

Research Article:

Improved Chebyshev Distance FOX Algorithm for Solving Numerical and Engineering Optimization Problems

Musa A. Muhammad^{1*}, Hardi M. Mohammed², Zrar Kh. Abdul³

¹Computer Science Department, College of Science, Charmo University, Sulaimani, Chamchamal 46023, KRG, Iraq;

²Department of Software Engineering, College of Engineering and Computational Science, Charmo University, Chamchamal 46023, Sulaimani-Kurdistan Region, Iraq.

³Department of Software Engineering, College of Engineering and Computational Science, Charmo University, Chamchamal 46023, Sulaimani-Kurdistan Region—Iraq.

<https://doi.org/10.31530/cjnst.2026.2.1.8>

Received: 15 February 2026

Revised: 17 April 2026

Accepted: 22 April 2026

***Corresponding author info:** Musa A. Muhammad,
Email: musa.ayub@chu.edu.iq

Copyright © 2026 Musa A. Muhammad, Hardi M. Mohammed, Zrar Kh. Abdul, Charmo Journal of Natural Sciences (CJNST).

This is an open access article published under the terms of the Creative Commons Attribution License (CC BY 4.0), which permits use, distribution, and reproduction in any medium, provided the original work is properly cited.



Abstract

Background: FOX optimizer, inspired by the smart hunting behavior of red foxes, is a powerful optimization algorithm used to solve difficult engineering and real-world problems. However, constrained exploration often affects the conventional FOX algorithm, leading to premature convergence and agent accumulation at local optima.

Aim: The aim of this research is to improve the performance of the FOX optimization algorithm by optimizing the exploration and exploitation processes of the FOX algorithm. The proposed changes aim to improve the global search process in the initial phase of the search process and ensure a good local search process in the later phase of the search process. The purpose of this research is to improve the balance between exploration and exploitation and to reduce the risk of premature convergence and entrapment in local optima.

Methodology: For this, the Chebyshev Distance FOX algorithm (CDFOX) is proposed, which effectively facilitates exploration and exploitation through a well-structured, multi-step process. First, Chebyshev distance is used to calculate how far two points are by taking the largest absolute difference in any single dimension. Second, using a chaos map in both time and exploration improves the algorithm by adding controlled randomness. This helps the search agents avoid getting stuck, explore the search space more widely, and reach better solutions faster. Finally, adaptive scaling of the search performance expands the search radius to preserve diversity and enable wider, circular exploration of the search space.

Result: CDFOX was tested rigorously for its performance against 10 established and competitive optimizers with 63 benchmarks, comprising 23 traditional test problems, 10 benchmarks from CEC2019, and 30 benchmarks from CEC2017. From these results, it was confirmed that CDFOX performed better than or equally as well as other optimizers for 82% of classical benchmarks, 80% of CEC2019 benchmarks, and 43% of CEC2017 benchmarks. Moreover, results from real-world engineering problems like pressure vessel design, gear train design, and cantilever beam problems further support its effectiveness and its efficiency. and merits in addressing hard problems in various domains.

Conclusion: CDFOX is an improvement over the original FOX optimizer in the sense that the exploration and exploitation of the search space are enhanced through a multi-strategy approach. The adaptive nature of the CDFOX helps in the avoidance of premature convergence and enhances the global searchability. It is evident from the results that CDFOX is a powerful optimizer for solving complex optimization problems.

Keywords: CDFOX algorithm, metaheuristic, exploration, exploitation, optimization.

1. Introduction

Finding the optimal solution to a problem by identifying variable values that, when based on constraints, minimize or maximize an objective function is known as optimization [1]. Optimization in mathematics helps people find the optimal solution to make rational use of human, material, and financial resources [2].

Optimization problems are solved by using different types of optimization algorithms. Deterministic and stochastic algorithms are the two fundamental categories into which optimization problem-solving strategies are typically divided [3]. Deterministic approaches are well-suited for problems that are linear, convex, low-dimensional, continuous, and differentiable [4]. These algorithms follow a fixed set of steps and produce the same result for a given input. However, despite their effectiveness in certain situations, deterministic algorithms have several limitations. They often lack sufficient exploration capabilities, making them prone to getting trapped in local optima. Additionally, they struggle with solving complex, irregular, or high-dimensional problems, and their performance is highly sensitive to initial conditions [5]. Well-known examples of deterministic algorithms are binary search [8], merge sort [6], and Linear Search [7].

On the other hand, a number of stochastic algorithms have been developed in recent times as powerful alternatives to solve difficult optimization problems. These methods make use of randomness to explore the solution space and thereby allow a better balance between exploration and exploitation. Generally speaking, researchers are increasingly shifting towards the use of stochastic approaches to tackle a broad range of complex high-dimensional, non-convex, non-linear, discontinuous, and non-differentiable problems in science, engineering, industry, and real-world applications. The main reasons come from the fact that deterministic approaches are inefficient to solve such problems [12–13]. Stochastic algorithms provide several advantages regarding their stronger global search capabilities, reduced sensitivity to initial conditions, greater flexibility, and adaptability. Indeed, these features make it suitable for solving dynamic, noisy, and nonlinear optimization problems. Modern stochastic optimization was born in the 1970s when John Holland introduced the Genetic Algorithm (GA) [10], followed by Particle Swarm Optimization [11] and Differential Evolution (DE) [12]. After these pioneering works, the area has gone through considerable growth, with the proposal of numerous metaheuristic algorithms based on inspiration from natural and biological processes.

The metaheuristic algorithms are probabilistic and have the goal of searching for the best solution for the optimization problem. In other words, the term "metaheuristics" refers to a class of intelligent strategies for the purpose of heuristic solution efficiency improvement [16]. The algorithm inspirations are derived from different sources: biology principles [13], animal behavior [14], human strategy [15], and natural phenomena [16]. Metaheuristic algorithms can be divided into four main groups based on their source of inspiration: swarm intelligence-based, human-based, physics-based, and evolutionary-based.

Swarm Intelligence Algorithms (SIAs) are modeled after the behavior of swarms in nature, such as the foraging behavior of some animals [17]. These techniques mimic interactions between individuals or groups that help for doing a strong and effective search to find the best solution in the search space [18]. The popular examples for SIAs are PSO [19], which was inspired by how birds or fish move in the group to find food, and Ant Colony Optimization (ACO), which mimics finding the shortest route to find food.

A human-based metaheuristic algorithm that is inspired by the behavior of humans, such as learning and social interaction, and the relationships between humans [20]. It is not old compared to other algorithms, but in recent years they have grown. Teaching-Learning-Based Optimization (TLBO) [21] is based on the teaching and learning technique in the classroom. Socio-inspired evolutionary algorithms (SIEA) [22] are based on human social learning and evolution. Physics-based algorithms have been inspired by process and law and concepts of physics [23]. Thermal Exchange Optimization (TEO) [24] is based on the process of heat transfer between objects in the thermal system. The water cycle algorithm (WCA) [25] is based on the natural water cycle process, especially the flow of the river and stream. Evolutionary-based algorithms are inspired by the evolution of biology: natural selection, mutation, and reproduction.

Metaheuristic algorithms have been applied in several domains, such as engineering design problems [26], medical diagnostics and treatment planning [27], optimization problems with constraints [28], economic models [29], and even in problem-solving tasks such as scheduling. However, metaheuristic algorithms encounter challenges when addressing certain issues, among which is their inability to explore well and their tendency to get trapped in local optimum solutions. These limitations reflect an inadequate ability to search a diverse set of solutions in complex, non-convex optimization problems [34–36]. As a result, algorithms may suffer from premature convergence and an ineffective balance between exploration and exploitation.

The FOX-inspired optimization algorithm (FOX) is a recent intelligent optimization algorithm developed by Hardi and Tarik in 2022. FOX is classified as a biologically inspired metaheuristic algorithm. It performs very well in certain engineering optimization problems and has the advantage of having a simple structure and fast convergence speed [37–39]. Foxes exhibit a unique balance between exploration and exploitation in their search for food and for carefully scanning their environment, adapting for changing conditions, and collaborating with others. The FOX algorithm has some limitations that make it less effective for complex optimization problems. These include converging, being very sensitive to parameter settings, being less efficient in high-dimensional search spaces, not being able to explore well, and getting stuck in local optima. These problems show that the FOX optimizer needs to improve its exploration mechanism and overall search balance [11][40].

In this study, researchers improve the exploration problem of FOX. Thus, to overcome the exploration problem of the FOX optimizer, different techniques are used inside the FOX optimizer. The main contributions of this study are:

- Modifying the original FOX distance measure by the Chebyshev distance measure for finding distance,
- Introducing chaos map in the time and exploration phase for improving the exploration phase.
- Evaluating proposed CDFOX using three benchmark functions: Classical, CEC2017, and CEC2019,
- Validating results using statistical measurement to determine the significant of results,
- Solving three real-world problems, such as pressure vessel design, gear train design, and cantilever beam design.

The remainder of this article is structured as follows: In the second section, overall, is the mathematical model of the original FOX. The third section is a literature review on algorithms that have limitations in exploration and get stuck in local optima and balance between exploration and exploitation. In the fourth section, methodology talks about the improvement of the Fox algorithm and explains the rule that was modified. In the fifth section, the results of CDFOX against FOX and other algorithms are evaluated, and discussion will show and analyze the results. The final section has a conclusion.

