the 45 tested functions. Table 4 shows each tested algorithm's time complexity and function evaluation. Where n is the value number of agents in each iteration, m is the dimension and $Maxi$ represents the

maximum number of iterations the algorithm runs through. It's notable that all algorithms except AO have the same linear time complexity, where increasing iterations linearly increase the time complexity.

The mean actual runtime of each algorithm throughout the 30 times it ran has been established in the results tables; it's notable that there is a slight difference between the FOX and CLFOX runtimes. CLFOX took a bit more time to find the best solution in almost every function compared to FOX, and this delay is due to Lévy flight generation, chaotic initialization, and running an interval every 10 iterations. All simulations were performed using MATLAB R2025b on a MacBook Pro with 8 GB of unified memory and an 8-core Apple M2 CPU.

Table 3: Hyperparameters of CLFOX and all compared algorithms.

Algorithm	Parameter	Value
# (For all algorithms)	**N - Population size**	**30**
CLFOX	Iteration	500
	c1, c2	0.18, 0.81
	B	1.9
	p, r	[0, 1]
FOX	Iteration	500
	c1, c2	0.18, 0.81
	p, r	[0, 1]
AO	Iteration	250
	α, δ	0.1
DOA	Iteration	500
	P, Q	0.5, 0.7
	β1, β2	[-2, 2], [-1, 1]
	Survival threshold	0.3
HEOA	Iteration	500
	A	0.6
	LN, EN	0.4
	FN	0.1
	β	1.5
RSO	Iteration	500
	x, y	1, 5
	R	[1, 5]
	C	[0, 2]
TSA	Iteration	500
	xmin, xmax	1, 4
	xr	[1, 4]
	c2, c3	[0, 1]
HOA	Iteration	500
	θ	[0, 50]
	SF	{1, 2}
GGO	Iteration	500
POA	Iteration	500

Table 4: Information and time complexity of the compared algorithms.

Algorithm	Year	Time Complexity	Objective Function per Iteration	Max Iterations	Agent × Iterations
Main FOX	2022	$O(nm(1+Maxi))$	1	500	15000
CLFOX		$O(nm(1+Maxi))$	1	500	15000
AO [65]	2021	$O(nm(1+2Maxi))$	2	250	15000
DOA [66]	2021	$O(nm(1+Maxi))$	1	500	15000
HEOA [67]	2024	$O(nm(1+Maxi))$	1	500	15000
RSO [68]	2021	$O(nm(1+Maxi))$	1	500	15000
TSA [69]	2020	$O(nm(1+Maxi))$	1	500	15000
HOA [70]	2024	$O(nm(1+Maxi))$	1	500	15000
GGO [71]	2024	$O(nm(1+Maxi))$	1	500	15000
POA [72]	2024	$O(nm(1+Maxi))$	1	500	15000

4.1. Selection of the Chaotic Map

As shown in table 2, thirteen chaotic maps have been used in the proposed algorithm's initialization and experimented on using the 45 benchmark functions. Among all of the maps, the best one was chosen, in other words, the map that overcame FOX the most. The results of these experiments are illustrated in figure 5, which highlights the three chaotic maps that produced the most notable enhancements over the initial FOX algorithm across the majority of test cases.

Among these, with a win rate of 86.33%, the Cubic map demonstrated the most consistent and significant enhancement. Regarding the successful tests of the Cubic map, 83.61% were significant. As shown in the figure 5, the Singer map also has an 86.33% win rate but 79.17% of the successful test cases of the map were significant, which is why the Cubic map was chosen over the Singer and all other maps. It was therefore selected as the primary method for initializing the population in the improved CLFOX algorithm.

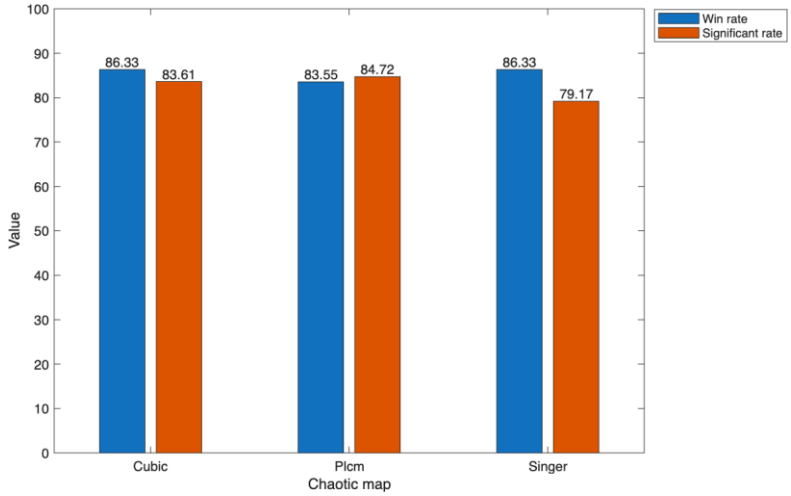

Figure 5: Three best chaotic map rate results for the proposed algorithm in comparison with the FOX optimization algorithm.

4.2. Selection and Evaluation of the Lévy Flight Parameter

To determine the most effective value for β in the context of CLFOX, the parameter was tested across a range from 0 to 2. The evaluation was conducted using 45 benchmark functions, comparing the original FOX algorithm's performance to the improved version. The results, summarized in figure 6, indicate that setting $\beta = 1.9$ yields the most consistent improvements. Specifically, this configuration outperformed the original FOX in 42 out of the 45 test functions, and had the highest value of significance, demonstrating its robustness and effectiveness in enhancing exploratory behavior.

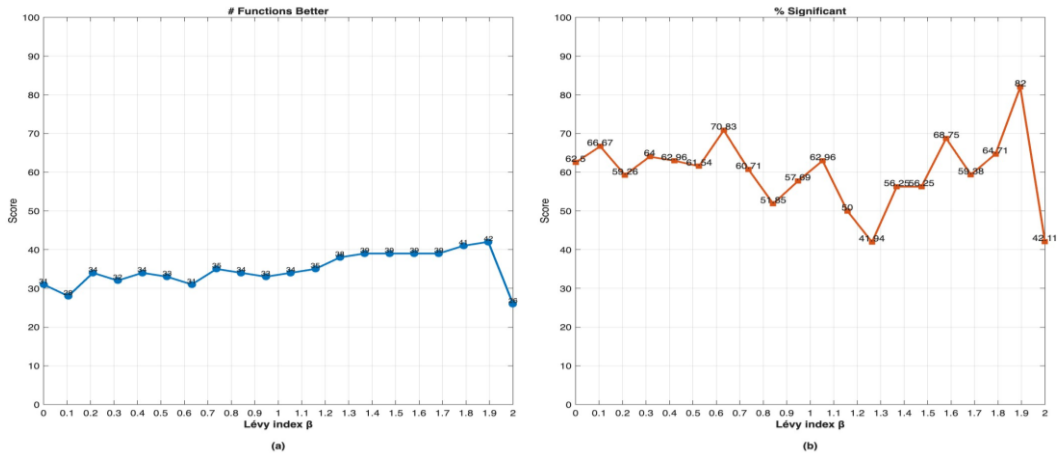

Figure 6: The Lévy index shows better results among the 45 functions compared to FOX. (a) the number of better results functions and (b) the siginificance rate of each test.

Figure 7 illustrates that the Lévy flight mechanism facilitates a more extensive exploration of the search space than the conventional random walk when applied to 23 classical benchmark functions. This improved exploratory behavior aids in preserving population diversity and markedly lowers the algorithm's probability of becoming caught in local optima.

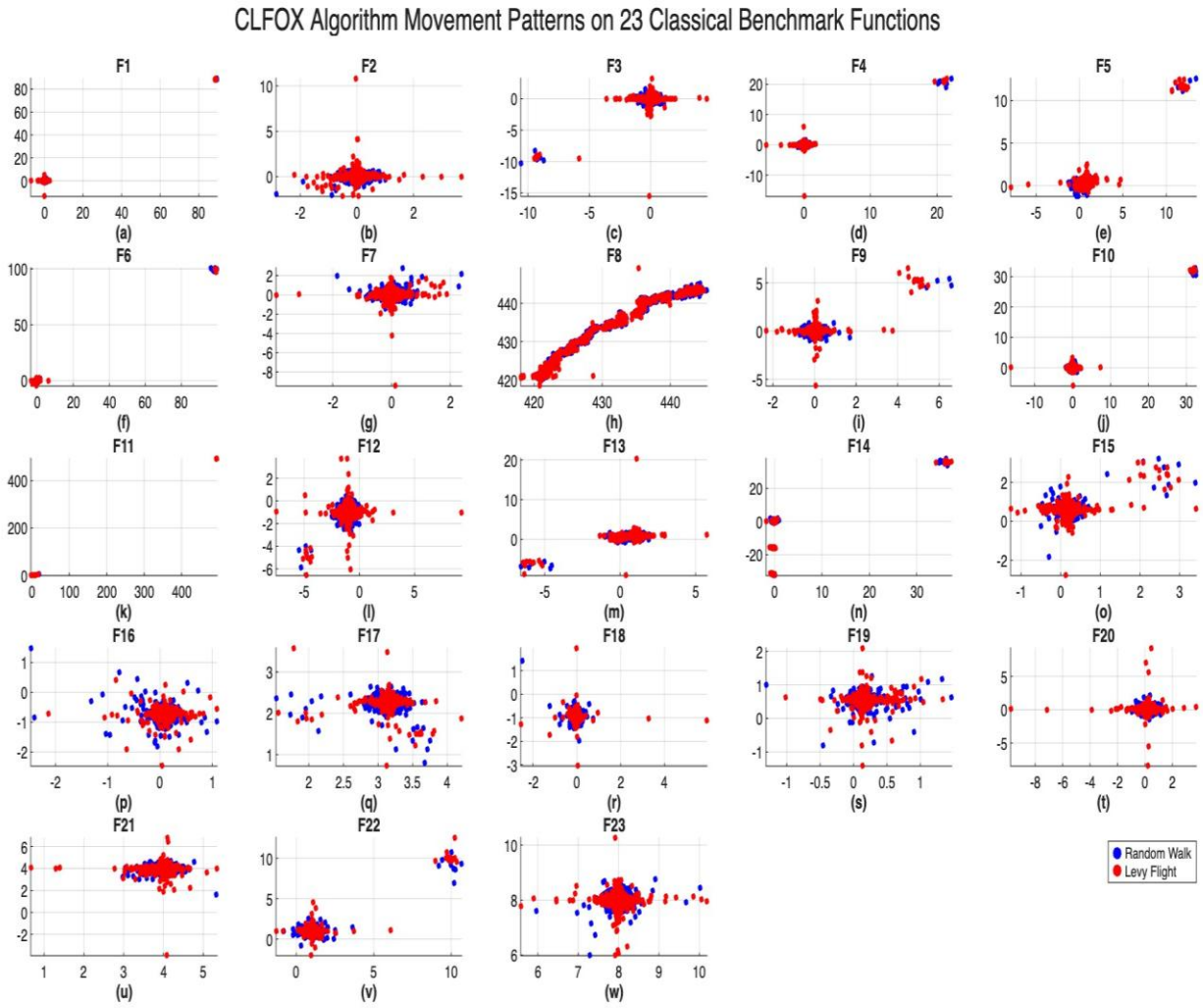

Figure 7: Lévy flight vs random walk search agents, (a-w) tested on 23 classical benchmark functions.

4.3. Interval-Based Operator Evaluation

To show the impact of adding the interval-based strategy to the proposed algorithm, CLFOX was evaluated using three sets of benchmark functions, both with and without the interval strategy. Figure 8 illustrates the impact of the interval on CLFOX in comparison to FOX. CLFOX with the interval strategy achieved better results than FOX in 42 out of 45 functions. When the interval strategy was removed from CLFOX, the number of functions on which it outperformed FOX decreased to 33 across the three benchmark sets.

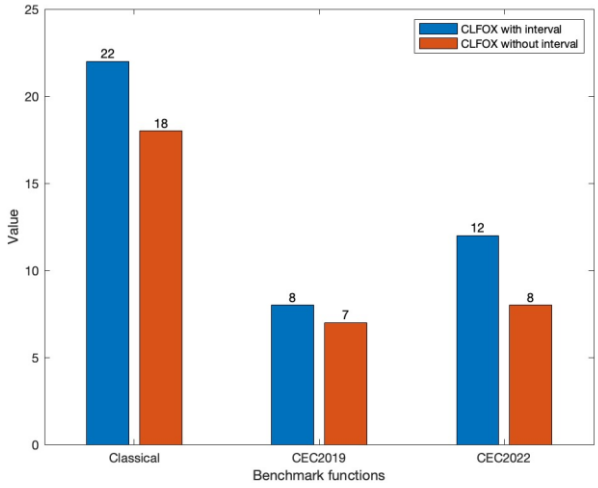

Figure 8: CLFOX with and without interval results compared to FOX.

4.4. CLFOX Evaluation using Classical Benchmark Functions

To assess the effectiveness of the suggested technique, one of the function sets used is classical benchmark functions. The set contains 23 functions. Using classical benchmark functions is a challenging and important way to evaluate an algorithm because the set contains functions with different properties. Functions 1 to 7 are unimodal methods, functions eight to thirteen have multimodal properties, and functions 14 to 23 are fixed-dimensional functions. The variety in the set lets researchers use classical benchmark functions to evaluate all possible weaknesses of their algorithms. In other words, if an algorithm has issues with a function from 1 to 7, it means that the algorithm has weak exploitation, since those functions are unimodal and only have a single global optimum. Also, when an algorithm scores low in functions from 8 to 13, which are multimodal functions, it means that the algorithm lacks when it comes to exploring the search area, so the exploration phase must be improved. Functions from 14 to 23, which are fixed-dimensional functions, can be used to evaluate both the exploitation and exploration of an algorithm.

Table 5 displays the results as average (Avg), standard deviation (Std), median, best, worst and runtime for each function over the nine algorithms and CLFOX. The results demonstrate that CLFOX delivers a superior performance in 22 functions compared to the FOX optimization algorithm, and overall outperforms 15 functions compared to all other algorithms. The functions where CLFOX outperforms all other algorithms have a bolded value inside the table. CLFOX outperforms FOX in all unimodal functions except function 7, and CLFOX does not take the first rank in function 5 compared to all other algorithms. It is notable that the two functions (7 and 5) have the same properties, as they are both unimodal, non-separable, non-rotated and asymmetric functions. Otherwise, CLFOX outperforms FOX in all multi-modal and fixed-dimension functions. This proves that CLFOX is more capable of exploring than FOX, and that CLFOX maintains the harmony between exploitation and exploration.

Based on the results of table 5, figure 9 was created. The figure 9 shows the average rank and place of the algorithms based on their results regarding the 23 classical benchmark functions.

Table 5: Avarage, standard deviation, median, best, worst and runtime values for CLFOX and the compared algorithms, evaluated using classical benchmark functions.