2. Related work

Metaheuristic algorithms can be developed by researchers. These proposed algorithms then would be modified or hybridized. Thus, FOX is one of the algorithms that has been proposed by its authors. This section focuses on all papers that used FOX as a modification and hybridization, and it is used for solving applications:

In the paper[36], the FOX algorithm was modified. Network optimization, image processing, and engineering have all effectively employed swarm intelligence algorithms to solve difficult optimization problems. Among these, the FOX optimization algorithm—which was initially presented in 2022 and was inspired by the fox hunting method—has garnered a lot of interest due to its effectiveness and ease of use. However, the FOX method may have limited population diversity, poor global exploration, and a high likelihood of reaching local optima when confronted with challenging optimization issues. Several enhanced algorithms have been proposed to address the aforementioned problem. For instance, adaptive inertia weight can balance exploration and exploitation in the optimization process, and chaotic tent maps can enhance population variety and exploration in the early phases. Spiral search strategies can broaden the search process, while Levy flight strategies can help the algorithm in escaping local optima during long-distance flights. While some of the FOX algorithm problems have been resolved by these algorithms, most of them focus on particular aspects and may make algorithmic complexity or reliability in performance across various problems challenging. This study offers a novel method called "Adaptive Spiral Flight and Multi-Strategy Fusion FOX" (ASFFOX) that may successfully increase the variety, exploration capacity, convergence rate, and robustness in addition to addressing the aforementioned problems. Additionally, the suggested algorithm's effectiveness in resolving CEC2017 test problems may be confirmed in two engineering application scenarios and contrasted with seven traditional

swarm intelligence techniques. The suggested approach can attain greater computational stability, a quicker rate of convergence, and greater precision, according to the experimental results. Research will be done in the future to increase optimization accuracy in low-dimensional issues and flexibility in many real-world scenarios. A summary of some representative FOX enhancements and their applications is given in Table 1.

The mFOX (modified FOX) algorithm, which was modified in [20] and aims to offer a better balance between exploration and exploitation, is one such enhancement. By adding opposition-based learning (OBL), the technique was integrated into mFOX to increase variety in the initial population and increase the likelihood of finding good regions early in the search. Additionally, by removing the minimum time part and changing the range of its decreasing values from 1 to 0, the control parameter was enhanced, allowing for more exploration during search iterations to provide a larger search space and prevent local-optima problems. The position update was also enhanced using the relative positions of search agents rather than just the best answer. Four real-world engineering problems and 45 benchmark functions were used to assess mFOX performance. The results showed that mFOX typically beats a number of well-known optimization strategies. While there are many benefits to mFox and related algorithms, there are also certain drawbacks. For instance, even while OBL increases population variety, it may not necessarily guarantee better search space exploration in high-dimensional situations and may even increase computational costs. Furthermore, a balanced exploration and exploitation of the search space in various optimization problems may not always be guaranteed by parameter adjustment procedures, which may require careful tuning. All of these challenges need the development of more effective and simple techniques for enhancing exploration and exploitation. Table 1 summarizes a few typical FOX improvements and their uses.

In paper [37], the FOX optimization algorithm has been updated recently with a focus on enhancing its exploration and exploitation abilities and increasing its scope of use across various problem domains. The initial updates to the algorithm focused on introducing fixed changes in the exploration and exploitation ratios, as well as fixed jump equations or distance calculations, to achieve faster convergence and precise search results when compared to other optimization techniques such as PSO, GWO, and GA. Nevertheless, the optimization technique has become less flexible in various complicated problem areas as a result of the algorithm's fixed modifications. The accuracy and convergence characteristics for targeting prey were further enhanced by later improvements, adding physical-inspired components like ultrasound-based distance estimation. However, these formulations were fixed, which decreased dynamism. Hybridizing FOX with other nature-inspired algorithms, like Differential Evolution or Grey Wolf Optimization, was another advancement that improved FOX's efficiency by utilizing the best aspects of several algorithms. However, this also increased parameterization and computational complexity, which would not be advantageous for scalability, particularly for large dimensions and/or confined regions.

These improvements developed upon FOX and enhanced it in various ways, yet they once again lacked robustness, generality, and dynamism. Thus, this study suggests Improved FOX (IFOX), which has a fully adaptive feature that simplifies FOX and enhances its basic formulas in addition to dynamically adjusting exploration and exploitation. In both benchmark problems and real-world applications, IFOX has demonstrated its effectiveness in handling various optimization issues; in fact, it has outperformed the original FOX algorithm by 40%. Furthermore, as Table 1 summarizes, IFOX has outperformed other cutting-edge metaheuristics.

The FOX algorithm was hybridized in [38] and has been widely used for feature selection tasks in addition to other metaheuristic algorithms such as the genetic algorithm, Aquila optimizer, whale optimization algorithm, and gray wolf optimizer. The FOX algorithm may still have problems with poor exploitation and premature convergence in complicated and high-dimensional domains, despite its straightforward structure and fast convergence rate. The FOX algorithm's diversity, convergence rate, and exploration-exploitation tradeoff are all improved by the use of chaotic maps, hybridization approaches, and transfer functions, among other modifications suggested in the literature. In this case, transfer functions are used to perform effective binary conversion for feature selection issues, while the leadership structure in Grey Wolf Optimizer enhances exploitation. Nevertheless, the majority of these algorithms continue to struggle with handling the restrictions and increased computational complexity of binary mapping approaches. This study suggests a unique S-shaped FOX-GWO algorithm that combines Grey Wolf Optimizer operators and improves binary mapping techniques to solve the difficulties encountered by current algorithms in solving optimization problems and classification problems. According to the experimental results, the suggested FOX-GWO algorithm reduces the number of selected features for 61.11% of the datasets, produces better optimization outcomes for

72.22% of optimization issues, and performs better in classification accuracy for 83.33% of the datasets. This algorithm's great accuracy and robustness in handling high-dimensional data make it effective in addressing optimization and classification issues, even if it could take longer to process. Table 1 summarizes some of the most recent developments in FOX and its uses.

The FOX algorithm has been used in tuning machine learning algorithm parameters: Deep learning models have been used for aerosol prediction using satellite photos in reference [39], because they are good at detecting the spatial patterns in the images. Aerosol concentration levels in the atmosphere were predicted using the symmetric U-Net architecture using a set of photos from previous satellite measurements. ConvLSTM2D layers were employed in the design to address the weaknesses of conventional convolutional models in capturing the temporal dependencies in the aerosol data. This allows for the simultaneous learning of spatial and temporal dependencies in the data. The introduction of ConvLSTM2D layers enhanced the models' complexity, making their performance heavily reliant on the choice of hyperparameter values, such as learning rate and dropout, which have a major impact on the models' stability. Because the search space may not be efficiently explored, the conventional techniques of fine-tuning the hyperparameter values rely on trial and error, which takes time. The Fox Optimization algorithm was used for effective tuning. The goal of FOX is to efficiently explore the parameter space's promising areas while avoiding the poor random search that typically occurs during the conventional tuning technique. The FOX tuning technique and the symmetric U-Net architecture make the tuning methodical and dependable, increasing accuracy of predictions provided using the optimized models. Table 1 summarizes the experimental results, which show that the optimized model outperformed the conventional models, including the ResNet3D-101, with an AUC-PR value of 0.572.

Table 1: Summary of Related Work on FOX Optimization Algorithm Modifications

Ref	Modification	Purpose	Application	Limitation	Advantage
[36]	Chaotic FOX	Improve diversity.	1. Benchmark functions 2. Feature selection 3. Engineering optimization 4. Continuous optimization	Limited late-stage improvement	Enhanced diversity, faster convergence, better local optima avoidance
	Adaptive FOX	Balance exploration/exploitation		Parameter sensitivity	
	Levy Fox	Escape local optima.		Convergence instability	
	Spiral FOX	Diversify the search path.		Higher complexity	
[38]	Hybrid FOX–GWO	Combine FOX exploration with GWO exploitation.	Feature selection and optimization tasks	Stronger exploitation and improved convergence accuracy	Increased computational complexity

[39]	Integration of ConvLSTM2D layers with a symmetric U-Net architecture and FOX-based hyperparameter optimization	To capture both spatial and temporal aerosol patterns and automatically optimize model hyperparameters	Aerosol concentration prediction from satellite imagery	Increased model complexity and high sensitivity to hyperparameter settings	FOX enables systematic hyperparameter tuning and improves prediction accuracy and model stability, achieving better performance (AUC-PR = 0.572) compared to baseline models.
[18]	Modified FOX (mFOX)	Achieve better balance between exploration and exploitation.	Benchmark functions (45) and real-world engineering problems (4)	OBL may increase computational effort; parameter adjustment may need careful tuning and not always maintain balance in high-dimensional problems.	Improves convergence and solution quality, maintains computational efficiency, reduces risk of local optima, and expands search space through relative position updates.
[37]	IFOX – Adaptive exploration-exploitation	Automatically adjusts the balance between exploration and exploitation based on the current solution quality	Classical benchmark function, CEC2019, CEC2021, CEC2022, and engineering problems.	May require initial tuning of control parameters for optimal performance.	Robust performance, high adaptability, efficient convergence across diverse

3. Methodology

3.1. FOX Optimization Algorithm

The FOX algorithm emulates the hunting strategy of a red fox while digging into the snow with the only motive of catching its prey. Foxes hunt their prey to survive with very brilliant strategies [32 - 33]. The position of the red fox is represented as $X_{initial}$. With every iteration, the utilization of the standard objective function makes sure that the fitness level of the search agent is identified. In addition to this, the cost of every searching agent is compared with others. Additionally find out the best fitness along with the best position expressed as $Best X$. The current row's fitness, expressed as $fitness_i + 1$, and the previous row's fitness, also expressed as fitness, are compared for iterations [41].