Algorithm		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23
CLFOX	Avg	0	0	0	0	0.21307	3.78E-07	0.000148	-1739.538	0	4.44E-16	0	1.05E-07	2.05E-07	4.6769	0.00032097	-1.0316	0.39789	3	-3.8628	-3.2743	-6.2523	-6.159	-5.8621
	Std	0	0	0	0	0.075462	2.53E-07	0.0001316	288.5844	0	0	0	8.78E-08	2.83E-07	3.7654	2.92E-05	9.98E-10	1.38E-10	1.61E-08	1.95E-07	0.059355	2.1889	2.1583	1.8649
	Median	0	0	0	0	0.20342	3.26E-07	0.00011383	-1719.852	0	4.44E-16	0	9.47E-08	9.72E-08	3.9683	0.00031046	-1.0316	0.39789	3	-3.8628	-3.322	-5.0552	-5.0877	-5.1285
	Best	0	0	0	0	0.011362	5.27E-08	3.77E-06	-2094.914	0	4.44E-16	0	2.07E-08	7.39E-09	0.998	0.00030749	-1.0316	0.39789	3	-3.8628	-3.322	-10.1532	-10.4029	-10.5364
	Worst	0	0	0	0	0.34614	9.02E-07	0.00052317	-1314.953	0	4.44E-16	0	3.88E-07	1.42E-06	12.6705	0.00042538	-1.0316	0.39789	3	-3.8628	-3.2026	-5.0552	-5.0877	-5.1285
	Runtime	2.64E-02	2.40E-02	3.04E-02	2.30E-02	2.95E-02	2.32E-02	2.93E-02	2.63E-02	2.48E-02	2.51E-02	2.79E-02	5.12E-02	5.17E-02	1.57E-01	2.44E-02	2.32E-02	2.09E-02	2.06E-02	2.66E-02	2.87E-02	2.91E-02	3.14E-02	3.55E-02
FOX	Avg	0	0	0	0	0.26439	1.07E-06	9.59E-05	-1328.495	0	4.44E-16	0	3.97E-07	0.0003668	10.5779	0.00058861	-0.95001	0.39789	5.7	-3.8628	-3.2358	-5.2266	-5.6233	-5.8495
	Std	0	0	0	0	0.098361	8.91E-07	9.39E-05	258.8722	0	0	0	4.36E-07	0.002006	3.6243	0.00046326	0.24904	4.65E-10	8.2385	1.62E-06	0.057468	0.93051	1.6205	1.8698
	Median	0	0	0	0	0.23491	7.54E-07	4.91E-05	-1325.055	0	4.44E-16	0	2.47E-07	3.94E-07	12.6705	0.00033472	-1.0316	0.39789	3	-3.8628	-3.2021	-5.0552	-5.0877	-5.1285
	Best	0	0	0	0	0.14558	9.99E-08	9.51E-06	-1759.336	0	4.44E-16	0	2.60E-08	3.33E-08	0.998	0.00030761	-1.0316	0.39789	3	-3.8628	-3.322	-10.1532	-10.4029	-10.5364
	Worst	0	0	0	0	0.64683	4.36E-06	0.00042427	-849.2697	0	4.44E-16	0	1.95E-06	0.010988	12.6705	0.0016165	-0.21546	0.39789	30	-3.8628	-3.1847	-5.0552	-5.0877	-5.1285
	Runtime	1.77E-02	1.94E-02	2.61E-02	1.86E-02	2.51E-02	1.85E-02	2.49E-02	2.16E-02	1.96E-02	2.01E-02	2.29E-02	4.74E-02	4.61E-02	1.53E-01	1.97E-02	1.94E-02	1.72E-02	1.69E-02	2.26E-02	2.39E-02	2.46E-02	2.68E-02	3.07E-02
AO	Avg	6.75E-56	2.48E-31	7.65E-56	4.37E-29	0.0035285	4.24E-05	0.00019972	-1885.274	0	4.44E-16	0	2.76E-05	3.89E-05	2.9949	0.00054715	-1.0309	0.39829	3.0831	-3.8449	-3.1165	-10.1351	-10.3762	-10.5138
	Std	3.70E-55	1.36E-30	3.00E-55	2.33E-28	0.0070582	7.41E-05	0.00017546	299.2784	0	0	0	3.82E-05	5.56E-05	3.1973	0.00014221	0.0009626	0.00049542	0.098661	0.02021	0.097248	0.022864	0.044945	0.032167
	Median	3.00E-79	5.30E-40	1.29E-76	1.71E-40	0.0012646	9.75E-06	0.00013346	-2091.956	0	4.44E-16	0	1.01E-05	1.04E-05	1.992	0.0005284	-1.0313	0.39814	3.0518	-3.8515	-3.1338	-10.1438	-10.3956	-10.5239
	Best	1.71E-86	3.90E-43	1.76E-82	2.34E-44	1.51E-05	2.75E-07	8.08E-07	-2094.908	0	4.44E-16	0	1.65E-07	1.14E-07	0.998	0.00032507	-1.0316	0.3979	3.0032	-3.8614	-3.2417	-10.1531	-10.4026	-10.5363
	Worst	2.03E-54	7.45E-30	1.43E-54	1.28E-27	0.036103	0.0003568	0.00062573	-1300.625	0	4.44E-16	0	0.00017037	0.00022989	12.6705	0.00097235	-1.027	0.40033	3.4542	-3.7818	-2.9136	-10.0598	-10.2271	-10.4062
	Runtime	2.38E-02	2.44E-02	3.74E-02	2.29E-02	3.33E-02	2.29E-02	3.51E-02	2.95E-02	2.47E-02	2.63E-02	3.24E-02	7.96E-02	7.93E-02	2.95E-01	2.50E-02	2.45E-02	1.98E-02	1.92E-02	3.02E-02	3.31E-02	3.47E-02	3.98E-02	4.75E-02
DOA	Avg	1.67E-60	1.09E-40	1.28E-62	5.17E-40	1.9696	0.030691	0.00036382	-1626.297	0	4.44E-16	0	0.0034812	0.014938	2.577	0.0015645	-1.0316	0.3979	3.0141	-3.853	-3.2554	-8.0766	-7.5483	-7.0245
	Std	9.13E-60	5.97E-40	6.38E-62	2.56E-39	1.5071	0.13293	0.00037042	247.301	0	0	0	0.013639	0.030388	2.2115	0.0037691	2.03E-10	8.03E-05	0.07706	0.034815	0.10893	2.6377	3.1982	3.4648
	Median	6.48E-167	1.04E-86	3.66E-160	3.80E-86	2.0585	1.65E-11	0.00023153	-1655.016	0	4.44E-16	0	1.48E-14	0.00053388	0.998	0.00030803	-1.0316	0.39789	3	-3.8628	-3.3067	-10.1532	-10.4022	-5.8121
	Best	0	2.77E-26	0	1.89E-25	0.0045454	8.29E-29	4.27E-06	-1976.476	0	4.44E-16	0	2.23E-31	6.41E-26	0.998	0.00030749	-1.0316	0.39789	3	-3.8628	-3.322	-10.1532	-10.4029	-10.5364
	Worst	5.00E-59	3.27E-39	3.49E-61	1.40E-38	3.9884	0.70753	0.0017696	-1137.529	0	4.44E-16	0	0.068342	0.15041	10.7632	0.020363	-1.0316	0.39833	3.4221	-3.7124	-2.7836	-2.6305	-2.2234	-2.42
	Runtime	3.01E-02	3.14E-02	3.71E-02	3.08E-02	3.43E-02	3.05E-02	3.62E-02	3.26E-02	3.19E-02	3.39E-02	3.50E-02	5.91E-02	6.13E-02	1.69E-01	3.08E-02	3.28E-02	2.97E-02	2.94E-02	3.23E-02	3.58E-02	3.64E-02	4.00E-02	4.40E-02
HEOA	Avg	2.14E-96	7.18E-42	2.64E-106	3.77E-42	1.7864	0.00034614	0.0001748	-1591.330	1.5011	4.44E-16	0.41453	0.10685	0.00022278	9.9252	0.00034655	-1.0313	0.39839	7.5902	-3.7848	-3.0532	-8.1675	-8.9607	-9.0993
	Std	1.17E-95	3.81E-41	1.25E-105	2.06E-41	1.6486	0.001417	0.00018992	266.9458	2.2193	0	0.15537	0.56752	0.00047813	3.7163	5.16E-05	0.00034294	0.0009707	10.2074	0.048522	0.088319	1.3514	1.3912	1.4441
	Median	1.79E-106	1.27E-46	3.18E-110	6.96E-51	2.7164	4.98E-05	0.00011665	-1584.972	2.76E-11	4.44E-16	0.43759	0.00022146	4.20E-05	12.6705	0.00032757	-1.0315	0.39807	3.0171	-3.7687	-3.0617	-8.5524	-9.417	-9.3924
	Best	7.66E-111	1.56E-48	1.69E-115	7.82E-55	0.0037411	8.00E-06	7.88E-07	-2093.370	0	4.44E-16	0.11486	3.50E-06	2.61E-06	2.9821	0.00031112	-1.0316	0.39789	3	-3.8627	-3.2694	-10.1475	-10.3947	-10.5233
	Worst	6.40E-95	2.09E-40	6.84E-105	1.13E-40	3.5604	0.0078303	0.00087289	-1043.303	5.1568	4.44E-16	0.7827	3.1115	0.0020161	12.6705	0.00050755	-1.0305	0.40308	30.0628	-3.7141	-2.8589	-5.2396	-6.0904	-5.6367
	Runtime	6.43E-02	7.78E-02	8.48E-02	7.75E-02	8.25E-02	7.77E-02	8.35E-02	8.29E-02	7.83E-02	7.88E-02	8.20E-02	1.05E-01	1.06E-01	2.11E-01	7.63E-02	7.22E-02	7.02E-02	6.98E-02	7.75E-02	8.43E-02	8.20E-02	8.48E-02	8.86E-02
RSO	Avg	9.73E-275	1.52E-14	6.37E-275	9.11E-13	3.1709	0.058267	0.00030555	-1473.096	0	4.44E-16	0	0.028236	0.2381	2.8397	0.0011332	-1.0315	0.41331	3.0001	-3.5173	-1.8691	-0.79557	-1.0228	-1.5324
	Std	0	8.24E-14	0	4.96E-13	0.15962	0.10732	0.0003086	192.0728	0	0	0	0.026646	0.055507	2.3186	0.00036987	0.00016318	0.01519	0.0001441	0.23337	0.58316	0.44724	0.71667	0.82236
	Median	0	0	0	0	3.1223	3.61E-05	0.00020656	-1494.103	0	4.44E-16	0	0.036258	0.24106	2.9821	0.0012465	-1.0316	0.40985	3	-3.5325	-1.782	-0.68512	-0.79965	-1.239
	Best	0	0	0	0	2.9765	4.56E-06	1.08E-05	-1894.148	0	4.44E-16	0	1.21E-05	0.087354	0.998	0.00042673	-1.0316	0.39801	3	-3.8445	-2.8002	-2.3548	-3.7777	-3.3682
	Worst	2.92E-273	4.52E-14	1.91E-273	2.72E-13	3.6926	0.25001	0.0012376	-1018.275	0	4.44E-16	0	0.078543	0.33157	10.7632	0.0016627	-1.0311	0.47475	3.0006	-2.9709	-0.80998	-0.35056	-0.4194	-0.60494
	Runtime	8.48E-03	9.46E-03	1.60E-02	8.85E-03	1.39E-02	8.90E-03	1.48E-02	1.20E-02	9.79E-03	1.05E-02	1.36E-02	3.72E-02	3.70E-02	1.41E-01	9.75E-03	9.36E-03	7.02E-03	6.61E-03	1.26E-02	1.43E-02	1.45E-02	1.70E-02	2.04E-02

Table 5: continue.