To ensure that both phases of exploration and exploitation are balanced, there is an inventory of random variables to cater to equal iterations of both phases. A variable r is responsible for affording a 50% probability of occurring to both phases of FOX; hence, equivalent iterations of both

phases are 50%. The above-mentioned algorithm is responsible for balancing, as it prevents the gathering of results in terms of local optima. A variable α is responsible for lessening search performance on the best position ($BestX$) after an iteration, hence improving the chase of an agent. The fitness value and position of search agents, along with exploring, disengage if both possess equivalent positions. The following sections describe both phases of exploitation and exploration.

3.1.1. Exploitation

In this stage, the prey could die because of an existing condition. The random variable named p varies from 0 to 1. When p indicates a value over 0.18, a new location for the red fox has to be discovered. To do this, the values for distance from fox to prey ($Dist_{Fox_Prey_{it}}$), distance sound travels ($Dist_{S_T_{it}}$), jump ($Jump_{it}$), and number of iterations (it) need to be calculated. Lastly, a random number for sound travel time $Time_{S_T_{it}}$ is generated between 0 and 1, which can be multiplied by the speed of sound in air Sp_{Sto} find the sound distance, as can be found in Eq. 1 [33].

$$Dist_S_T_it = Sp_S \times Times_S_T_it \quad (1)$$

The formula's distance sound travels $Dist_S_T_it$ and iteration count (it) indicate how far sound travels. The number of iterations as of right now is the parameter. The speed of sound is denoted by SP_S . The propagation time of sound is represented by $Time_s_T_it$ a random value between 0 and 1. As indicated by Eq. 2, the value of (Sp_S) may be

calculated using the time it takes for sound to reach the fox and prey as well as the optimal location at that precise instant [33].

$$SP_S = \frac{BestPosition_{it}}{Time_s_T_it} \quad (2)$$

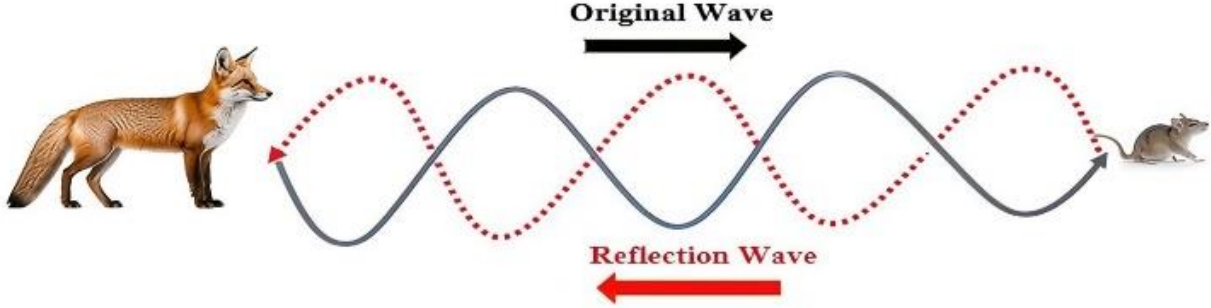


Figure 1: Calculating distance by using ultrasound

The distance that sound travels is determined using Eq. 1. The distance between the fox and the target $Dist_Fox_Prey_it$ may be calculated if it $Dist_S_T_it$ is cut in half. Figure 1 illustrates how a sensor transmits a sound wave to an item and subsequently gets a signal back. The sensor multiplies the sound travel distance by half. Using either option does not change the outcome since half of the signal is needed. In more detail, as described in Eq. 3 [33].

$$Distance_Fox_Prey_{it} = Distance_S_T_{it} \times 0.5 \quad (3)$$

Once the distance between the red fox and the prey has been determined, it is time to identify a new spot where the fox will need to jump to reach the objective. In other words, the red fox will now calculate the amount of jumping needed with a known distance that is referred to as $Jump_it$. Additionally, the following equation will calculate $Jump_it$.

$$Jump_{it} = 0.5 * 9.81 \times t^2 \quad (4)$$

The average time taken for sound wave propagation and jumping movement of foxes on snow are computed through the use of the parameter t . From the experiment conducted, it is clear that the proposed method improves convergence speed and solution precision through the control of the step size and movement of search agents. This value is squared since there would be steps up and down during jumps. This is denoted as 9.81 and shows the acceleration due to gravity. To obtain the transition tt , the sum of the $Time_S_T_{it}$ divided by dimensions is used. The computation of $MinT$ and tt is provided by Eq. 3. To obtain the average time t , tt is divided by 2. Since the jump value changes twice, both gravity and meantime involve multiplying by 0.5. Then, the jump value involves the multiplication by $c1$, $Dist_Fox_Prey_it$. Every time the fox jumps northeast, the range for $c1$ changes from $[0, 0.18]$. The new

location of the fox is given by the equation below when $p > 0.18$ (Eq. 5 [33]).

$$X_{it+1} = Distance_Fox_Prey_{it} * Jump_{it} \times C1 \quad (5)$$

In searching for a new position for the fox, Eq. 5 and Eq. 6 are applied. Because p only one equation at a time is solved. If ' $p \leq 0.18$,' the equation is multiplied by $c2$. If $p > 0.18$, the new position is determined through the equation. When $p < 0.18$, the position must be determined through Eq. 6. $c2$ lies within the range of $[0.18, 1]$.

$$X_{it+1} = Dist_Fox_Prey_{it} \times Jump_{it} \times C2 \quad (6)$$

Turn $c2, c1$ they have values of 0.82 and 0.18. The values given above can be discovered by a fox's movement if it jumps northeast or in other directions. The fox, therefore, jumps toward the northeast if $p > 0.18$ to find a new position $Jump_{it}$ and $Dist_Fox_Prey_{it}$ that has finally been increased by $c1$. Accordingly, it has a high probability of taking advantage of a new site. However, if p it is less than 0.18, the fox undoubtedly jumps to the northeastern side. This indicates that there is a roughly 18% chance of the prey being killed. Thus, $c2$ it has been multiplied by $Jump_{it}$ and $Dist_Fox_Prey_{it}$.

3.1.2. Exploration

At this point, the fox investigates randomly based on its best position, without a specific jumping strategy. The variable search performance a and the minimum time variable $MinT$ are used to ensure random walking and searching. The computation of $MinT$ the variables is illustrated in Eq. 7 and Eq. 8, and the minimum amount tt , $MinT$ can be calculated by finding the minimum amount of tt [33].

$$tt = \frac{\sum (Time_{S_{T_{it}}}(i,:))}{dimensions}, MinT = Min(tt) \quad (7)$$

By dividing the total $Time_S_T_it(i,:)$ by the problem dimensions, the average of minimal time tt is determined [33].

$$a = 2 \times \left(it - \left(\frac{1}{Max_{it}} \right) \right) \quad (8)$$

The search phase of a FOX is significantly influenced by the calculation of $MinT$ a variable, with the goal of finding the best solution near the best solution. The use of

$rand(1, dimension)$ random variables. r balances exploration and exploitation phases.

$$X_{(it+1)} = \frac{BestX_{(it)} \times rand(1, dimension) \times MinT}{\times a} \quad (9)$$

The best solution $BestX_{it}$ has a significant impact on the exploration phase, as shown in Eq. 9. The details of the FOX can be seen in Algorithm 1 and Figure 2.

Algorithm 1: FOX optimization algorithm Pseudo cod

```

1: Initialize fox population  $X_i(i = 1, 2, \dots, n)$ 
2: While iteration <  $MAX_{it}$ 
3: Initialize Calculate the distance from fox to prey, Time(i, :), jump, a, and best fitness.
4: Calculate the fitness of each search agent
5:   IF1  $Fitness_i > Fitness_{i+1}$ 
6:     Best Fitness =  $Fitness_{i+1}$ 
7:     Best X = X (i, :)
8:   End IF1
9:   IF2  $r \geq 0.5$ 
10:     Initialize time randomly
11:     Calculate Distance_Sound_travel in Eq (1)
12:     Calculate Speed sound in Eq (2)
13:     Calculate distance fox to prey in Eq (3)
14:      $t = \text{Average}(\text{Time}(i, :))$ 
15:      $T = Tt/2$ 
16:     Calculate jump using Eq (4)
17:
18:     IF3  $P > 0.18$ 
19:
20:       Find  $X_{it+1}$  using Eq (5)
21:
22:     ElseIF  $P \leq 0.18$ 
23:
24:       Find  $X_{it+1}$  using Eq (6)
25:     End IF3
26:   Else
27:     Explore  $X_{(it+1)}$  using Eq (9)
28:   End IF2
29:   Check the limitation
30:   Update search agent by calculating their fitness
31:
32:   Update BestX
33:    $it = it + 1$ 
34: End While
35: Return BestX and BestFitness

```

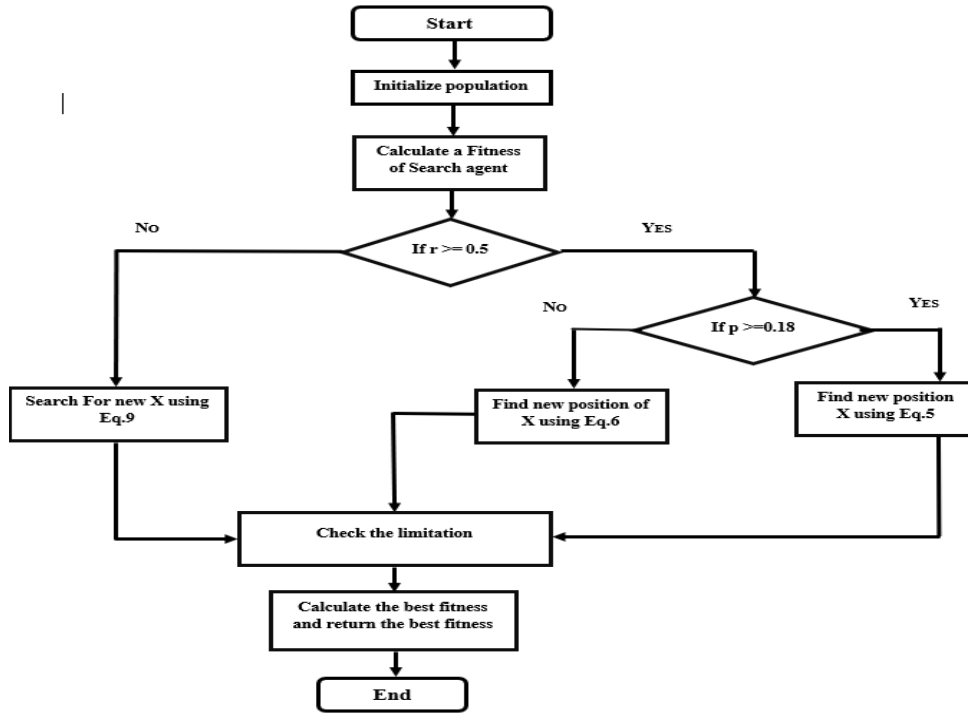


Figure 2: Flowchart of the original FOX algorithm

3.2. Improvement of Chebyshev Distance FOX (CDFOX)

These are the methods used for optimizing FOX performance within exploration and exploitation phases: *Exploitation*

3.2.1. Exploitation

Several methods have been proposed to modify the exploitation phase of FOX:

3.2.2. Chaotic maps (time)

Chaotic maps are widely used in optimization algorithms for problem-solving [42]. They are key functions in nonlinear systems and are also applied in methods like random number generation.

In chaotic maps, using a logistic map in Eq. 10 $Time(i,:)$ adds controlled randomness to the fox's travel time, helping it move toward the prey more quickly and efficiently. Chaotic sequence modulates the magnitude and timing of each agent's steps, allowing them to explore locally in a non-repetitive, dynamic way. This helps agents fine-tune their positions around the best solutions, improving convergence and avoiding getting stuck too early during the exploitation phase.

$$Time_{n+1} = a Time_n(1 - Time_n) \quad (10)$$

Eq. 10 is defined as $Time_{n+1} = a Time_n(1 - Time_n)$ where $Time_n$ represents the value of the sequence at iteration n . The iteration index is defined as $n = 0, 1, 2, \dots, N$, where N denotes the maximum number of iterations. The parameter a is the control parameter that determines the dynamic behavior of the sequence. In this study, the value $a = 4$ is used for all experiments to ensure reproducibility and consistency of results. The term $(1 - Time_n)$ represents the nonlinear component that constrains the sequence within the interval $[0, 1]$ and produces chaotic behavior when $a = 4$. The initial value is selected such that $Time_0 \in (0, 1)$. Defining all variables immediately after the equation, specifying the range of, n and clearly stating that it $a = 4$ is fixed

All experiments enhance clarity, reproducibility, and scientific rigor.

3.2.3. Chebyshev Distance

Chebyshev Distance is used to calculate how far two points are by taking the largest absolute difference in any single dimension [43]. Chebyshev distance focuses on the largest difference in any single dimension and is used to measure how far a fox is from the prey by identifying the dimension with the greatest absolute difference. To determine the fox's distance from its prey, the FOX algorithm modifies Eq. 3 into Eq. 11. In Eq.

11, the Chebyshev distance is used to calculate the distance between the best position $BestPos$ and the current position X_i of CDFOX in the population. The Chebyshev distance is formally defined as:

$$\text{Chebyshev Distance}_{fox_rate_i} = \max \left(\left| Best_{pos_i} - X_{i,j} \right| \right) \quad (11)$$

where d represents the number of dimensions, $BestPos_j$ is the j -th dimension of the best solution, and $X_{i,j}$ is the j -th dimension of the i -th individual. The operator $\max_j$ selects the maximum absolute difference across all dimensions, which accurately represents the Chebyshev distance.

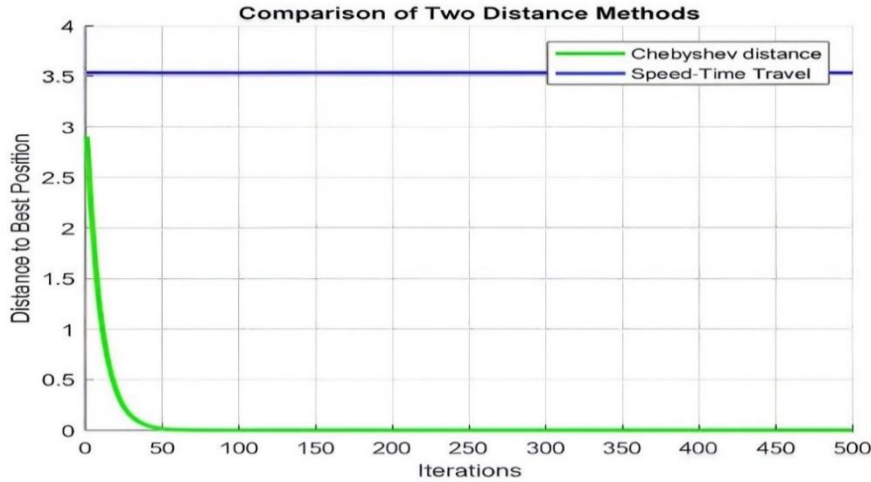


Figure 3 Comparison between FOX distance ultrasound and Chebyshev Distance of CDFOX

Figure 3 shows the comparison between travel techniques of Chebyshev distance and distance by using ultrasound (speed of sound travel) over 500 iterations. As can be seen in Figure 3, the Chebyshev distance approaches a value close to zero in a matter of iterations. This may indicate that the population diversity is reduced in the later stages. However, this situation is handled in the proposed method using the chaos map. The chaos map introduces randomness to the search process through the added nonlinear elements. This strategy distributes the search agents in the search space once again using the chaos strategy, even if the Chebyshev distance is low. This is a good strategy for avoiding local optima and conducting effective global searches in the later stages. On the other hand, the $Dist_Fox_Prey_{it}$ one that is used in FOX does not tend to become closer to the optimal point over the course of time. It is not going to change. When it comes to reaching the destination in a short amount of time, the direct distance that CDFOX employs is superior. Because of this, it is more helpful for monitoring moving targets or doing quick optimization.

$$X_{it+1} = \text{Chebyshev Distance}_{fox_rate_{it}} \times \text{Jump}_{it} \times C1 \quad (12)$$

Eq. 5 and Eq. 6, modified to Eq. 12 and Eq. 13, are utilized to determine the fox's new location. Because of p , only one equation may be processed at a time. The formulation is multiplied by $c2$ if $p \leq 0.18$. The site is determined using the formulation if $p > 0.18$. For $p < 0.18$ the location, which must be calculated by Eq. 12, $c2$ it is between $[0, 19, 1]$.

$$X_{it+1} = \text{Chebyshev Distance}_{fox_rate_{it}} \times \text{Jump}_{it} \times C2 \quad (13)$$

In turn, $c2$ they $c1$ have values of 0.82 and 0.18. The values given above can be discovered by a fox's movement, if it jumps northeast or in other directions. The fox, therefore, jumps toward the northeast if $p > 0.18$. Additionally find a new position, Jump_{it} and chebyshev $\text{Distance}_{fox_rate_{it}}$ have finally been increased by $c1$. Accordingly, it has a high probability of taking advantage of a new site. However, if p is less than 0.18, the fox undoubtedly jumps to the northeastern side. This indicates that there is a roughly 18% chance of the prey being killed. Thus, $c2$ it has been multiplied by Jump_{it} and Chebyshev $\text{Distance}_{fox_rate_{it}}$.

3.2.4. Exploration

This part uses a chaotic map to enhance FOX exploring.