TSA	Avg	7.67E-60	8.86E-36	9.12E-44	3.29E-20	3.4546	0.47534	0.0013548	-1381.469	4.0481	0.25363	0.22316	0.64056	0.27577	8.7017	0.012323	-1.0264	0.39791	12.2683	-3.8622	-3.2546	-6.41	-6.3488	-7.2225
	Std	2.95E-59	1.74E-35	3.12E-43	9.35E-20	0.84744	0.23057	0.00093632	240.176	2.4832	0.81486	0.11349	1.0856	0.14395	4.655	0.019156	0.011989	2.14E-05	22.6131	0.0014014	0.10458	3.427	3.4658	3.8532
	Median	2.32E-62	2.66E-37	4.55E-47	3.05E-21	3.7612	0.50043	0.0012644	-1371.756	4.1157	5.77E-15	0.21352	0.08021	0.29921	10.7632	0.00089431	-1.0316	0.3979	3	-3.8627	-3.3196	-5.0278	-5.054	-10.1328
	Best	5.55E-66	5.21E-42	5.02E-56	1.07E-23	1.2493	5.41E-05	0.00038378	-1749.341	0	4.00E-15	0.033043	4.02E-05	9.69E-05	1.992	0.00030809	-1.0316	0.39789	3	-3.8628	-3.3215	-10.0615	-10.3604	-10.4447
	Worst	1.57E-58	5.66E-35	1.55E-42	4.89E-19	4.5294	1	0.0044803	-870.2625	11.358	3.5778	0.5526	3.4558	0.53802	18.3043	0.069865	-1	0.39797	92.0449	-3.8568	-2.8387	-2.5946	-1.8268	-1.8499
	Runtime	1.25E-02	1.42E-02	2.08E-02	1.35E-02	1.85E-02	1.35E-02	1.96E-02	1.68E-02	1.50E-02	1.51E-02	1.92E-02	4.16E-02	4.14E-02	1.46E-01	1.35E-02	1.12E-02	8.94E-03	8.66E-03	1.50E-02	1.94E-02	1.83E-02	2.06E-02	2.52E-02
HOA	Avg	5.54E-29	4.13E-15	3.50E-27	6.90E-15	0.11117	0.16001	0.00029171	-1447.964	6.26E-11	6.13E-15	0.012199	0.097223	0.0012008	5.8842	0.00035616	-1.0309	0.3979	4.9891	-3.8613	-3.2132	-8.225	-8.6301	-8.7458
	Std	1.50E-28	5.54E-15	5.37E-27	6.23E-15	0.19334	0.099315	0.00022601	218.1086	3.43E-10	4.13E-15	0.024777	0.22374	0.0018111	2.8333	5.33E-05	0.0016758	4.49E-05	5.4762	0.0017205	0.085056	2.1733	1.2872	1.7849
	Median	8.19E-30	2.09E-15	4.61E-28	4.85E-15	0.030979	0.15601	0.00021306	-1437.419	0	4.00E-15	0	0.046293	0.00069579	6.9033	0.00033199	-1.0316	0.39789	3.0198	-3.862	-3.2486	-9.3569	-8.7734	-9.6448
	Best	2.59E-32	6.79E-17	5.69E-31	5.06E-16	0.00064936	0.026301	1.63E-05	-2089.247	0	4.44E-16	0	0.0033029	3.12E-05	1.0121	0.00030872	-1.0316	0.39789	3	-3.8628	-3.3188	-10.147	-10.3978	-10.5292
	Worst	7.77E-28	2.32E-14	2.20E-26	2.19E-14	0.94526	0.42237	0.0010256	-910.2696	1.88E-09	2.18E-14	0.084423	1.2505	0.0093112	12.6705	0.00053428	-1.0239	0.3981	30.0272	-3.8569	-3.0451	-2.8112	-5.8316	-5.0244
	Runtime	2.38E-02	2.52E-02	3.51E-02	2.77E-02	3.29E-02	2.43E-02	3.33E-02	3.09E-02	2.42E-02	2.95E-02	3.25E-02	5.70E-02	5.81E-02	1.71E-01	3.00E-02	2.98E-02	2.41E-02	2.62E-02	2.92E-02	3.23E-02	3.41E-02	3.63E-02	4.18E-02
GGO	Avg	34.7891	0.47647	157.6489	5.3683	575.2008	41.5943	0.010967	-1361.735	5.265	3.5499	0.63974	2.0748	6.3413	6.8706	0.0092188	-1.0278	0.41527	9.0768	-3.8193	-2.9222	-4.3471	-5.5744	-4.4554
	Std	57.848	0.40872	163.3963	4.1012	809.3555	49.5958	0.0095745	191.9543	2.8295	1.9108	0.50913	1.6687	20.2563	4.6838	0.01591	0.018138	0.04553	17.5173	0.046981	0.29283	2.8678	3.266	2.9755
	Median	11.0653	0.39948	113.7321	4.3167	260.0298	17.0275	0.006518	-1298.217	4.4348	2.9702	0.51963	1.6216	0.73809	5.9327	0.0036003	-1.0316	0.39797	3.0001	-3.8385	-3.0231	-2.6806	-3.8968	-2.8655
	Best	0.072439	0.0241	2.2899	0.48403	2.2195	0.099923	0.00087428	-1850.020	1.6783	0.58457	0.16821	0.0014008	0.11721	0.998	0.00058861	-1.0316	0.39789	3	-3.8625	-3.3117	-9.998	-10.3932	-10.4919
	Worst	271.4262	1.7087	805.098	18.9968	3706.4479	165.0208	0.032414	-1063.160	14.2581	9.9307	2.6074	5.6354	97.7815	17.3744	0.065072	-0.93197	0.59001	90.9986	-3.6947	-2.3362	-0.57741	-1.3908	-1.6326
	Runtime	1.23E-02	1.29E-02	1.57E-02	8.90E-03	1.40E-02	9.29E-03	1.44E-02	1.43E-02	1.30E-02	1.08E-02	1.40E-02	3.73E-02	3.70E-02	1.44E-01	1.02E-02	1.06E-02	1.18E-02	8.06E-03	1.62E-02	1.37E-02	1.49E-02	1.73E-02	2.13E-02
POA	Avg	2270.8254	11.0269	2804.6449	33.6596	338417.2711	2564.8971	0.59676	-1437.203	35.6476	16.0715	24.3405	709029.7322	3061947.241	2.8631	0.0026011	-1.0009	0.43206	3.6877	-3.8353	-2.8878	-1.145	-1.3641	-1.5099
	Std	906.1828	3.1921	1010.5528	8.0994	294977.4001	1169.6649	0.28581	102.4102	5.5591	2.2134	9.6838	1437055.141	3313603.731	1.4352	0.00082477	0.023938	0.028993	1.0887	0.014676	0.11172	0.42614	0.69075	0.74133
	Median	2148.946	11.1759	2590.4245	33.6681	236285.838	2560.1322	0.54096	-1420.916	36.3348	16.7278	24.5012	277076.0918	2166089.349	2.6081	0.0024143	-1.0039	0.42688	3.3642	-3.8401	-2.9006	-0.95685	-1.1313	-1.3211
	Best	462.2314	3.1951	1089.5125	20.2493	24914.8482	195.7593	0.10818	-1711.181	25.2178	10.3065	7.9004	20.579	10939.7309	0.99806	0.0015722	-1.0316	0.39951	3.0167	-3.8529	-3.1304	-2.4496	-4.3269	-4.86
	Worst	3876.3516	15.9873	4991.5629	53.2827	1220518.848	5364.8497	1.39	-1267.834	44.1174	19.5015	48.5343	7619924.914	14720845.41	6.1403	0.0059538	-0.94231	0.51415	8.7143	-3.7975	-2.6695	-0.72578	-0.85493	-0.89673
	Runtime	4.14E-03	4.87E-03	1.14E-02	4.15E-03	9.30E-03	4.23E-03	9.98E-03	7.29E-03	5.72E-03	7.09E-03	9.30E-03	3.25E-02	3.22E-02	1.36E-01	5.52E-03	6.17E-03	3.72E-03	3.43E-03	8.95E-03	9.31E-03	1.02E-02	1.26E-02	1.67E-02

CLFOX ranks top with an average score of 2 out of all other algorithms. After CLFOX, AO and DOA took the second and third place respectively. FOX takes fourth place, and the other algorithms come after. The results show that CLFOX not only outperforms FOX but also surpasses some of the most powerful and recent algorithms.

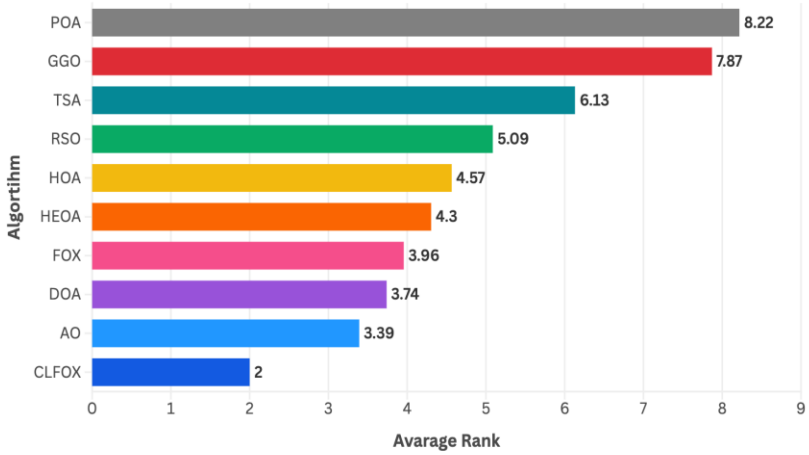

Figure 9: Average rank per algorithm.

Table 6 presents the p-value of CLFOX in comparison to the nine other algorithms. The p-value outputs prove that the CLFOX results for the 22 functions compared to FOX out of the 23 classical benchmark functions are all significant. The NaN values in the tables of statistical results indicate that the two algorithms obtained identical results with zero variance over the runs, and the rank-sum test is statistically undefined. Empty cells indicate cases where the compared algorithm outperformed CLFOX for that function.

The convergence curve of CLFOX and the nine other algorithms over the 23 classical benchmark methods are illustrated in figure 10. CLFOX exhibits superior convergence speed in comparison to almost all other compared algorithms. This can be clearly seen in functions F1-F6, F9-F13 and F15-F17. The efficient search strategy used by CLFOX, which upholds a fair trade-off between exploitation and exploration, is responsible for this quicker convergence. More importantly, in most cases, CLFOX produced noticeably better optimal solutions in addition to quicker convergence than the other nine algorithms.

Overall, CLFOX demonstrates clear superiority over FOX, outperforming it in 22 out of the 23 classical benchmark methods and achieving statistically significant results in almost all cases. The improved performance arises from the many adjustments implemented in the algorithm that boost its efficiency and reliability. For the unimodal functions F1–F6, CLFOX attains superior outcomes primarily due to the application of chaotic maps during the phase of initialization, enhancing population diversity and establishing a more effective beginning stage of the search process.

Table 6: Statistical results (P-value) for CLFOX compared to competitive algorithms using the classical benchmark functions set.

Compared Algorithm	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23
FOX	NaN	NaN	NaN	NaN	0.04515	4.08E-05		3.57E-06	NaN	NaN	NaN	5.46E-06	0.00044592	0.00044592	0.007959	0.0017666	0.0038481	0.00047138	0.00047138	6.28E-06	1.03E-06	0.0015969	0.00017836
AO	1.21E-12	1.21E-12	1.21E-12	1.21E-12		1.78E-10	0.29727		NaN	NaN	NaN	6.70E-11	1.46E-10		8.99E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	6.53E-08			
DOA	1.93E-10	1.21E-12	1.93E-10	1.21E-12	8.15E-05	0.0015969	0.0060971	0.12233	NaN	NaN	NaN	0.077272	0.37904		0.55921		4.56E-11	5.46E-10	8.48E-07	0.49178			
HEOA	1.21E-12	1.21E-12	1.21E-12	1.21E-12	0.27071	3.02E-11	0.85338	0.05746	5.38E-06	NaN	1.21E-12	3.02E-11	3.02E-11	1.79E-06	1.17E-05	3.69E-11	3.02E-11	3.02E-11	3.02E-11	9.76E-10			
RSO	0.002788	0.002788	0.00066167	0.002788	3.02E-11	3.02E-11	0.031466	0.00058737	NaN	NaN	NaN	3.02E-11	3.02E-11		3.02E-11	3.02E-11	3.02E-11	6.07E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
TSA	1.21E-12	1.21E-12	1.21E-12	1.21E-12	3.02E-11	3.02E-11	6.70E-11	2.43E-05	4.57E-12	6.31E-13	1.21E-12	3.02E-11	3.02E-11	0.00097669	3.52E-07	3.69E-11	3.02E-11	3.02E-11	3.02E-11	0.00055611			
HOA	1.21E-12	1.21E-12	1.21E-12	1.21E-12		3.02E-11	0.0018575	0.00026806	0.1608	1.99E-12	0.00031349	3.02E-11	3.02E-11	0.015634	2.77E-05	4.12E-06	1.47E-07	7.77E-09	3.02E-11	0.00055611			
GGO	1.21E-12	1.21E-12	1.21E-12	1.21E-12	3.02E-11	3.02E-11	3.02E-11	7.60E-07	1.21E-12	1.21E-12	1.21E-12	3.02E-11	3.02E-11	0.019878	3.02E-11	0.13727	1.53E-05	1.29E-06	3.02E-11	8.48E-09	0.00033679	0.093341	7.66E-05
POA	1.21E-12	1.21E-12	1.21E-12	1.21E-12	3.02E-11	3.02E-11	3.02E-11	2.00E-05	1.21E-12	1.21E-12	1.21E-12	1.21E-12	3.02E-11		3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11

For the multimodal functions F8–F13, CLFOX consistently outperforms FOX, an advantage attributable to two principal techniques. The initial aspect is the implementation of the Lévy flight mechanism, which improves the algorithm's exploration capacity and decreases the probability of early convergence to local optima. The second involves the periodic introduction of a random search agent every 10 iterations, a method that maintains diversity and prevents stagnation. The modification of the probability distribution across the three primary position-update equations guarantees that each has an equal likelihood of application, thus enhancing the balance between exploitation and exploration phases.

These combined strategies not only improve CLFOX's performance in solving unimodal and multimodal problems but also enhance its robustness regarding the fixed-dimensional methods F14–F23. The outputs confirm that CLFOX maintains a strong equilibrium between exploitation and exploration, leading to consistently better outcomes than FOX across nearly the entire benchmark suite.

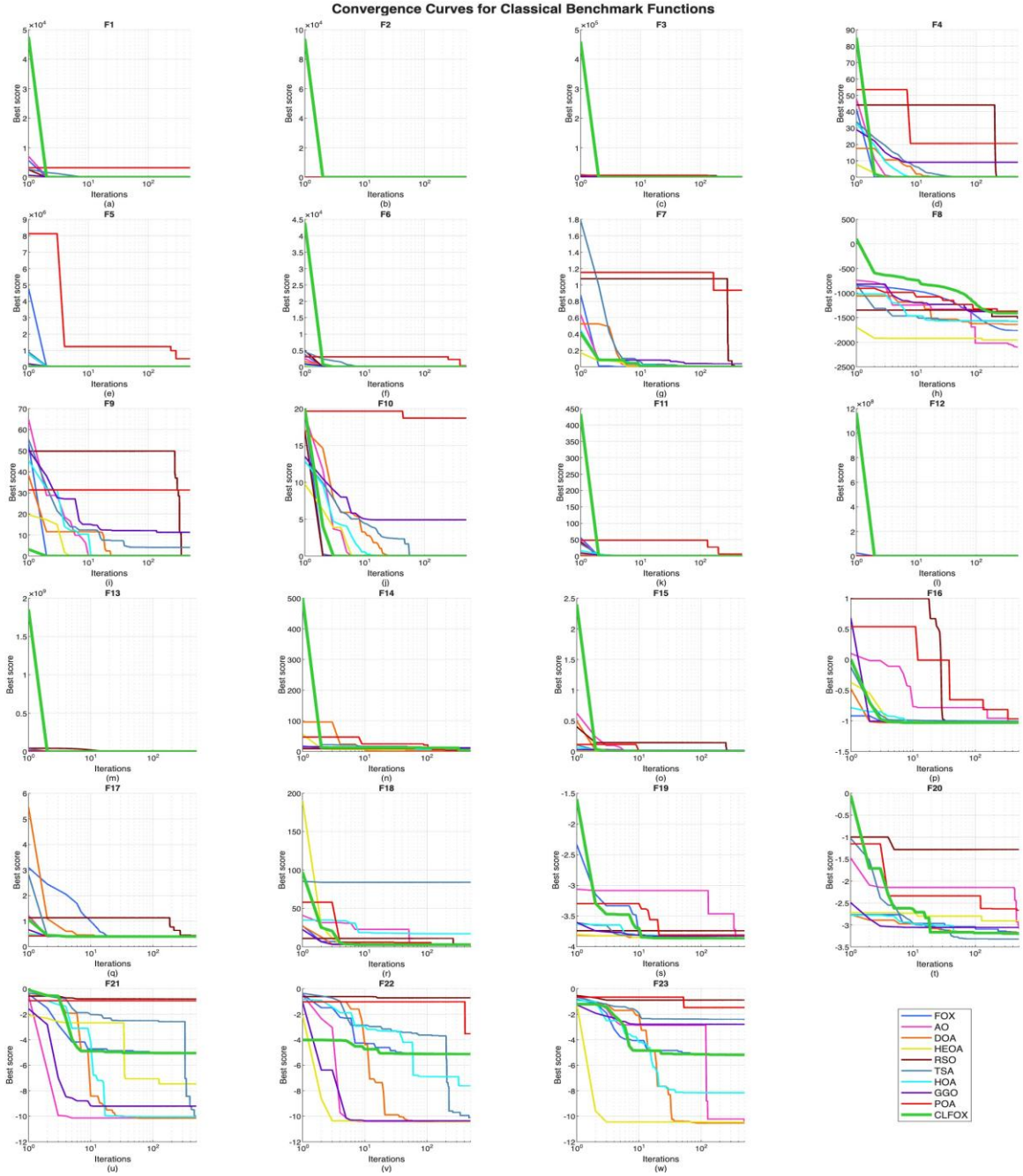

Figure 104: Convergence curve of the classical benchmark functions, from function 1 (a) to function 23 (w).

4.5. CLFOX Evaluation of the CEC2019 Benchmark Functions

The CEC2019 benchmark function is another set that was utilized to assess the performance of CLFOX and to compare it against different algorithms. The set contains 10 functions, all of which are multimodal functions that have several local optima. The initial three functions are neither shifted nor rotated, whereas the subsequent seven functions (4 to 10) have both shifting and rotation [67]. Consequently, they are adept at evaluating the reliability, accuracy, convergence rate, and efficacy of algorithms in overcoming local optima. Algorithms having issue with these functions often do so because they are lacking in the exploration step of their algorithm. Compared to the classical benchmark function set, these problems are considered to be more complex functions to solve.

According to table 7, the results of CLFOX are compared to nine other algorithms. The table 7 proves that CLFOX provides a better result than FOX in 8 out of the 10 methods of the CEC2019 benchmark function set. The only function where CLFOX does not perform well compared to the other algorithms is function 5 (Griewangk's Function). In this function, CLFOX outperforms only the main FOX and POA algorithms, while the other seven algorithms achieve better results. There are no appreciable differences between Griewangk's function and the other functions in the CEC2019 test set that could directly explain this weakness. However, there is one notable distinction: Griewangk's function has fewer local optima compared to the other functions. This difference may explain the weakness of CLFOX in that function, leading to the conclusion that CLFOX could still be further improved in terms of exploitation to achieve better results. Figure 11 shows the ranking of the algorithms, and CLFOX takes first place among them with a total average of 2.2.

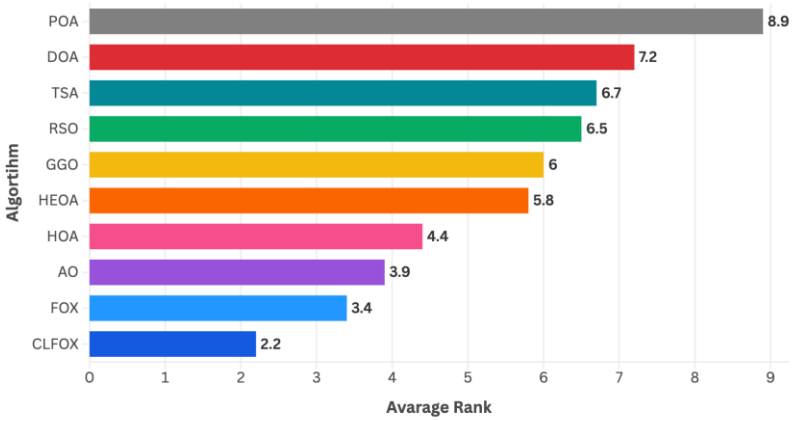

Figure 11: Average rank per algorithm for the CEC2019 set.

Table 7: CEC2019 results for CLFOX and the compared algorithms.

Algorithm		Cec1	Cec2	Cec3	Cec4	Cec5	Cec6	Cec7	Cec8	Cec9	Cec10
CLFOX	Avg	**19048.5849**	**18.3432**	**13.7024**	**998.2637**	5.0352	**4.05E+00**	**315.8113**	5.8437	**3.5479**	2.10E+01
	Std	22187.8783	0.00013028	4.02E-08	664.1001	0.95686	1.09E+00	138.9135	0.26628	0.32337	0.012973
	Median	1	18.3432	13.7024	809.302	4.9418	4.03E+00	263.9037	5.8413	3.4534	2.10E+01
	Best	1	18.343	13.7024	66.6819	3.0211	2.81E+00	1.17E+02	5.1322	3.3538	2.10E+01
	Worst	48006.1588	18.3436	13.7024	2740.5819	7.2009	6.57E+00	859.653	6.293	5.1138	2.10E+01
	Runtime	1.43E+00	3.01E-02	4.52E-02	3.12E-02	3.03E-02	6.58E-01	3.12E-02	3.11E-02	2.84E-02	3.32E-02
FOX	Avg	20984.4492	18.3436	13.7026	2131.5812	6.7142	4.81E+00	3.54E+02	5.7195	3.8197	2.10E+01
	Std	22898.1578	0.00026154	0.00063293	1220.1721	1.307	1.02E+00	2.38E+02	0.50898	0.48733	0.0064206
	Median	1	18.3435	13.7024	2067.4689	6.6412	4.83E+00	2.93E+02	5.8049	3.7012	2.10E+01
	Best	1	18.3432	13.7024	390.0414	4.4931	3.20E+00	6.13E+01	4.5274	3.3962	2.10E+01
	Worst	52516.166	18.3443	13.7049	5150.2418	8.7606	6.91E+00	956.9258	6.6835	5.3492	2.10E+01
	Runtime	1.42E+00	2.13E-02	3.54E-02	2.47E-02	2.36E-02	6.47E-01	2.41E-02	2.41E-02	2.15E-02	2.62E-02
AO	Avg	6.52E+04	1.84E+01	1.37E+01	1.74E+03	2.8837	1.22E+01	433.6434	5.7822	13.0339	2.14E+01
	Std	1.15E+04	1.75E-02	1.35E-05	9.51E+02	0.27499	6.30E-01	179.3923	0.48847	12.5987	0.21425
	Median	6.32E+04	1.84E+01	1.37E+01	1.60E+03	2.89	1.24E+01	398.0301	5.8598	7.3146	2.15E+01
	Best	4.69E+04	1.84E+01	1.37E+01	1.24E+02	2.32E+00	1.10E+01	1.28E+02	4.8152	5.1709	2.06E+01
	Worst	9.36E+04	1.84E+01	1.37E+01	3.34E+03	3.4761	13.5693	774.7863	6.809	60.2146	2.17E+01
	Runtime	2.89E+00	2.78E-02	5.74E-02	3.32E-02	3.34E-02	1.30E+00	3.39E-02	3.70E-02	2.80E-02	3.69E-02

Table 7: continue.

DOA	Avg	8.18E+04	1.84E+01	1.37E+01	4.24E+03	3.8122	12.9275	888.8147	6.7152	487.6286	2.15E+01
	Std	9.37E+04	1.04E-01	5.25E-04	2.79E+03	0.85817	0.99297	342.7094	0.49027	400.8926	0.83714
	Median	4.34E+04	1.83E+01	1.37E+01	4.00E+03	3.6108	1.31E+01	887.833	6.7422	350.0095	2.17E+01
	Best	36538.2527	1.83E+01	13.7024	5.89E+02	2.4379	1.04E+01	2.93E+02	5.6886	38.3055	1.71E+01
	Worst	4.51E+05	1.87E+01	1.37E+01	1.27E+04	6.2452	14.2239	1837.6864	7.4853	1855.1527	2.18E+01
	Runtime	1.45E+00	3.23E-02	4.73E-02	3.52E-02	3.41E-02	6.54E-01	3.62E-02	3.65E-02	3.41E-02	3.56E-02
HEOA	Avg	5.44E+04	1.85E+01	1.37E+01	2.82E+03	3.2887	12.117	957.4367	6.6229	398.4102	2.16E+01
	Std	5.73E+03	1.34E-01	1.76E-05	1.39E+03	0.36816	0.77625	188.2869	0.2793	245.0183	0.10685
	Median	5.33E+04	1.85E+01	1.37E+01	2.77E+03	3.2796	1.21E+01	967.2556	6.6667	4.61E+02	2.16E+01
	Best	4.40E+04	1.84E+01	1.37E+01	5.66E+02	2.7189	1.05E+01	5.11E+02	5.9581	30.8905	2.13E+01
	Worst	6.58E+04	1.90E+01	1.37E+01	6.66E+03	4.3337	13.4309	1359.8417	7.1293	766.1736	2.17E+01
	Runtime	1.49E+00	1.05E-01	1.17E-01	9.18E-02	9.18E-02	7.13E-01	9.23E-02	9.21E-02	8.97E-02	9.37E-02
RSO	Avg	5.76E+04	1.86E+01	1.37E+01	9.46E+03	4.6101	12.0658	759.7364	6.3874	615.0872	2.15E+01
	Std	8022.8338	3.63E-01	3.03E-05	2.32E+03	0.31355	0.75009	241.6487	0.37359	98.2145	0.079069
	Median	54846.1182	18.5787	13.7025	9536.4157	4.5814	1.22E+01	751.5693	6.4982	659.5245	2.15E+01
	Best	48493.6534	18.3456	13.7024	3488.0155	4.1946	1.01E+01	2.99E+02	5.3801	297.7361	2.13E+01
	Worst	7.77E+04	1.97E+01	1.37E+01	1.31E+04	5.3998	13.5402	1214.7672	6.8869	678.6778	2.16E+01
	Runtime	1.42E+00	9.85E-03	2.37E-02	1.31E-02	1.34E-02	6.41E-01	1.37E-02	1.38E-02	9.98E-03	1.44E-02
TSA	Avg	2.64E+08	1.93E+01	1.37E+01	4.23E+03	3.6972	12.1474	648.9509	6.3751	276.1972	21.4843
	Std	4.46E+08	7.16E-01	1.51E-03	2.90E+03	0.7026	0.66004	256.0807	0.53345	393.8778	0.081617
	Median	6.16E+06	1.93E+01	1.37E+01	4.15E+03	3.516	12.2234	645.9642	6.499	142.8302	2.15E+01
	Best	4.02E+04	1.84E+01	1.37E+01	1.46E+02	2.688	1.04E+01	281.4252	5.1036	3.7988	2.13E+01
	Worst	1.82E+09	2.06E+01	1.37E+01	1.14E+04	5.3654	13.2805	1360.986	7.0673	1733.149	21.6297
	Runtime	1.44E+00	2.25E-02	3.77E-02	2.12E-02	2.14E-02	6.47E-01	2.15E-02	2.20E-02	1.78E-02	2.24E-02
HOA	Avg	5.35E+04	1.88E+01	1.37E+01	3.86E+03	3.3814	11.7892	474.7093	5.7082	7.55E+02	2.12E+01
	Std	7.86E+03	2.50E-01	4.33E-06	1.31E+03	0.43373	0.81102	143.9718	0.56294	4.95E+02	9.77E-01
	Median	5.22E+04	1.88E+01	1.37E+01	3.95E+03	3.4038	12.0046	479.7732	5.6655	681.4956	2.15E+01
	Best	3.98E+04	1.84E+01	1.37E+01	1.16E+03	2.5829	9.8459	2.26E+02	4.5173	101.2396	1.61E+01
	Worst	7.97E+04	1.96E+01	1.37E+01	6.34E+03	4.3315	12.814	800.3575	6.9001	1.97E+03	2.16E+01
	Runtime	1.48E+00	3.13E-02	4.22E-02	3.23E-02	3.22E-02	6.67E-01	3.28E-02	3.44E-02	2.65E-02	3.37E-02
GGO	Avg	97315896563	427.7245	13.703	7095.6185	4.0087	9.9303	442.7255	5.5868	691.9275	21.0655
	Std	136318688210	792.8232	0.00079247	4237.1663	0.65054	1.5094	159.9242	0.50783	494.7138	0.91475
	Median	61164568613	186.5389	13.7027	5816.5641	4.0415	10.1472	483.2577	5.6991	519.252	21.1948
	Best	2173686434	21.2666	13.7024	1565.46	2.8608	6.3927	134.1041	4.4038	15.7141	16.2714
	Worst	723744748854	4239.2397	13.7059	19314.6125	5.1768	12.4674	843.2504	6.5398	2107.0827	21.5187
	Runtime	1.46E+00	9.69E-03	2.33E-02	1.23E-02	1.57E-02	6.37E-01	1.35E-02	1.41E-02	1.37E-02	1.52E-02
POA	Avg	647052760084	8769.0863	13.7026	14135.9045	6.5489	12.1182	676.3989	6.6702	5149.2485	21.5239
	Std	347516952047	2103.5102	9.18E-05	2976.5995	0.63417	0.50817	147.3041	0.2198	1130.6767	0.07662
	Median	576039484415	8837.3035	13.7025	14995.522	6.6518	12.0378	705.0797	6.6504	5390.9068	21.5306
	Best	139851895054	3300.4098	13.7025	6153.3699	4.7275	10.8911	196.3915	6.0269	2243.8198	21.3764
	Worst	1635610610011	11985.934	13.703	17916.3577	7.5025	12.8186	893.5511	7.0946	6688.0816	21.6743
	Runtime	1.45E+00	5.00E-03	1.86E-02	9.45E-03	9.95E-03	6.43E-01	1.03E-02	1.05E-02	6.19E-03	1.09E-02

The p-value of FOX in relation to the nine other algorithms in the CEC2019 benchmark methods is displayed in table 8. The p-value results demonstrate that CLFOX performs better than FOX in eight functions, seven of which are significant.

Figure 12 shows the convergence curve of CLFOX and the nine other algorithms across the 10 CEC2019 benchmark functions. According to the figure 10, CLFOX converges more quickly than practically every other algorithm that it was compared to. According to the convergence curves, CLFOX has an accelerated speed of convergence compared to all other algorithms in most of the methods, especially in functions CEC1-CEC3, CEC6-CEC8 and CEC10.

This better convergence curve of CLFOX is because of the improvement of the exploration of the algorithm throughout the iterations.

Table 8: Statistical results (P-value) for CLFOX compared to the competitive algorithms using the CEC2019 benchmark functions set.