3.2.5. Chaotic maps (Exploration)

In chaotic maps, the logistic map used in Eq. 14 is the same as in Eq. 10, but here it is applied for exploration, is the chaotic value at iteration, and is the control parameter that shapes the sequence's behavior. The term is obtained by multiplying the chaotic value by the given value.

$$x_{n+1} = a * x_n(1 - x_n) \quad (14)$$

$$a = \left[2 \times \left(it - \left(\frac{1}{\text{Max}_{it}} \right) \right) \right]^4 \quad (15)$$

The search performance expression in Eq. 15 and the logistic map in Eq. 14 directly influence the exploration process, as described in Eq. 16. For the logistic map described

in Eq. 14, the chaos generated is introduced here, ensuring randomness in the exploration process and helping the search agents explore a larger area in the search space. Eq. 8 modified to Eq. 15: increasing the value of the search performance expression to the power of four considerably increases the exploration ability, ensuring a larger search area, as illustrated in Figure 4. The blend of chaos and increased search performance helps increase agent movement variability, the ability to escape local optima, and CDFOX exploration capabilities. The CDFOX details can be found in Algorithm 2 and Figure 5.

$$X_i = Best_{pos} + randn(1, dimension) \times a^4 \cdot x_{n+1} \quad (16)$$

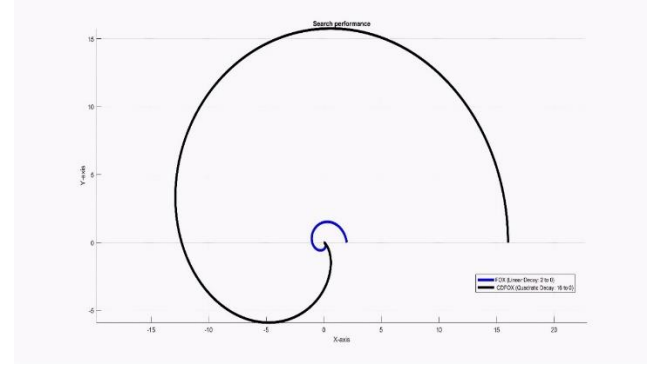


Figure 4: Compare Search performance CDFOX with FOX algorithm

To clearly illustrate the differences between the original FOX algorithm and the proposed CDFOX algorithm, the modified components are explicitly highlighted in Figure 4. Specifically, the exploration phase is improved by introducing a chaos map-based mechanism with search agent power control (modified Eq. 16). In addition, the exploitation phase is enhanced through Chebyshev distance-guided strategies (modified Eq. 12 and Eq. 13) to strengthen convergence accuracy and search efficiency.

Algorithm 2: CDFOX optimization algorithm pseudocode

```

1: Initialize the red FOX population  $X_i(i = 1, 2, \dots, n)$ 
2: While iter <  $MAX_{it}$ 
3: Initialize Chebyshev Distance_Fox_Prey, Chaotic maps_logistic map (Time (i,:), exploration) a, Best Fitness.
4: Calculate the fitness of each search agent
4:   IF  $Fitness_i < Fitness_{i+1}$ 
5:     Best Fitness= $Fitness_{i+1}$ 
6:     BestX= $X(i, :)$ 
7:   End IF
8:   IF  $r \geq 0.5$ 
9:     Initialize time randomly in Eq (10)
10:    Chebyshev Distance_Fox_Prey in Eq (11)
11:     $t = \text{Average}(\text{Time}(i, :))$ 
12:     $T = t/2$ 
13:    Calculate jump using Eq (4)
14:    Find  $X_{it+1}$  using Eq (12)
15:    IF  $P > 0.18$ 
16:      ElseIF  $P \leq 0.18$ 
17:        Find  $X_{it+1}$  using Eq (13)
18:      End IF
19:    Else
20:      Explore  $X_{(it+1)}$  using Eq (16)
21:    End IF
22:    Check the limitation
23:    Update Search agent by calculating their Fitness
24:    Update BestX
25:     $it = it + 1$ 
26:  End While
27: Return BestX and BestFitness

```

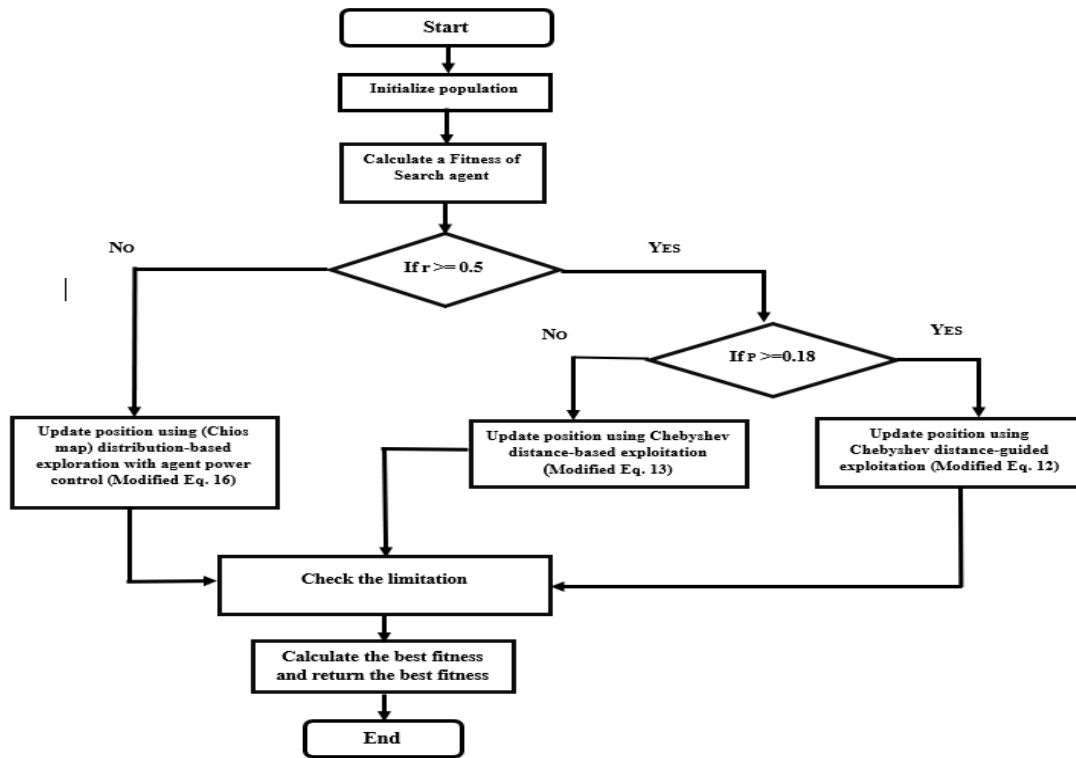


Figure.5: Flowchart of the proposed CDFOX algorithm

4. Results and Discussion

This section includes the benchmark functions and experimental results, and the performance of the proposed CDFOX algorithm is assessed and compared with the performance of eleven well-established algorithms using Classical, CEC2019, and CEC2017 benchmark test functions. In the present work, the CDFOX algorithm is tested for various numerical optimization problems. In particular, three real-world engineering optimization problems, ten CEC2019 functions, thirty CEC2017 functions, and twenty-three classical benchmark functions are adopted. The performance of the algorithms is tested under the same conditions. In particular, 30 agents are employed, the maximum iteration is 500, and the average number of independent runs is 30. MATLAB R2017a is employed on a Windows 10 environment with 8 GB of RAM. The performance of the algorithms is assessed using the best, average, worst, and standard deviation values.

The computational complexity of CDFOX is analyzed in this section. The initialization of CDFOX, which entails generating the population matrix and evaluating the objective function, takes the form of $O(Nm)$, wherein N denotes the number of FOXs and m denotes the number of problem variables. CDFOX involves only one updating phase whose

complexity takes the form of $O(NmT)$, with T denoting the total number of iterations of the algorithm. Thus, the computational complexity of CDFOX is $O(Nm(1+T))$. The computational complexity of competitor algorithms is given in Table 5. As we can see from this table, the computational complexity of such algorithms as FOX, CHOAFX, SSO, and FATA and AO, POA, PLO, and TROA are $O(Nm(1+2T))$; for SBOA and PO, the computational complexity is $O(Nm(1+3T))$, while that of mFOX is $O(Nm(1+4T))$. The complexity of competitor algorithms depends on the frequency at which the objective function is utilized within the iteration loop and the number of times the objective function is evaluated per iteration. For instance, in the case of the proposed CDFOX approach, the use of an objective function once inside the iteration loop makes its complexity be $O(Nm(1+4T))$. Therefore, it is quite clear that the computational complexity of the proposed CDFOX algorithm is comparatively low compared to some other algorithms. Nonetheless, to conduct a fair comparison between the metaheuristic approaches considered in this work, the number of iterations performed by each algorithm was modified such that the number of function evaluations (Fes) becomes equal, as seen in Table 2.

Table 2: Fairness evaluation criteria for all compared algorithm

Algorithms	agents	max iteration	Complexity	Calculation	FEs
CDFOX	30	500	$O(Nm(1+T))$	1*500	500
PO	30	167	$O(Nm(1+3T))$	3*167	500

CHOA	30	500	$O(Nm(1+T))$	1*500	500
FOX	30	500	$O(Nm(1+T))$	1*500	500
SSO	30	500	$O(Nm(1+T))$	1*500	500
AO	30	250	$O(Nm(1+2T))$	2*250	500
FATA	30	500	$O(Nm(1+T))$	1*500	500
POA	30	250	$O(Nm(1+2T))$	2*250	500
PLO	30	250	$O(Nm(1+2T))$	2*250	500
SBOA	30	167	$O(Nm(1+3T))$	3*167	500
TROA	30	250	$O(Nm(1+2T))$	2*250	500
mFOX	30	125	$O(Nm(1+4T))$	4*125	500

CDFOX algorithms are also compared to those of the FOX, Tyrannosaurus Rex Optimization Algorithm (TROA) [44], Chimp Optimization Algorithm (CHOA) [49], Pufferfish Optimization Algorithm (POA) [45], Fata Morgana Algorithm (FATA) [46], Parrot Optimizer (PO) [27], Artemis-inin Optimizer (AO) [47], Sperm Swarm Optimization (SSO) [48], Secretary Bird Optimization Algorithm (SBOA) [49], modified FOX (mFOX) [18], and Polar Lights Optimizer (PLO). It is then determined whether the results are statistically significant by comparing them. Consequently, additionally guarantee that the FOX is effective in resolving real-world applications, the FOX is employed

to address the pressure vessel design problem and the gear train design and string design problem.