Compared Algorithm	Cec01	Cec2	Cec3	Cec4	Cec5	Cec6	Cec7	Cec8	Cec9	Cec10
FOX	0.51198	1.47E-07	2.43E-05	7.66E-05	7.22E-06	0.0086844	0.8418		0.00058737	
AO	1.98E-11	3.02E-11	3.02E-11	0.0046371		3.02E-11	0.003183		3.02E-11	5.57E-10
DOA	0.00015929	0.17612	0.007818	3.35E-08		3.02E-11	8.89E-10	3.35E-08	3.02E-11	5.57E-10
HEOA	4.04E-11	3.02E-11	3.02E-11	1.87E-07		3.02E-11	8.99E-11	3.16E-10	3.02E-11	3.02E-11
RSO	1.79E-11	3.02E-11	3.02E-11	3.02E-11		3.02E-11	4.18E-09	6.53E-07	3.02E-11	3.02E-11
TSA	1.62E-10	3.02E-11	4.08E-11	6.74E-06		3.02E-11	1.25E-07	2.00E-05	6.07E-11	3.02E-11
HOA	3.20E-10	3.02E-11	0.0022658	2.87E-10		3.02E-11	2.77E-05		3.02E-11	8.48E-09
GGO	1.79E-11	3.02E-11	3.02E-11	8.15E-11		3.34E-11	0.0011143		3.02E-11	5.57E-10
POA	1.79E-11	3.02E-11	3.02E-11	3.02E-11	5.53E-08	3.02E-11	5.46E-09	6.07E-11	3.02E-11	3.02E-11

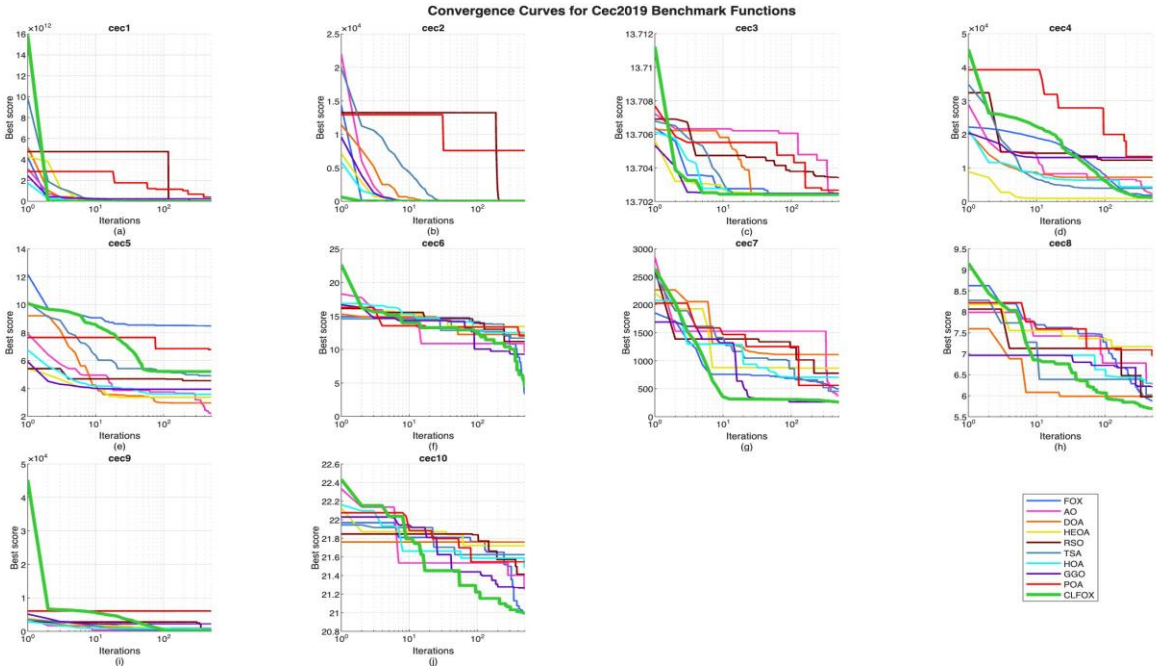

Figure 12: Convergence curves of the CEC2019 set functions, from function cec01 (a) to function cec10 (j).

Overall, CLFOX clearly outperforms FOX, developing a better outcome in 8 of the 10 CEC2019 benchmark methods and almost always obtaining statistically significant outcomes. The improved performance results from several changes made to the algorithm that increase its dependability and efficiency. The techniques that led to these improvements include introducing the Lévy flight strategy to the algorithm and adding an interval-based operator that inserts a random agent once every 10 iterations. These two strategies enabled the algorithm to search the search region more, maintain diversity, and, most importantly for the CEC2019 functions, helping it escape local optima, as most of the CEC2019 test benchmark functions contain numerous local optima.

4.6. CLFOX Evaluation on CEC2022 Benchmark Functions

Finally, CEC2022 is used to evaluate the effectiveness of CLFOX related to the other algorithms. The set contains 12 functions categorized into four types: one unimodal function which is F1, while F2–F5 are basic functions, F6–F8 are hybrid functions, and the three last functions inside the set that have composite specifications are functions F9–F12. The CEC2022 function set is included in this paper to assess the proposed algorithm and to highlight its strengths and weaknesses. Since the set contains four distinct kinds of functions, it provides a comprehensive way to evaluate an algorithm.

Table 9 presents the results of CLFOX and the nine other comparison algorithms regarding the CEC2022 benchmark test functions. CLFOX outperforms FOX in all 12 test functions and achieves better outcomes than all other algorithms in half of them, specifically functions CEC1, CEC2, CEC6, CEC9, CEC10, and CEC11. It is notable that CLFOX performs best in the unimodal function (CEC1), which is directly related to the exploitation capability of the algorithm. In other words, the exploitation potential

of CLFOX is higher than all other algorithms in the CEC2022 test functions. In the composite functions, compared to other algorithms, CLFOX surpasses them in three out of the four functions (CEC9–CEC11).

Figure 13 is an illustration based on the data presented in table 9. Based on how well each algorithm performed across the 12 CEC2022 benchmark functions, it displays the average rank and placement of each algorithm. Second position goes to the CLFOX algorithm, following AO. Compared to the greater differences among the other algorithms, it is noteworthy that the difference between the first-place algorithm (AO), that has an average rank of 2.17, and the second-place algorithm (CLFOX), is not that great. For instance, FOX has a total average rank of 5.83, whereas HOA and DOA, in third and fourth place, have an average rank of 5.08.

Table 9: Cec2022 results for CLFOX and the compared algorithms.

Algorithm		Cec01	Cec2	Cec3	Cec4	Cec5	Cec6	Cec7	Cec8	Cec9	Cec10	Cec11	Cec12
CLFOX	Avg	**2112.1099**	**443.3383**	644.6327	834.1933	1470.7457	**3.43E+03**	2103.8324	2239.3097	**2621.5731**	**2.52E+03**	**2757.6399**	2.93E+03
	Std	2343.5424	35.0771	1.22E+01	5.5272	50.3501	1.83E+03	34.8187	16.2807	42.9911	42.0117	140.518	3.23E+01
	Median	1310.0258	451.7484	646.8995	832.8336	1467.3999	2.76E+03	2098.0626	2235.0238	2633.6975	2.50E+03	2751.0751	2.93E+03
	Best	300	400.0501	617.8048	828.8537	1424.3202	1.88E+03	2.05E+03	2220.8199	2535.3987	2.50E+03	2600.028	2.87E+03
	Worst	10654.3528	544.7261	661.7286	850.7427	1705.8793	8.08E+03	2181.6817	2281.5209	2684.2503	2.68E+03	3201.1515	3.02E+03
	Runtime	3.72E-02	3.49E-02	3.75E-02	3.52E-02	3.57E-02	3.48E-02	3.95E-02	4.08E-02	3.82E-02	3.85E-02	4.26E-02	4.24E-02
FOX	Avg	5476.1535	478.8679	655.1654	835.5199	1483.9534	3.71E+03	2.17E+03	2414.8032	2627.1376	2.84E+03	2854.9662	3.00E+03
	Std	6800.7759	74.4403	8.1058	8.6191	87.833	2.10E+03	8.68E+01	152.4649	53.4669	397.3918	369.0141	5.66E+01
	Median	2531.439	469.233	656.552	832.8336	1467.9691	2.63E+03	2.17E+03	2435.7763	2629.4671	2.68E+03	2761.6479	2.98E+03
	Best	300.0005	400.067	637.7716	823.879	1431.2374	1.86E+03	2.06E+03	2218.1114	2531.8307	2.50E+03	2600.0273	2.92E+03
	Worst	24893.5924	711.3855	675.0144	857.7075	1907.6793	8.25E+03	2347.1431	2655.5324	2745.0394	3.65E+03	4543.5898	3.12E+03
	Runtime	2.91E-02	2.86E-02	3.14E-02	2.88E-02	2.95E-02	2.86E-02	3.36E-02	3.49E-02	3.20E-02	3.26E-02	3.65E-02	3.59E-02
AO	Avg	6.41E+03	4.96E+02	6.26E+02	8.25E+02	1116.0874	2.48E+05	2066.1399	2232.2436	2648.6904	2.56E+03	2788.5149	2.87E+03
	Std	2.73E+03	8.16E+01	8.48E+00	6.67E+00	145.0746	5.99E+05	30.2538	3.8147	38.3657	65.7118	115.7541	1.06E+01
	Median	5.59E+03	4.72E+02	6.24E+02	8.24E+02	1056.7654	8.36E+04	2055.8937	2231.9356	2652.9149	2.51E+03	2761.1564	2.87E+03
	Best	2.41E+03	4.21E+02	6.15E+02	8.11E+02	9.62E+02	1.36E+04	2.03E+03	2225.439	2560.3707	2.50E+03	2638.8259	2.87E+03
	Worst	1.57E+04	7.63E+02	6.47E+02	8.38E+02	1449.7116	3322479.145	2144.5871	2240.3009	2709.7271	2.65E+03	3256.2455	2922.2682
	Runtime	4.33E-02	4.18E-02	4.97E-02	4.34E-02	4.49E-02	4.23E-02	5.23E-02	5.50E-02	4.89E-02	5.07E-02	6.01E-02	5.99E-02
DOA	Avg	6.54E+03	5.68E+02	6.33E+02	8.36E+02	1256.5651	7448410.316	2080.318	2256.2564	2630.439	2.61E+03	3022.6131	2920.0928
	Std	4.04E+03	1.68E+02	1.35E+01	1.03E+01	261.479	37223025.01	31.6749	53.2306	48.2495	97.7115	319.9435	58.7654
	Median	6.29E+03	5.01E+02	6.35E+02	8.33E+02	1167.6754	2.10E+03	2072.3259	2233.5527	2627.8311	2.64E+03	2895.0469	2.90E+03
	Best	901.7775	4.07E+02	611.3333	8.22E+02	951.0797	1.85E+03	2.04E+03	2222.0633	2529.5854	2.50E+03	2704.1997	2.87E+03
	Worst	1.58E+04	1.11E+03	6.67E+02	8.61E+02	1855.1815	204092040.8	2137.8592	2443.1084	2735.9407	2.83E+03	4108.2924	3132.2899
	Runtime	4.22E-02	4.13E-02	4.47E-02	4.30E-02	4.35E-02	4.15E-02	4.78E-02	4.65E-02	4.41E-02	4.45E-02	4.82E-02	4.81E-02
HEOA	Avg	9.73E+03	5.08E+02	6.56E+02	8.49E+02	1630.6338	1518413.301	2127.7216	2237.113	2681.5204	2.66E+03	2810.1247	2964.9043
	Std	2.17E+03	7.69E+01	1.04E+01	1.18E+01	306.8715	1907849.766	37.5544	11.412	32.6577	19.9619	136.7484	80.5393
	Median	9.40E+03	4.89E+02	6.55E+02	8.46E+02	1576.222	8.87E+05	2139.4401	2234.577	2.69E+03	2.66E+03	2766.0302	2945.1858
	Best	6.52E+03	4.20E+02	6.38E+02	8.28E+02	1118.5221	3.14E+04	2.04E+03	2226.6406	2623.0911	2.62E+03	2730.9622	2.87E+03
	Worst	1.64E+04	7.71E+02	6.76E+02	8.72E+02	2211.8137	7167317.102	2171.602	2289.7921	2731.6014	2.71E+03	3359.9409	3159.5637
	Runtime	9.84E-02	9.83E-02	1.00E-01	9.78E-02	9.89E-02	9.82E-02	1.02E-01	1.04E-01	1.01E-01	1.01E-01	1.05E-01	1.06E-01
RSO	Avg	4.50E+03	1.05E+03	6.50E+02	8.45E+02	1395.4572	5042462.696	2104.3379	2290.1098	2706.9982	2.53E+03	3170.2802	2908.4781
	Std	2295.0529	2.65E+02	7.82E+00	8.05E+00	99.5683	13030679.49	31.4482	64.465	52.573	53.0061	236.1304	33.1048
	Median	4182.3628	1016.2001	650.7031	846.5824	1422.99	7.41E+04	2096.233	2260.66	2701.2113	2.52E+03	3085.2786	2900.4393
	Best	1826.9598	598.7777	632.9379	827.4301	1171.2172	5.17E+03	2.06E+03	2234.2956	2605.7851	2.51E+03	2904.2891	2.87E+03
	Worst	1.11E+04	1.60E+03	6.68E+02	8.62E+02	1588.5994	47217368.86	2209.2037	2461.5516	2882.439	2.72E+03	3875.089	2973.4602
	Runtime	1.74E-02	1.64E-02	2.04E-02	1.74E-02	1.80E-02	1.66E-02	2.15E-02	2.29E-02	1.99E-02	2.12E-02	2.69E-02	2.64E-02
TSA	Avg	7.00E+03	5.50E+02	6.34E+02	8.50E+02	1529.027	4023041.37	2086.6605	2289.3293	2667.713	3038.7913	3085.3962	2916.3921
	Std	3.16E+03	1.84E+02	1.73E+01	1.42E+01	478.8098	12065929.58	38.4333	66.3905	59.7916	536.9521	348.3374	55.1146
	Median	7.58E+03	4.75E+02	6.31E+02	8.51E+02	1366.8746	74965.0859	2074.9913	2250.4097	2664.3549	2.80E+03	2976.4936	2904.7919
	Best	3.32E+02	4.02E+02	6.09E+02	8.23E+02	984.3619	1.23E+04	2027.5469	2210.7499	2562.6021	2.50E+03	2725.2498	2.87E+03
	Worst	1.27E+04	1.19E+03	6.81E+02	8.88E+02	2557.0975	40041000.88	2164.6973	2445.1573	2811.6051	4086.8868	3960.8501	3135.6333
	Runtime	2.54E-02	2.41E-02	2.84E-02	2.52E-02	2.59E-02	2.44E-02	2.91E-02	3.07E-02	2.77E-02	2.90E-02	3.44E-02	3.39E-02
HOA	Avg	6.64E+03	7.55E+02	6.31E+02	8.31E+02	1145.5517	58946173.52	2071.072	2245.0028	2.69E+03	2.61E+03	3214.4958	2994.3032
	Std	2.16E+03	2.46E+02	7.95E+00	8.92E+00	149.6293	81892158.92	23.5798	51.4519	3.02E+01	1.35E+02	380.1734	41.1191
	Median	6.51E+03	6.83E+02	6.32E+02	8.31E+02	1102.2628	15538957.58	2067.054	2229.612	2696.7304	2.62E+03	3077.0573	2982.4797
	Best	2.38E+03	4.60E+02	6.15E+02	8.10E+02	953.3065	159326.4829	2.03E+03	2218.5903	2641.8631	2.50E+03	2765.3605	2911.2088
	Worst	1.21E+04	1.37E+03	6.48E+02	8.52E+02	1581.1502	296203755.8	2116.8436	2447.9534	2.75E+03	3.25E+03	3927.3618	3100.0013
	Runtime	3.75E-02	3.64E-02	4.05E-02	3.80E-02	3.86E-02	3.75E-02	4.30E-02	4.23E-02	4.01E-02	4.12E-02	4.58E-02	4.55E-02
GGO	Avg	11803.8327	1035.9802	631.2195	835.844	1253.1506	78547532.07	2073.5443	2251.9354	2714.9483	2669.5517	3451.7968	3020.9012
	Std	5259.7825	360.9367	9.1267	8.7474	229.9306	151695327.8	38.2468	40.5338	35.9168	225.0779	307.8748	65.4674
	Median	11497.6332	973.8383	630.3546	835.5306	1202.5295	15290440.03	2061.8138	2235.2659	2717.2964	2636.997	3459.9178	3011.8496
	Best	4156.239	535.9717	613.6999	819.7964	943.8164	1973.5551	2029.4926	2223.7326	2652.1184	2502.0945	2935.8558	2898.6727
	Worst	25876.5644	2040.323	647.6138	850.5062	1878.8939	714972772.2	2167.1031	2356.635	2798.4624	3471.3404	4094.3469	3258.3949
	Runtime	1.86E-02	2.08E-02	2.06E-02	1.84E-02	1.90E-02	1.79E-02	2.26E-02	2.40E-02	2.13E-02	2.12E-02	2.44E-02	2.50E-02

Table 9: continue.

POA	Avg	31853.5252	811.5534	673.7671	915.1987	4085.8508	97431676.55	2129.4345	2259.5299	2766.2476	2575.2867	3372.4031	2872.7916
	Std	6080.6468	102.4288	6.84E+00	11.593	548.5708	62773335.48	18.8134	10.6088	47.8896	28.3395	219.5515	1.9652
	Median	32652.8155	819.5732	673.1029	918.689	4082.2985	84099577.14	2128.1511	2257.934	2775.7515	2567.5705	3371.9531	2873.3183
	Best	19251.2482	548.6522	660.5794	883.6258	3170.1816	2657350.715	2083.059	2243.696	2654.3733	2528.2104	2960.2921	2867.3396
	Worst	42503.8787	1017.9451	688.7793	934.119	5284.4335	261479298.3	2163.0691	2281.2634	2845.7601	2632.3964	3754.3432	2875.8948
	Runtime	1.35E-02	1.28E-02	1.68E-02	1.38E-02	1.44E-02	1.29E-02	1.79E-02	1.93E-02	1.61E-02	1.81E-02	2.44E-02	2.35E-02

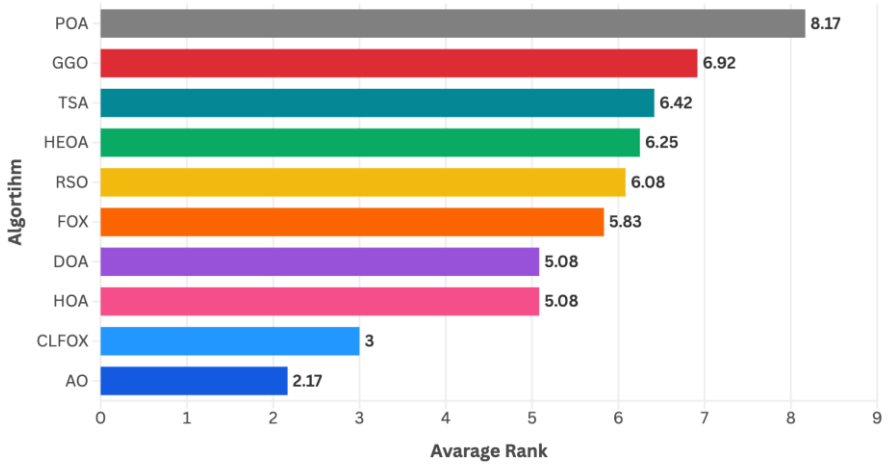

Figure 13: Average rank per algorithm for the CEC2022 set.

The p-value of FOX in relation to the other nine algorithms is displayed in table 10. The p-value results demonstrate the significance of the CLFOX outcomes in seven functions compared to FOX among the 12 CEC2022 benchmark functions.

Figure 14 illustrates the convergence curves for CLFOX and the other compared algorithms. It is observable that in functions CEC7 and CEC12, CLFOX does not exhibit a fast or stable convergence curve, as the algorithm progresses to a certain point and then the curve suddenly drops. In contrast, CLFOX shows good and fast convergence behavior in all other functions of the CEC2022 benchmark set.

Table 10: Statistical results (P-value) for CLFOX compared to the competitive algorithms using the CEC2022 benchmark functions set.

Compared Algorithm	CEC1	CEC2	CEC3	CEC4	CEC5	CEC6	CEC7	CEC8	CEC9	CEC10	CEC11	CEC12
FOX	0.016285	0.033874	0.00072951	0.30418	0.52014	0.3871	0.0012362	3.16E-05	0.66273	1.64E-05	0.28378	6.28E-06
AO	4.31E-08	0.00062027				3.0199E-11			0.015014	0.00095207	0.093341	
DOA	5.86E-06	4.42E-06		0.87663		0.39527		0.52978	0.64142	0.00010407	2.00E-05	
HEOA	3.47E-10	3.16E-05	0.0021566	5.19E-07	0.053685	3.02E-11	0.010315		1.86E-06	4.62E-10	0.021506	0.42896
RSO	2.43E-05	3.02E-11	0.16238	2.38E-07		6.07E-11	0.87663	3.83E-06	2.19E-08	0.00030059	5.07E-10	
TSA	2.38E-07	0.0015178		1.86E-06	0.19073	3.02E-11		0.023243	0.0022658	1.09E-05	6.74E-06	
HOA	1.01E-08	1.09E-10				3.02E-11		0.17145	2.83E-08	9.53E-07	3.65E-08	7.09E-08
GGO	4.62E-10	3.34E-11		0.37904		1.29E-09		0.40354	7.38E-10	3.52E-07	8.99E-11	2.83E-08
POA	3.02E-11	3.02E-11	3.34E-11	3.02E-11	3.02E-11	3.02E-11	0.00062027	9.51E-06	1.21E-10	1.43E-08	7.39E-11	

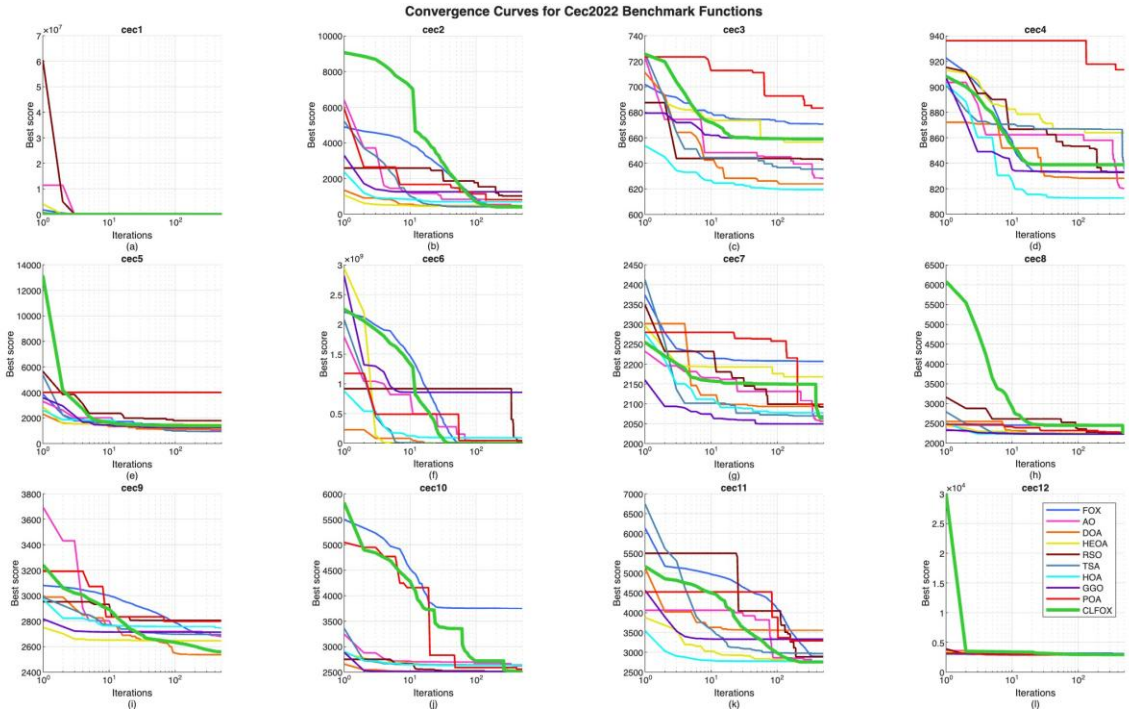

Figure 14: Convergence curve of the CEC2022 benchmark functions, from function cec1 (a) to function cec12 (l).

Overall, CLFOX outperforms FOX in all 12 CEC2022 benchmark functions and achieves statistically significant results in seven of them, indicating clear superiority. The improved performance is the result of several modifications to the algorithm that increase its dependability and efficiency. However, it can also be observed that the AO algorithm surpasses CLFOX in this set, although none of the other algorithms outperformed CLFOX in the other benchmark sets.

There are four functions in which CLFOX does not achieve the best performance: CEC3, CEC5, CEC7, and CEC12. These functions are multimodal, shifted, and rotated. This suggests that, despite the overall improvements and strong performance of CLFOX, there is still room to boost the algorithm's exploration capabilities and establish a better equilibrium between exploration and exploitation.

4.7. Statistical Results of Real-World Problems

CLFOX has been assessed using six engineering design issues and compared with nine alternative optimization techniques. Each algorithm was run with 30 search agents over 500 iterations, yielding 15,000 function evaluations per run, facilitating a fair and thorough comparison. Table 11 displays the outcomes of CLFOX alongside the comparison algorithms for each engineering issue. The findings indicate that CLFOX outperforms FOX in five of the six engineering problems, with the only case where FOX performs slightly better being the Tension/Compression String Design (TSD) problem. Furthermore, CLFOX outperforms all remaining algorithms across all tested real-world problems with the exception of the TSA algorithm, which achieves results marginally close to CLFOX.

All tested problems are multimodal. Due to the new exploration-enhancing techniques introduced in CLFOX such as Lévy flight and interval-based search adjustments, the algorithm achieves superior performance in multimodal landscapes. In addition, all real-world problems used in this study are fixed-dimension, providing evidence that CLFOX performs well in fixed-dimension optimization tasks. Further testing on scalable problems is recommended to fully assess its scalability. The engineering problems encompass many variable types: several are continuous, including the Three-bar Truss Design problem (TBD), Cantilever Beam Design problem (CBD), and Welded Beam Design problem (WBD). The Gear Train Design problem (GTD) problem is discrete, and the Pressure Vessel Design problem (PVD) is a mixed continuous-discrete issue. CLFOX exhibits robust performance across all areas, highlighting its capacity to manage a range of variables. This versatility arises from the refined

methods introduced in CLFOX, including the optimized primary search criterion and the implementation of chaotic population initialization in place of conventional random initialization. The only engineering problem that CLFOX could not outperform FOX on was the TSD design problem. Figure 15 shows the CLFOX and FOX convergence curves when solving the TSD problem.

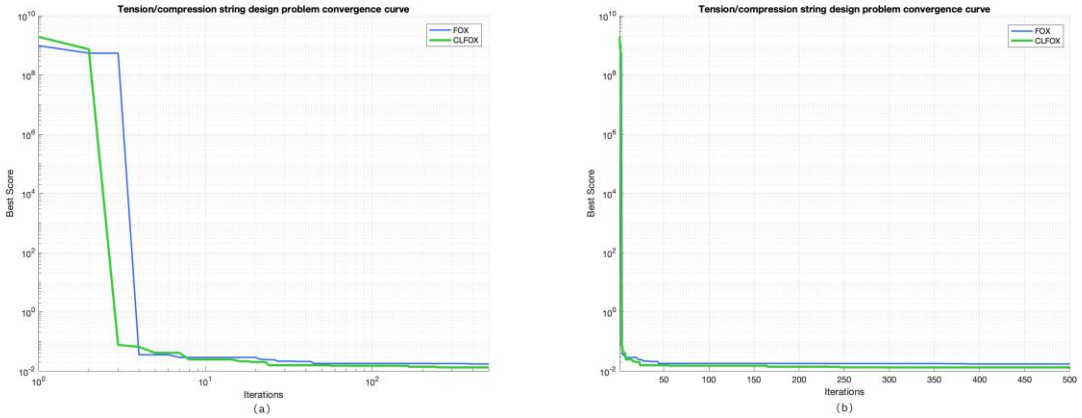

Figure 15: CLFOX vs FOX string design convergence curves, where (a) is a zoomed-in view of (b) the highlighting of the early iterations.

As shown in table 11, the total average of FOX is less than CLFOX in the TSD problem. In other words, if the total average is considered, FOX reaches a more optimal solution. The average of FOX is less than CLFOX because in the very first iterations, CLFOX started with larger values that made a significant change to the average. The figure 15 also shows that CLFOX converges faster than FOX and reaches a more optimal solution.

Table 11: Engineering problem results for CLFOX and compared algorithms.

Algorithm		PVD	TSD	TBD	GTD	CBD	WBD
CLFOX	Avg	8313.4139	0.014507	263.8965	5.86E-16	1.34	1.81E+00
	Std	2778.4215	0.0017327	1.03E-03	7.96E-16	1.54E-05	2.19E-01
	Median	7336.3048	0.01419	263.8962	4.05E-16	1.34	1.73E+00
	Best	5897.2163	0.012696	263.8959	1.57E-19	1.34	1.49E+00
	Worst	19006.1165	0.017725	263.9016	4.08E-15	1.34	2.21E+00
	Runtime	5.67E-02	5.35E-02	4.62E-02	2.56E-02	3.83E-02	6.60E-02
FOX	Avg	12479.8236	0.01369	263.8983	2.60E-15	1.34	2.03E+00
	Std	5689.149	0.001584	0.0057168	4.54E-15	2.51E-05	3.30E-01
	Median	9571.9016	0.012829	263.8965	8.75E-16	1.34	1.96E+00
	Best	6308.9451	0.012674	263.8959	2.81E-18	1.34	1.59E+00
	Worst	24106.24	0.017696	263.9268	2.05E-14	1.3401	2.74E+00
	Runtime	4.67E-02	4.93E-02	4.26E-02	1.82E-02	3.35E-02	6.09E-02
AO	Avg	7.43E+03	2.55E-02	2.65E+02	9.44E-10	1.3491	1.94E+00
	Std	2.44E+03	7.75E-03	6.35E-01	2.12E-09	0.0061129	2.89E-01
	Median	6.90E+03	2.46E-02	2.64E+02	5.84E-11	1.3476	1.87E+00
	Best	6.05E+03	1.35E-02	2.64E+02	9.77E-14	1.34E+00	1.54E+00
	Worst	2.01E+04	5.14E-02	2.66E+02	1.04E-08	1.3675	2.6266
	Runtime	8.57E-02	8.46E-02	7.10E-02	2.03E-02	5.15E-02	1.07E-01
DOA	Avg	6.82E+03	7.88E+07	2.65E+02	8.74E-14	1.3806	1.7632
	Std	1.63E+03	2.17E+08	4.78E+00	3.22E-13	0.073821	0.44701
	Median	6.21E+03	1.29E-02	2.64E+02	0.00E+00	1.3436	1.57E+00
	Best	5880.6722	1.27E-02	263.8958	0.00E+00	1.34	1.47E+00
	Worst	1.30E+04	9.30E+08	2.83E+02	1.30E-12	1.6314	3.0434
	Runtime	5.98E-02	6.05E-02	5.45E-02	2.63E-02	4.49E-02	7.30E-02
HEOA	Avg	2.35E+04	1.44E+06	2.64E+02	3.34E-15	1.4225	2.895
	Std	1.15E+04	7.86E+06	5.31E-01	1.74E-14	0.040634	0.62066
	Median	2.16E+04	2.02E-02	2.64E+02	0.00E+00	1.4301	3.04E+00
	Best	9.16E+03	1.32E-02	2.64E+02	0.00E+00	1.3576	1.54E+00
	Worst	6.03E+04	4.31E+07	2.66E+02	9.52E-14	1.5045	4.5758
	Runtime	9.07E-02	1.04E-01	9.49E-02	7.18E-02	9.27E-02	1.17E-01
RSO	Avg	1.75E+04	4.88E+08	2.69E+02	4.83E-03	2.4988	69614393.6
	Std	14305.6529	3.85E+08	4.04E+00	7.92E-03	1.0756	166908921.2
	Median	12643.4507	571158159.3	267.0047	0.0011385	2.1394	3.86E+00
	Best	7610.8326	0.013022	264.6271	3.37E-06	1.3984	2.44E+00
	Worst	7.47E+04	9.30E+08	2.79E+02	3.38E-02	5.9221	782639036.2
	Runtime	3.69E-02	3.75E-02	3.10E-02	6.43E-03	2.30E-02	4.84E-02

Table 11: continue.