These eleven optimizers were carefully selected because they are recent, well-established, and widely recognized metaheuristic algorithms in the optimization literature. Many of them are nature-inspired methods with mechanisms similar to FOX and CDFOX, ensuring a fair and meaningful comparison. In addition, they have shown strong and competitive performance on standard benchmark functions and real-world engineering design problems in previous studies. Therefore, comparing CDFOX with these algorithms provides a rigorous, comprehensive, and reliable evaluation of its effectiveness and competitiveness.

Table 3: Description of function of CEC2017

Algorithms	agents	max iteration	Complexity	Calculation	FES
CDFOX	30	500	$O(Nm(1+T))$	1*500	500
PO	30	167	$O(Nm(1+3T))$	3*167	500
CHOA	30	500	$O(Nm(1+T))$	1*500	500
FOX	30	500	$O(Nm(1+T))$	1*500	500
SSO	30	500	$O(Nm(1+T))$	1*500	500
AO	30	250	$O(Nm(1+2T))$	2*250	500
FATA	30	500	$O(Nm(1+T))$	1*500	500
POA	30	250	$O(Nm(1+2T))$	2*250	500
PLO	30	250	$O(Nm(1+2T))$	2*250	500
SBOA	30	167	$O(Nm(1+3T))$	3*167	500
TROA	30	250	$O(Nm(1+2T))$	2*250	500
mFOX	30	125	$O(Nm(1+4T))$	4*125	500

CDFOX algorithms are also compared to those of the FOX, Tyrannosaurus Rex Optimization Algorithm (TROA) [44], Chimp Optimization Algorithm (CHOA) [49], Pufferfish Optimization Algorithm (POA) [45], Fata Morgana Algorithm (FATA) [46], Parrot Optimizer (PO) [27], Artemis-inin Optimizer (AO) [47], Sperm Swarm Optimization (SSO) [48], Secretary Bird Optimization Algorithm (SBOA) [49], modified FOX (mFOX) [18], and Polar Lights Optimizer (PLO). It is then determined whether the results are statistically significant by comparing them. Consequently, additionally guarantee that the FOX is effective

in resolving real-world applications, the FOX is employed to address the pressure vessel design problem and the gear train design and string design problem.

These eleven optimizers were carefully selected because they are recent, well-established, and widely recognized metaheuristic algorithms in the optimization literature. Many of them are nature-inspired methods with mechanisms similar to FOX and CDFOX, ensuring a fair and meaningful comparison. In addition, they have shown strong and competitive performance on standard benchmark functions and real-world engineering design problems in

previous studies. Therefore, comparing CDFOX with these algorithms provides a rigorous, comprehensive, and reliable evaluation of its effectiveness and competitiveness.

Table 3: Description of function of CEC2017

Categories	ID	Function Name / Components	Function ID
Unimodal Functions	F1	Shifted and Rotated Bent Cigar Function	100
	F2	Shifted and Rotated Sum of Different Power Functions	200
	F3	Shifted and Rotated Zakharov Function	300
	F4	Shifted and Rotated Rosenbrock's Function	400
	F5	Shifted and Rotated Restraining's Function	500
	F6	Shifted and Rotated Expanded Scoffer's F6 Function	600
Multimodal Functions	F7	Shifted and Rotated Lunacek-Brautigan Function	700
	F8	Shifted and Rotated Non-Continuous Restraining's Function	800
	F9	Shifted and Rotated Levy Function	900
	F10	Shifted and Rotated Schwefel's Function	1000
	F11	Hybrid Function 1 (N=3)	1100
	F12	Hybrid Function 2 (N=3)	1200
Hybrid Function	F13	Hybrid Function 3 (N=3)	1300
	F14	Hybrid Function 4 (N=4)	1400
	F15	Hybrid Function 5 (N=4)	1500
	F16	Hybrid Function 6 (N=4)	1600
	F17	Hybrid Function 6 (N=5)	1700
	F18	Hybrid Function 6 (N=5)	1800
	F19	Hybrid Function 6 (N=5)	1900
	F20	Hybrid Function 6 (N=6)	2000
	F21	Composition Function 1 (N=3)	2100
	F22	Composition Function 2 (N=3)	2200
Composition Functions	F23	Composition Function 3 (N=4)	2300
	F24	Composition Function 4 (N=4)	2400
	F25	Composition Function 5 (N=5)	2500
	F26	Composition Function 6 (N=5)	2600
	F27	Composition Function 7 (N=6)	2700
	F28	Composition Function 8 (N=6)	2800
	F29	Composition Function 9 (N=3)	2900
	F30	Composition Function 10 (N=3)	3000
		Range:[−100,100] ^D	

Table 4: compares algorithms for thirty CEC2017 benchmark test functions.