TSA	Avg	6.24E+03	1.31E-02	2.64E+02	1.40E-11	1.3416	1.4914
	Std	3.48E+02	4.41E-04	1.05E-02	3.99E-11	0.00069859	0.0063418
	Median	6.13E+03	1.29E-02	2.64E+02	3.02E-12	1.3416	1.4907
	Best	5.91E+03	1.27E-02	2.64E+02	5.35E-16	1.3405	1.48E+00
	Worst	7.20E+03	1.44E-02	2.64E+02	2.16E-10	1.343	1.5078
	Runtime	4.04E-02	3.97E-02	3.23E-02	9.56E-03	2.54E-02	5.14E-02
HOA	Avg	1.00E+04	1.43E-02	2.64E+02	8.54E-17	1.3614	2.8641
	Std	2.80E+03	1.37E-03	2.45E-01	3.68E-16	0.014747	0.38155
	Median	8.96E+03	1.39E-02	2.64E+02	3.58E-22	1.3568	2.7839
	Best	7.16E+03	1.28E-02	2.64E+02	7.40E-28	1.3428	2.0788
	Worst	1.77E+04	1.82E-02	2.65E+02	1.97E-15	1.4034	3.7521
	Runtime	5.95E-02	6.18E-02	5.29E-02	2.55E-02	4.22E-02	7.12E-02
GGO	Avg	15759.6451	305839959.7	264.7358	0.00010283	6.7139	3.5159
	Std	5059.5689	383148330.1	1.3389	0.00039273	1.324	1.2255
	Median	15480.0166	0.20425	264.1655	0	6.9455	3.4205
	Best	8914.8118	0.020712	263.8964	0	3.7432	2.0006
	Worst	30760.642	990844134	269.1114	0.0021473	8.7361	7.8879
	Runtime	3.82E-02	4.03E-02	3.39E-02	8.45E-03	2.40E-02	5.36E-02
POA	Avg	15931.7675	0.01321	264.7981	1.35E-09	6.2566	2.3585
	Std	3665.3995	6.95E-05	6.00E-01	2.12E-09	1.1823	0.39296
	Median	16298.1698	0.013214	264.6833	3.68E-10	6.2464	2.369
	Best	8538.1158	0.012984	263.9218	1.42E-13	3.75	1.5673
	Worst	22456.116	0.013329	266.2652	8.87E-09	9.4421	3.0874
	Runtime	3.27E-02	3.48E-02	3.01E-02	3.71E-03	1.82E-02	4.55E-02

5. Discussion

The improvements made to develop CLFOX made the algorithm perform better than the main FOX optimization algorithm and all other compared algorithms in this study. This is demonstrated by the CLFOX results for the benchmark function sets used to evaluate its performance. For the classical benchmark functions, CLFOX is better than FOX in 22 of the 23 functions. CLFOX outperforms FOX in 8 out of 10 functions in the CEC2019 set. CLFOX is better than FOX in all 12 functions in the CEC2022 set. Additionally, to verify the practical application of CLFOX, it was used to solve six real-world engineering problems and found solutions for all of them. In five out of six problems, CLFOX was better than FOX.

These results are broadly in line with recent studies that have improved the original FOX algorithm by adding more mechanisms to enhance its search efficiency. For example, OBL4FOX [19] improved FOX using opposition-based learning and nonlinear dynamic control variable strategies. IFOX [18] proposed an adaptive strategy to improve the balance between exploration and exploitation. Similarly, ASFFOX [20] used a chaotic map, adaptive inertia weight, Lévy flight and spiral search to improve the search process. Other studies have also shown the potential of hybridizing FOX with other optimization and learning strategies such as FOX-TSA [24] and FOX-GWO [29]. These studies confirm that the performance of FOX can be improved by modifying it. CLFOX uses a different improvement strategy that integrates a Cubic chaotic map, Lévy flight, extended main condition and interval-based operator into one framework. The steady improvement of CLFOX over FOX in all benchmark sets indicates that these mechanisms are able to cooperate well to improve population diversity, exploration, and the ability to escape local optima.

The four major improvements made in the original FOX are responsible for such strong results. For population initialization, the Cubic chaotic map is used, which improves the population diversity from the beginning of the search. A Lévy flight mechanism was added to the algorithm to explore the whole of the search area more widely since FOX has a bad exploration. Furthermore, the main condition of the algorithm was extended by three cases instead of two, which resulted in an enhanced balance between exploitation and exploration. Finally, an interval-based operator was added to avoid the algorithm getting stuck in local optima, and to facilitate its exit if it does.

While CLFOX had promising results in the benchmark tests as well as real-world problems, there is still room for enhancement. Although CLFOX is slightly weak regarding unimodal functions, it underperforms when it comes to several multimodal functions in the CEC2022 set, which shows that the

exploration ability has been greatly improved compared to FOX but can be further improved still. Future work should be aimed at strengthening the exploration mechanism to obtain more consistently optimal results.

6. Conclusions

In this study, a new improved version of the FOX optimization algorithm titled CLFOX is introduced. FOX is one of the powerful new optimization algorithms in the domain, developed to solve optimization problems. Although it has some limitations, this paper introduces another version to overcome them.

CLFOX uses four new strategies to improve FOX. Firstly, CLFOX uses a chaotic map to initialize the population instead of regular random population initialization, specifically the Cubic map. Secondly, a Lévy flight-based exploration strategy is added to the algorithm along with the original FOX exploration strategy. The third technique increases the main condition of the algorithm from two conditions to three. Last but not least, an interval-based operator is added to CLFOX. The interval runs once every 10 iterations and adds a new random agent to the population.

To show the performance of CLFOX and to determine its impact in the field of optimization, three sets of benchmark functions were used. The classical benchmark function set, the CEC2019 set, and the CEC2022 set were employed to assess CLFOX's performance. CLFOX was also compared to eight recent powerful optimization algorithms, including AO, DOA, HEOA, RSO, TSA, HOA, GGO, and POA. The results show that, on average, CLFOX was able to outperform all of the compared algorithms, which means CLFOX has a better result and impact.

CLFOX is used to solve six engineering problems, specifically PVD, TSD, TBD, GTD, CBD, and WBD. CLFOX was also in contrast to the other nine algorithms in resolving these problems, showing a better impact and reaching more optimal solutions, which shows that CLFOX techniques have a significant influence on improving the algorithm.

This high performance of CLFOX across different environments and test problems is due to the new techniques that have been added to the algorithm. Adding a cubic map to initialize the population leads the algorithm to have a diverse population in its initial steps. The Lévy flight technique improves exploration in the proposed version, stopping the algorithm from getting stuck in local optima. Additionally, the main condition of the algorithm has been increased from two to three: one condition focuses on exploitation, another on exploration, and the third also emphasizes exploration, preserving the balance between exploitation and exploration. The interval-based operator, which runs once every 10 iterations and adds a new random agent to the population, maintains the population's diversity along with the increasing iterations. It also assists in the algorithm's departure from local optima.

Despite the fact that there is a notable distinction between FOX and CLFOX, the proposed CLFOX still has certain drawbacks and could be further enhanced. It has significant shortcomings when it comes to handling certain situations, particularly when dealing with sets that contain a variety of function types. The algorithm's exploration could be further enhanced because CLFXO performs better than FOX in practically all multimodal functions, but still fails to obtain the most minimal value.

Author contributions: Sima Abdulla Salih: Conceptualization, Formal Analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – original draft. **Hardi Mohammed Mohammed:** Project administration, Resources, Supervision, Validation, Writing – review & editing. **Zrar Khalid Abdul:** Project administration, Supervision, Writing – review & editing.

Data availability: No data was used for the research described in this article.

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

Funding: The authors did not receive support from any organization for the conducting of the study.

References

[1] H. M. Mohammed and T. A. Rashid, "FOX: a fox-inspired optimization algorithm," *Applied Intelligence,* vol. 53, pp. 1030–1050, Apr. 2023, doi: 10.1007/s10489-022-03533-0.

[2] M. Khishe and M. R. Mosavi, "Chimp optimization algorithm," *Expert Systems with Applications*, vol. 149, art. no. 113338, Jul. 2020, doi: 10.1016/j.eswa.2020.113338.

[3] J. F. Salih, H. M. Mohammed, and Z. K. Abdul, "Modified fitness dependent optimizer for solving numerical optimization functions," *IEEE Access*, vol. 10, pp. 83916–83930, Aug. 2022, doi: 10.1109/ACCESS.2022.3197290.

[4] E. H. Houssein, M. K. Saeed, G. Hu, and M. M. Al-Sayed, "Metaheuristics for solving global and engineering optimization problems: review, applications, open issues and challenges," *Archives of Computational Methods in Engineering,* vol. 31, pp. 4485–4519, Aug. 2024, doi: 10.1007/s11831-024-10168-6.

[5] N. Aguirre, E. Cuevas, A. Luque-Chang, and H. Escobar-Cuevas, "An improved swarm optimization algorithm using exploration and evolutionary game theory for efficient exploitation," *The Journal of Supercomputing*, vol. 81, art.no. 574, Mar. 2025, doi: 10.1007/s11227-025-07007-1.

[6] N. Fang and Q. Cao, "Leaf in wind optimization: a new metaheuristic algorithm for solving optimization problems," *IEEE Access*, vol. 12, pp. 56291–56308, Apr. 2024, doi: 10.1109/ACCESS.2024.3390670.

[7] L. Mei and Q. Wang, "Structural optimization in civil engineering: a literature review," *Buildings*, vol. 11, no. 2, p. 66, Feb. 2021. doi: 10.3390/buildings11020066.

[8] P. Trojovský and M. Dehghani, "Pelican optimization algorithm: a novel nature-inspired algorithm for engineering applications," *Sensors*, vol. 22, no. 3, art. no. 855, Feb. 2022, doi: 10.3390/s22030855.

[9] Z. Liu *et al.*, "A new hybrid algorithm for vehicle routing optimization," *Sustainability*, vol. 15, no. 14, art. no. 10982, Jul. 2023, doi: 10.3390/su151410982.

[10] H. Li and N. Shi, "Application of genetic optimization algorithm in financial portfolio problem," *Computational Intelligence and Neuroscience*, vol. 2022, no. 1, art. no. 5246309, Jul. 2022, doi: 10.1155/2022/5246309.

[11] J. Lian *et al.*, "Parrot optimizer: algorithm and applications to medical problems," *Computers in Biology and Medicine*, vol. 172, art. no. 108064, Apr. 2024. doi: 10.1016/j.compbiomed.2024.108064.

[12] A. Moura and M. Pinho, "A scheduling optimization approach to reduce outpatient waiting times for specialists," *Healthcare*, vol. 13, no. 7, art. no. 749, Apr. 2025**,** doi: 10.3390/healthcare13070749.

[13] S. Deb, D. S. Abdelminaam, M. Said, and E. H. Houssein, "Recent methodology-based gradient-based optimizer for economic load dispatch problem," *IEEE Access*, vol. 9, pp. 44322–44338, Mar. 2021, doi: 10.1109/ACCESS.2021.3066329.

[14] N. A. Alsamarai and O. N. Uçan, "Improved performance and cost algorithm for scheduling IoT tasks in fog–cloud environment using gray wolf optimization algorithm," *Applied Sciences*, vol. 14, no. 4, art. no. 1670, Feb. 2024, doi: 10.3390/app14041670.

[15] K. N. Vhatkar and G. P. Bhole, "Particle swarm optimisation with grey wolf optimisation for optimal container resource allocation in cloud," *IET Networks*, vol. 9, no. 4, pp. 189–199, Jul. 2020, doi: 10.1049/iet-net.2019.0157.

[16] B. Abdollahzadeh and F. S. Gharehchopogh, "A multi-objective optimization algorithm for feature selection problems," *Engineering with Computers*, vol. 38, pp. 1845–1863, Aug. 2022, doi: 10.1007/s00366-021-01369-9.

[17] B. Gülsün and M. R. Aydin, "Optimizing a machine learning algorithm by a novel metaheuristic approach: A case study in forecasting," *Mathematics*, vol. 12, no. 24, art. no. 3921, Dec. 2024, doi: 10.3390/math12243921.

[18] M. A. Jumaah, Y. H. Ali, and T. A. Rashid, "An improved FOX optimization algorithm using adaptive exploration and exploitation for global optimization," *PLOS ONE*, vol. 20, no. 9, art. no. e0331965, Sep. 2025, doi: 10.1371/journal.pone.0331965.

[19] D. Wang, C. Gao, and B. Gao, "Improved fox optimization algorithm based on opposition-based learning mechanism and control variable for solving engineering optimization problems," *IAENG International Journal of Computer Science*, vol. 53, no. 6, pp. 2316–2327, Jun. 2026.

[20] Z. Zhang, X. Wang, and L. Cao, "FOX optimization algorithm based on adaptive spiral flight and multi-strategy fusion," *Biomimetics*, vol. 9, no. 9, art. no. 524, Sep. 2024, doi: 10.3390/biomimetics9090524.

[21] M. A. Jumaah, Y. H. Ali, and T. A. Rashid, ""Artificial liver classifier: a new alternative to conventional machine learning models" *Frontiers in Artificial Intelligence*, vol. 8, art. no. 1639720, Aug. 2025, doi: 10.3389/frai.2025.1639720.

[22] E. BAŞ, "Binary FOX optimization algorithm based U-shaped transfer functions for knapsack problems," *Proceedings of the 7th International Tokyo Conference on Innovative Studies of Contemporary Sciences*, Tokyo, Japan, 2023, pp. 168–177.

[23] R. Sharma *et al.*, "Comparative performance analysis of binary variants of FOX optimization algorithm with half-quadratic ensemble ranking method for thyroid cancer detection," *Scientific Reports*, vol. 13, no. 1, art. no. 21800, Dec. 2023, doi: 10.1038/s41598-023-46865-8.

[24] S. A. Aula and T. A. Rashid, "FOX-TSA hybrid algorithm: Advancing for superior predictive accuracy in tourism-driven multi-layer perceptron models," *Systems and Soft Computing*, vol. 6, art. no. 200178, Dec. 2024, doi: 10.1016/j.sasc.2024.200178.