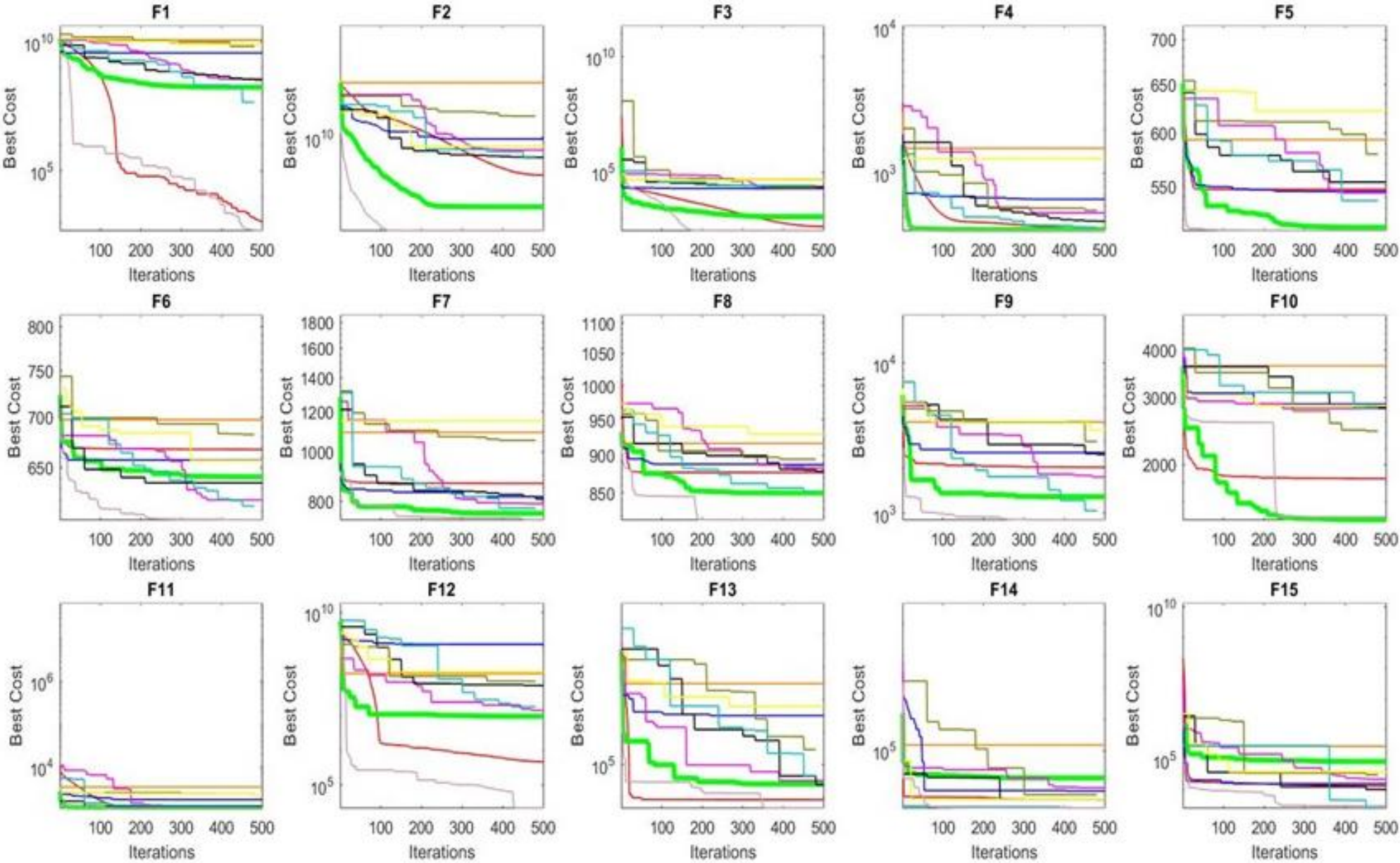
Benchmark	Statistica	CDFOX	FOX	TROA	CHWA	POA	FATA	PO	AO	SBOA	SSO	PLO	mFOX
Function	measure												
F1	Av	2.70E+03	3.62E+03	1.59E+10	7.20E+08	8.01E+09	6.59E+09	7.35E+07	1.52E+08	1.08E+10	1.02E+09	8.74E+09	3732.33737
	best	1.01E+02	2.40E+02	6.90E+09	1.44E+08	4.33E+09	1.85E+09	3.87E+06	2.29E+07	4.78E+09	3.15E+08	3.41E+09	110.470357
	worst	1.24E+04	2.43E+04	2.51E+10	1.30E+09	1.16E+10	8.97E+09	3.64E+08	3.95E+08	1.66E+10	2.19E+09	1.58E+10	44695.9469
	std	2.75E+03	5.55E+03	4.59E+09	4.47E+08	1.84E+09	1.72E+09	8.72E+07	1.04E+08	2.83E+09	4.36E+08	3.24E+09	8121.16152
	Rank	1	2	12	6	9	8	4	5	11	7	10	3
F2	Av	2.0E+02	2.77E+09	5.03E+14	7.66E+10	2.39E+10	2.65E+13	1.06E+08	6.77E+09	2.12E+13	9.62E+08	8.58E+12	766290.891
	best	200	2.00E+02	4.35E+10	2.96E+06	5.15E+08	4.69E+08	6.41E+03	3.08E+06	1.06E+09	7.74E+06	8.80E+08	203.560172
	worst	200.0003	8.19E+10	3.65E+15	1.75E+12	1.18E+11	1.80E+14	1.69E+09	1.51E+11	9.65E+13	9.82E+09	1.65E+14	8661375.52
	std	6.52E-05	1.50E+10	8.69E+14	3.19E+11	2.90E+10	4.90E+13	3.13E+08	2.73E+10	3.08E+13	1.82E+09	3.03E+13	1980822.13
	Rank	1	5	12	8	7	11	3	6	10	4	9	2
F3	Av	3.00E+02	5.17E+03	5.89E+06	1.76E+04	3.57E+04	3.84E+04	7.07E+03	2.08E+04	1.83E+05	1.04E+04	3.27E+04	2307.88324
	best	3.00E+02	3.00E+02	1.46E+04	5.27E+03	2.51E+04	1.44E+04	1.24E+03	3.91E+03	1.60E+04	5.71E+03	1.26E+04	885.893353
	worst	3.00E+02	1.41E+04	1.29E+08	2.24E+04	5.44E+04	9.50E+04	1.35E+04	4.12E+04	3.59E+06	1.71E+04	6.26E+04	6586.23553
	std	2.41E-06	4.82E+03	2.36E+07	3.86E+03	8.17E+03	2.05E+04	3.41E+03	8.75E+03	6.46E+05	2.40E+03	1.34E+04	1.45E+03
	Rank	1	4	13	7	10	11	5	8	12	6	9	3
F4	Av	4.05E+02	4.22E+02	2.02E+03	5.50E+02	1.28E+03	1.33E+03	4.40E+02	4.52E+02	1.38E+03	4.62E+02	1.07E+03	441.2105
	best	4.00E+02	4.00E+02	8.79E+02	4.15E+02	7.21E+02	5.12E+02	4.07E+02	4.14E+02	5.85E+02	4.28E+02	5.46E+02	400.0666
	worst	4.09E+02	4.88E+02	3.97E+03	7.73E+02	1.78E+03	1.70E+03	5.01E+02	5.87E+02	2.62E+03	5.00E+02	1.82E+03	554.0985
	std	2.02E+00	2.19E+01	8.55E+02	8.84E+01	2.34E+02	4.04E+02	2.75E+01	3.91E+01	5.02E+02	1.32E+01	3.67E+02	42.4555
	Rank	1	2	12	7	9	10	3	5	11	6	8	4
F5	Av	543.7313	5.51E+02	6.23E+02	5.42E+02	6.22E+02	5.82E+02	5.32E+02	5.55E+02	6.18E+02	5.38E+02	5.99E+02	544.7515
	best	518.9042	5.22E+02	5.63E+02	5.18E+02	6.05E+02	5.60E+02	5.19E+02	5.38E+02	6.00E+02	5.29E+02	5.77E+02	521.8891
	worst	599.4947	5.98E+02	6.48E+02	5.61E+02	6.41E+02	6.09E+02	5.51E+02	5.71E+02	6.46E+02	5.62E+02	6.28E+02	577.6062
	std	19.1997	2.16E+01	1.91E+01	9.45E+00	9.47E+00	1.44E+01	7.71E+00	8.97E+00	1.32E+01	7.95E+00	1.28E+01	12.962
	Rank	4	6	12	3	11	8	1	7	10	2	9	5
F6	Av	6.26E+02	6.56E+02	7.02E+02	6.19E+02	6.76E+02	6.46E+02	6.31E+02	6.11E+02	6.85E+02	6.36E+02	6.79E+02	619.253234
	best	6.01E+02	6.27E+02	6.68E+02	6.11E+02	6.56E+02	6.29E+02	6.09E+02	6.04E+02	6.36E+02	6.22E+02	6.51E+02	608.562361

	worst	6.62E+02	6.77E+02	7.32E+02	6.28E+02	6.88E+02	6.64E+02	6.64E+02	6.26E+02	7.12E+02	6.54E+02	7.05E+02	634.259229
	std	1.90E+01	9.23E+00	1.66E+01	4.18E+00	8.93E+00	8.04E+00	1.21E+01	5.02E+00	1.81E+01	7.43E+00	1.38E+01	6.81786971
	Rank	4	8	12	2	9	7	5	1	11	6	10	3
	Av	7.31E+02	8.67E+02	1.20E+03	8.10E+02	1.21E+03	9.08E+02	7.83E+02	7.60E+02	1.15E+03	7.94E+02	1.06E+03	749.471465
	best	7.19E+02	8.34E+02	9.71E+02	7.51E+02	1.03E+03	8.41E+02	7.29E+02	7.34E+02	9.70E+02	7.68E+02	9.35E+02	718.363673
F7	worst	7.50E+02	8.77E+02	1.46E+03	8.51E+02	1.40E+03	9.60E+02	8.48E+02	7.82E+02	1.38E+03	8.19E+02	1.22E+03	769.849422
	std	7.63E+00	7.64E+00	1.22E+02	2.35E+01	9.66E+01	3.17E+01	2.30E+01	1.13E+01	1.04E+02	1.23E+01	7.39E+01	10.9142027
	Rank	1	7	11	6	12	8	4	3	10	5	9	2
	Av	8.36E+02	8.73E+02	9.68E+02	8.73E+02	9.24E+02	9.03E+02	8.56E+02	8.43E+02	9.48E+02	8.74E+02	9.26E+02	831.8888
	best	8.12E+02	8.18E+02	9.29E+02	8.51E+02	8.89E+02	8.84E+02	8.26E+02	8.19E+02	8.98E+02	8.48E+02	8.87E+02	809.9496
F8	worst	8.65E+02	9.26E+02	1.03E+03	8.93E+02	9.46E+02	9.21E+02	8.99E+02	8.67E+02	9.86E+02	8.93E+02	9.72E+02	853.7275
	std	1.15E+01	1.84E+01	2.75E+01	1.23E+01	1.41E+01	9.97E+00	1.52E+01	1.17E+01	2.43E+01	9.95E+00	2.38E+01	9.796
	Rank	2	5	12	6	9	8	4	3	11	7	10	1
	Av	1.05E+03	2.09E+03	5.37E+03	1.88E+03	3.32E+03	3.76E+03	1.35E+03	1.16E+03	4.11E+03	1.54E+03	3.04E+03	1071.00043
	best	9.00E+02	1.20E+03	2.04E+03	1.20E+03	2.06E+03	1.91E+03	1.06E+03	9.30E+02	1.70E+03	1.11E+03	1.67E+03	931.832022
F9	worst	1.54E+03	2.88E+03	8.70E+03	3.20E+03	4.53E+03	5.39E+03	1.92E+03	1.49E+03	6.52E+03	2.31E+03	4.61E+03	1414.56731
	std	2.20E+02	4.09E+02	1.66E+03	4.95E+02	6.32E+02	9.45E+02	2.25E+02	1.63E+02	1.15E+03	2.71E+02	7.42E+02	123.710582
	Rank	1	7	12	6	9	10	4	3	11	5	8	2
	Av	1.87E+03	2.56E+03	3.66E+03	2.58E+03	2.72E+03	3.45E+03	2.29E+03	2.76E+03	3.61E+03	2.39E+03	3.11E+03	2181.42554
	best	1.38E+03	1.78E+03	2.99E+03	1.89E+03	2.26E+03	2.75E+03	1.64E+03	2.25E+03	2.96E+03	1.98E+03	2.46E+03	1617.55184
F10	worst	2.47E+03	3.35E+03	4.12E+03	2.87E+03	2.99E+03	3.78E+03	2.84E+03	3.24E+03	4.01E+03	2.80E+03	3.45E+03	2727.459
	std	2.51E+02	4.03E+02	2.86E+02	1.85E+02	1.79E+02	2.36E+02	2.92E+02	2.82E+02	2.55E+02	2.16E+02	2.04E+02	292.7492
	Rank	1	6	13	7	8	11	4	9	12	5	10	3
	Av	1.16E+03	1.20E+03	1.04E+04	1.22E+03	2.00E+03	3.43E+03	1.20E+03	1.52E+03	5.35E+03	1.21E+03	3.28E+03	1168.51296
	best	1.12E+03	1.10E+03	2.24E+03	1.17E+03	1.41E+03	1.27E+03	1.13E+03	1.20E+03	1.36E+03	1.14E+03	1.31E+03	1116.05756
F11	worst	1.27E+03	1.33E+03	2.44E+04	1.25E+03	3.04E+03	5.48E+03	1.32E+03	2.03E+03	1.07E+04	1.27E+03	6.24E+03	1251.17725
	std	3.81E+01	6.27E+01	7.23E+03	2.01E+01	3.83E+02	1.43E+03	4.50E+01	2.45E+02	2.38E+03	2.89E+01	1.28E+03	38.9836748
	Rank	1	3	12	6	8	10	3	7	11	5	9	2
	Av	1.05E+06	5.59E+06	3.11E+09	1.09E+07	3.22E+08	7.03E+08	1.07E+07	1.87E+07	1.37E+09	1.90E+07	1.06E+09	1255125.51
	best	5.70E+03	1.15E+05	5.20E+08	4.15E+06	7.00E+07	2.94E+07	5.34E+05	1.15E+06	2.36E+08	5.45E+06	1.88E+08	8317.04081
F12	worst	3.19E+06	2.16E+07	7.14E+09	1.65E+07	7.47E+08	1.75E+09	3.62E+07	4.73E+07	3.99E+09	4.53E+07	3.48E+09	4755745.84