[25] S. A. Aula and T. A. Rashid, "FOX-TSA: Navigating complex search spaces and superior performance in benchmark and real-world optimization problems," *Ain Shams Engineering Journal*, vol. 16, no. 1, art. no. 103185, Jan. 2025, doi: 10.1016/j.asej.2024.103185.

[26] M. A. Jumaah, Y. H. Ali, and T. A. Rashid, "Efficient Q-learning hyperparameter tuning using FOX optimization algorithm," *Results in Engineering*, vol. 25, art. no. 104341, Mar. 2025, doi: 10.1016/j.rineng.2025.104341.

[27] B. Ahmed, S. R. Naqvi, T. Akram, L. Peng, and F. Almarshad, "A hybrid deep learning-ViT model and a meta-heuristic feature selection algorithm for efficient remote sensing image classification," *International Journal of Computational Intelligence Systems*, vol. 18, no. 1, art. no. 72, Dec. 2025, doi: 10.1007/s44196-025-00838-z.

[28] A. Masbakhah, U. Sa'adah, and M. Muslikh, "Feature selection risk factors cervical cancer using hybrid methods random forest and FOX-inspired optimization algorithm," *CAUCHY: Jurnal Matematika Murni dan Aplikasi*, vol. 9, no. 2, pp. 352–367, Nov. 2024, doi: 10.18860/ca.v9i2.29582.

[29] A. K. Feda, M. Adegboye, O. R. Adegboye, E. B. Agyekum, W. Fendzi Mbasso, and S. Kamel, "S-shaped grey wolf optimizer-based FOX algorithm for feature selection," *Heliyon*, vol. 10, no. 2, art. no. e24192, Jan. 2024, doi: 10.1016/j.heliyon.2024.e24192.

[30] M. A. Jumaah, Y. H. Ali, T. A. Rashid, and S. Vimal, "FOXANN: A method for boosting neural network performance," *Journal of Soft Computing and Computer Applications*, vol. 1, no. 1, p. 2, Jun. 2024, doi: 10.70403/3008-1084.1001.

[31] A. Euldji, M. Laidi, M. Hentabli, A. Madani, and S. Hanini, "FOX-inspired optimizer with support vector regression modeling of water-based CaCO3-CuO-SiO2 trihybrid nanofluids thermal properties: comparative study," *Croatica Chemica Acta*, vol. 98, no. 1, pp. 15-26, Jan. 2025, doi: 10.5562/cca4137.

[32] B. Köse, B. Demirtürk, and Ş. Konca, "Two approaches for solving nonlinear equation systems: Newton Raphson and Red Fox," *Türk Doğa ve Fen Dergisi*, vol. 14, no. 2, pp. 1–10, Jun. 2025, doi: 10.46810/tdfd.1570735.

[33] M. U. Zafar and S. Gul, "Q-FOX: A reinforcement learning framework with FOX-inspired adaptive hyperparameter optimization," *Proceedings of the 2025 IEEE 22nd International Conference on Smart Communities: Improving Quality of Life Using AI, Robotics and IoT (HONET)*, Lahore, Pakistan, 2025, pp. 110–115, doi: 10.1109/HONET67928.2025.11318528.

[34] R. Sharma *et al.*, "A framework for detecting thyroid cancer from ultrasound and histopathological images using deep learning, meta-heuristics, and MCDM algorithms," *Journal of Imaging*, vol. 9, no. 9, art. no. 173, Sep. 2023, doi: 10.3390/jimaging9090173.

[35] R. Mousa, S. Chamani, M. Morsali, M. Kazzazi, P. Hatami, and S. Sarabi, "Enhancing skin cancer diagnosis using late discrete wavelet transform and new swarm-based optimizers," *Machine Learning with Applications*, vol. 23, art. no. 100811, Mar. 2026, doi: 10.1016/j.mlwa.2025.100811.

[36] L. Xu and M. Mohammadi, "Brain tumor diagnosis from MRI based on Mobilenetv2 optimized by contracted fox optimization algorithm," *Heliyon*, vol. 10, no. 1, art. no. e23866, Jan. 2024, doi: 10.1016/j.heliyon.2023.e23866.

[37] B. Ocaña-Tienda *et al.*, "A comprehensive dataset of annotated brain metastasis MR images with clinical and radiomic data," *Scientific Data*, vol. 10, no. 1, art. no. 208, Dec. 2023, doi: 10.1038/s41597-023-02123-0.

[38] S. Thakur, R. K. Barai, A. Bhattacharya, and A. Ghosh, "Comparative analysis of sliding mode controllers for trajectory tracking and vibration reduction in a two-link flexible manipulator with different sliding surfaces," *Arabian Journal for Science and Engineering*, vol. 49, no. 12, pp. 16711–16728, Dec. 2024, doi: 10.1007/s13369-024-09196-y.

[39] P. Zhang, A. Daraz, S. A. Malik, C. Sun, A. Basit, and G. Zhang, "Multi-resolution based PID controller for frequency regulation of a hybrid power system with multiple interconnected systems," *Frontiers in Energy Research*, vol. 10, art. no. 1109063, Jan. 2023, doi: 10.3389/fenrg.2022.1109063.

[40] O. Tunca and S. Carbas, "Design cost minimization of a reinforced concrete column section using overnew swarm-based optimization algorithms," *Neural Computing and Applications*, vol. 36, no. 27, pp. 16941–16958, Sep. 2024, doi: 10.1007/s00521-024-09998-z.

[41] M. A. Ebrahim, E. E. Ahmed, M. M. Salama, and M. M. R. Ahmed, "Multi-objective optimization for optimum distributed generation integration in distribution networks," *Proceedings of the International Middle East Power Systems Conference (MEPCON)*, Cairo, Egypt, 2024, pp. 1-8, doi: 10.1109/MEPCON63025.2024.10850254.

[42] R. K. Hamad and T. A. Rashid, "A Systematic Study of Krill Herd and FOX Algorithms," in *Proceedings of the 1st International Conference on Innovation in Information Technology and Business (ICIITB 2022)*, Atlantis Press International BV, 2023, pp. 168–186. doi: 10.2991/978-94-6463-110-4_13.

[43] D. P. Nikezić, D. S. Radivojević, N. S. Mirkov, I. M. Lazović, and T. A. Miljojčić, ""Symmetric U-Net model tuned by FOX metaheuristic algorithm for global prediction of high aerosol concentrations," *Symmetry.*, vol. 16, no. 5, art. no. 525, May 2024, doi: 10.3390/sym16050525.

[44] B. Wu, S. Fan, "Improved artificial bee colony algorithm with chaos," in *Proceedings of the International Conference on Computer Science and Service System (CSSS 2011)*, *Communications in Computer and Information Science*, vol. 210, Nanjing, China, 2011, pp. 51–56, doi: 10.1007/978-3-642-22694-6_8.

[45] H. M. Mohammed and T. A. Rashid, "Chaotic fitness-dependent optimizer for planning and engineering design," *Soft Computing.*, vol. 25, pp. 14281–14295, Nov. 2021, doi: 10.1007/s00500-021-06135-z.

[46] S. Gopi and P. Mohapatra, "Chaotic aquila optimization algorithm for solving global optimization and engineering problems," *Alexandria Engineering Journal*, vol. 108, pp. 135–157, Dec. 2024, doi: 10.1016/j.aej.2024.07.058.

[47] Y. Fu, D. Liu, S. Fu, J. Chen, and L. He, "Enhanced aquila optimizer based on tent chaotic mapping and new rules," *Scientific Reports*, vol. 14, art. no. 3013, Dec. 2024, doi: 10.1038/s41598-024-53064-6.

[48] A. Abdollahpour, A. Rouhi, and E. Pira, "An improved gazelle optimization algorithm using dynamic opposition-based learning and chaotic mapping combination for solving optimization problems," *The Journal of Supercomputing Supercomputing.*, vol. 80, no. 9, pp. 12813–12843, 2024, doi: 10.1007/s11227-024-05930-3.

[49] Y. Li, X. Lin, and J. Liu, "An improved gray wolf optimization algorithm to solve engineering problems," *Sustainability (Switzerland)*, vol. 13, no. 6, art. no. 3208, Mar. 2021, doi: 10.3390/su13063208.

[50] J. Cai, X. Ma, L. Li, Y. Yang, H. Peng, and X. Wang, "Chaotic ant swarm optimization to economic dispatch," *Electric Power Systems Research*, vol. 77, no. 10, pp. 1373–1380, Aug. 2007, doi: 10.1016/j.epsr.2006.10.006.

[51] N. Dong, X. Fang, and A. G. Wu, "A novel chaotic particle swarm optimization algorithm for parking space guidance," *Mathematical Problems in Engineering*, vol. 2016, 2016, pp. 1-14, doi: 10.1155/2016/5126808.

[52] G. I. Sayed, G. Khoriba, and M. H. Haggag, "A novel chaotic salp swarm algorithm for global optimization and feature selection," *Applied Intelligence*, vol. 48, no. 10, pp. 3462–3481, Oct. 2018, doi: 10.1007/s10489-018-1158-6.

[53] A. H. Gandomi, X. S. Yang, S. Talatahari, and A. H. Alavi, "Firefly algorithm with chaos," *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, no. 1, pp. 89–98, Jan. 2013, doi: 10.1016/j.cnsns.2012.06.009.

[54] Z. Qi, D. Yingjie, Y. Shan, L. Xu, H. Dongcheng, and X. Guoqi, "An improved coati optimization algorithm with multiple strategies for engineering design optimization problems," *Scientific Reports*, vol. 14, art. no. 20435, Dec. 2024, doi: 10.1038/s41598-024-70575-4.

[55] J. Du, J. Zhang, S. Li, and Z. Yang, "Enhanced artificial hummingbird algorithm with chaotic traversal flight," *Scientific Reports*, vol. 14, p. 25892, Dec. 2024, doi: 10.1038/s41598-024-77115-0.

[56] N. El Ghouate *et al.*, "Improving the kepler optimization algorithm with chaotic maps: comprehensive performance evaluation and engineering applications," *Artificial Intelligence Review*, vol. 57, no. 11, art. no. 313, Nov. 2024, doi: 10.1007/s10462-024-10857-5.

[57] M. Chawla and M. Duhan, "Levy flights in metaheuristics optimization algorithms–a review," *Applied Artificial Intelligence*, vol. 32, no. 9–10, pp. 802–821, Nov. 2018, doi: 10.1080/08839514.2018.1508807.

[58] B. Bhatt, H. Sharma, K. Arora, G. P. Joshi, and B. Shrestha, "Levy flight-based improved Grey Wolf optimization: A solution for various engineering problems," *Mathematics*, vol. 11, no. 7, art. no. 1745, Apr. 2023, doi: 10.3390/math11071745.

[59] Z. Yan, J. Yan, Y. Wu, and C. Zhang, "An improved hybrid mayfly algorithm for global optimization," *The Journal of Supercomputing*, vol. 79, no. 6, pp. 5878–5919, Apr. 2023, doi: 10.1007/s11227-022-04883-9.

[60] P. He and W. Wu, "Levy flight-improved grey wolf optimizer algorithm-based support vector regression model for dam deformation prediction," *Frontiers in Earth Science*, vol. 11, art. no. 1122937, Jan. 2023, doi: 10.3389/feart.2023.1122937.

[61] Y. Liu and B. Cao, "A novel ant colony optimization algorithm with Levy flight," *IEEE Access*, vol. 8, pp. 67205–67213, Apr. 2020, doi: 10.1109/ACCESS.2020.2985498.

[62] J. Liu, J. Shi, F. Hao, and M. Dai, "A novel enhanced global exploration whale optimization algorithm based on Lévy flights and judgment mechanism for global continuous optimization problems," *Engineering with Computers*, vol. 39, no. 4, pp. 2433–2461, Mar. 2022, doi: 10.1007/s00366-022-01638-1.

[63] T. Lou, G. Guan, Z. Yue, Y. Wang, R. Qi, and S. Tong, "A competitive game optimization algorithm for unmanned aerial vehicle path planning," *Cluster Computing*, vol. 28, no. 9, art. no. 573, Aug. 2025, doi: 10.1007/s10586-025-05186-3.

[64] L. Abualigah, M. A. Elaziz, D. Yousri, M. A. A. Al-qaness, A. A. Ewees, and R. A. Zitar, "Augmented arithmetic optimization algorithm using opposite-based learning and lévy flight distribution for global optimization and data clustering," *Journal of Intelligent Manufacturing*, vol. 34, no. 8, pp. 3523–3561, Sep. 2022, doi: 10.1007/s10845-022-02016-w.

[65] L. Abualigah, D. Yousri, M. Abd Elaziz, A. A. Ewees, M. A. A. Al-qaness, and A. H. Gandomi, "Aquila optimizer: a novel meta-heuristic optimization algorithm," *Computers & Industrial Engineering*, vol. 157, art. no. 107250, Jul. 2021, doi: 10.1016/j.cie.2021.107250.

[66] H. Peraza-Vázquez, A. F. Peña-Delgado, G. Echavarría-Castillo, A. B. Morales-Cepeda, J. Velasco-Álvarez, and F. Ruiz-Perez, "A bio-inspired method for engineering design optimization inspired by dingoes hunting strategies," *Mathematical Problems in Engineering*, vol. 2021, pp. 1-19, 2021, doi: 10.1155/2021/9107547.

[67] J. Lian and G. Hui, "Human evolutionary optimization algorithm," *Expert Systems with Applications*, vol. 241, art. no. 122638, May 2024, doi: 10.1016/j.eswa.2023.122638.

[68] G. Dhiman, M. Garg, A. Nagar, V. Kumar, and M. Dehghani, "A novel algorithm for global optimization: rat swarm optimizer," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 8, pp. 8457–8482, Aug. 2021, doi: 10.1007/s12652-020-02580-0.

[69] S. Kaur, L. K. Awasthi, A. L. Sangal, and G. Dhiman, "Tunicate swarm algorithm: a new bio-inspired based metaheuristic paradigm for global optimization," *Engineering Applications of Artificial Intelligence*, vol. 90, art. no. 103541, Apr. 2020, doi: 10.1016/j.engappai.2020.103541.

[70] S. O. Oladejo, S. O. Ekwe, and S. Mirjalili, "The hiking optimization algorithm: a novel human-based metaheuristic approach," *Knowledge-Based Systems*, vol. 296, art. no. 111880, Jul. 2024, doi: 10.1016/j.knosys.2024.111880.

[71] E. S. M. El-kenawy, N. Khodadadi, S. Mirjalili, A. A. Abdelhamid, M. M. Eid, and A. Ibrahim, "Greylag goose optimization: nature-inspired optimization algorithm," *Expert Systems with Applications*, vol. 238, p. 122147, Mar. 2024, doi: 10.1016/J.ESWA.2023.122147.

[72] O. Al-Baik *et al.*, "Pufferfish optimization algorithm: a new bio-inspired metaheuristic algorithm for solving optimization problems," *Biomimetics*, vol. 9, no. 2, art. no. 65, Jan, 2024, doi: 10.3390/biomimetics9020065.