F13	std	9.02E+05	6.13E+06	1.82E+09	2.92E+06	1.61E+08	4.88E+08	8.75E+06	1.34E+07	9.46E+08	1.05E+07	7.68E+08	1405822.59
	Rank	1	3	12	5	8	9	4	6	11	7	10	2
	Av	1.41E+04	1.97E+04	4.69E+08	4.49E+04	1.06E+07	2.26E+08	2.48E+04	8.79E+04	7.80E+07	6.76E+04	5.20E+07	7602.69625
	best	2.64E+03	2.46E+03	4.14E+06	9.88E+03	2.21E+05	3.05E+06	2.78E+03	2.14E+03	7.32E+05	9.50E+03	1.01E+05	2092.95955
	worst	4.91E+04	4.85E+04	1.59E+09	1.74E+05	3.17E+07	1.24E+09	2.26E+05	9.21E+05	4.56E+08	1.85E+05	1.95E+08	17392.0641
F14	std	1.16E+04	1.25E+04	4.14E+08	3.39E+04	1.04E+07	3.03E+08	4.08E+04	1.88E+05	8.99E+07	4.97E+04	5.08E+07	4142.25136
	Rank	2	3	12	5	8	11	4	7	10	6	9	1
	Av	5.67E+03	6.26E+03	1.59E+07	6.67E+03	7.49E+03	4.54E+04	4.32E+03	3.37E+04	2.48E+06	3.97E+03	9.89E+05	1959.21144
	best	1.50E+03	1.52E+03	5.92E+04	3.63E+03	1.82E+03	1.65E+03	1.52E+03	2.22E+03	4.50E+03	1.66E+03	1.59E+03	1481.35592
	worst	2.05E+04	2.11E+04	1.47E+08	7.89E+03	4.43E+04	7.67E+05	1.49E+04	2.89E+05	2.09E+07	5.69E+03	2.40E+07	3514.74998
F15	std	5.93E+03	6.00E+03	2.83E+07	9.33E+02	8.24E+03	1.40E+05	3.45E+03	5.99E+04	4.75E+06	1.44E+03	4.38E+06	561.184177
	Rank	4	5	12	6	7	9	3	8	11	2	10	1
	Av	6.19E+03	3.47E+04	2.18E+07	1.97E+04	5.11E+04	6.03E+04	1.20E+04	3.00E+04	4.41E+06	8.29E+03	2.02E+05	7.75E+03
	best	2.09E+03	3.31E+03	7.88E+04	4.45E+03	4.06E+03	8.90E+03	2.81E+03	1.87E+03	3.61E+03	2.18E+03	2.66E+03	1.76E+03
	worst	8.74E+03	1.24E+05	1.04E+08	2.93E+04	3.16E+05	8.31E+05	3.46E+04	4.25E+05	3.84E+07	2.33E+04	2.02E+06	2.49E+04
F16	std	1.72E+03	3.04E+04	2.73E+07	8.26E+03	6.20E+04	1.60E+05	8.78E+03	7.84E+04	8.45E+06	5.72E+03	4.00E+05	5.21E+03
	Rank	1	7	12	5	8	9	4	6	11	3	10	2
	Av	1.89E+03	2.19E+03	2.84E+03	1.97E+03	2.05E+03	2.39E+03	1.94E+03	1.92E+03	2.62E+03	1.92E+03	2.38E+03	1.91E+03
	best	1.61E+03	1.62E+03	2.34E+03	1.76E+03	1.84E+03	1.88E+03	1.63E+03	1.65E+03	2.13E+03	1.69E+03	2.02E+03	1.62E+03
	worst	2.22E+03	2.82E+03	3.38E+03	2.22E+03	2.28E+03	2.68E+03	2.16E+03	2.20E+03	3.36E+03	2.12E+03	2.87E+03	2.10E+03
F17	std	1.63E+02	2.98E+02	2.61E+02	1.31E+02	1.05E+02	2.07E+02	1.55E+02	1.67E+02	2.44E+02	1.26E+02	2.15E+02	132.0332
	Rank	1	8	12	6	7	10	5	4	11	3	9	2
	Av	1.83E+03	2.05E+03	2.56E+03	1.80E+03	2.13E+03	2.17E+03	1.80E+03	1.82E+03	2.26E+03	1.80E+03	2.07E+03	1770.53889
	best	1.75E+03	1.76E+03	1.99E+03	1.75E+03	1.93E+03	1.92E+03	1.75E+03	1.73E+03	1.86E+03	1.77E+03	1.86E+03	1749.34283
	worst	2.01E+03	2.49E+03	4.12E+03	1.86E+03	2.37E+03	2.47E+03	1.95E+03	2.00E+03	2.85E+03	1.87E+03	2.29E+03	1805.92982
F18	std	6.17E+01	1.69E+02	3.80E+02	2.39E+01	1.08E+02	1.21E+02	3.77E+01	7.67E+01	1.98E+02	1.97E+01	1.11E+02	15.7612832
	Rank	6	7	12	3	9	10	3	5	11	2	8	1
	Av	1.69E+04	2.35E+04	1.52E+09	1.08E+05	7.92E+06	1.49E+08	3.92E+04	3.78E+05	3.19E+08	4.40E+05	1.27E+08	10682.5403
	best	2.08E+03	2.46E+03	5.03E+07	2.52E+04	4.10E+04	3.16E+04	6.94E+03	6.64E+03	2.84E+06	2.82E+04	1.41E+06	2057.38634
	worst	3.60E+04	4.60E+04	4.96E+09	2.84E+05	3.24E+07	1.40E+09	1.24E+05	1.79E+06	1.53E+09	1.33E+06	5.29E+08	34752.0217
F18	std	1.18E+04	1.17E+04	1.26E+09	6.87E+04	8.31E+06	3.03E+08	2.76E+04	4.78E+05	3.49E+08	4.12E+05	1.38E+08	8130.0928

	Rank	2	3	12	5	8	10	4	6	11	7	9	1
F19	Av	1.18E+04	1.46E+04	1.26E+08	2.52E+04	2.64E+05	6.44E+07	5.75E+04	8.00E+04	1.38E+07	5.88E+04	5.15E+06	4837.11976
	best	2.02E+03	2.32E+03	6.99E+04	2.72E+03	2.54E+04	7.97E+04	2.08E+03	2.12E+03	1.65E+05	2.07E+03	5.62E+03	1923.48955
	worst	2.99E+04	3.46E+04	1.57E+09	5.29E+04	1.32E+06	4.86E+08	4.04E+05	1.36E+06	6.93E+07	3.06E+05	5.28E+07	14713.1318
	std	9.92E+03	9.36E+03	3.08E+08	1.03E+04	2.58E+05	1.28E+08	8.79E+04	2.52E+05	2.02E+07	9.94E+04	1.12E+07	3590.6835
	Rank	2	3	12	4	8	11	5	7	10	6	9	1
F20	Av	2.18E+03	2.25E+03	2.56E+03	2.27E+03	2.33E+03	2.39E+03	2.19E+03	2.10E+03	2.47E+03	2.17E+03	2.35E+03	2093.82958
	best	2.06E+03	2.07E+03	2.22E+03	2.09E+03	2.20E+03	2.15E+03	2.06E+03	2.04E+03	2.25E+03	2.08E+03	2.20E+03	2044.00196
	worst	2.31E+03	2.43E+03	2.88E+03	2.35E+03	2.42E+03	2.65E+03	2.30E+03	2.23E+03	2.60E+03	2.30E+03	2.50E+03	2208.05233
	std	8.42E+01	8.48E+01	1.57E+02	7.03E+01	6.49E+01	1.42E+02	6.83E+01	5.45E+01	9.50E+01	5.28E+01	7.55E+01	39.2073758
	Rank	4	6	12	7	8	10	5	2	11	3	9	1
F21	Av	2.33E+03	2.37E+03	2.47E+03	2.35E+03	2.36E+03	2.43E+03	2.26E+03	2.35E+03	2.43E+03	2.31E+03	2.41E+03	2310.68681
	best	2.20E+03	2.20E+03	2.41E+03	2.22E+03	2.28E+03	2.31E+03	2.21E+03	2.27E+03	2.34E+03	2.22E+03	2.29E+03	2200.775
	worst	2.39E+03	2.48E+03	2.52E+03	2.38E+03	2.42E+03	2.49E+03	2.35E+03	2.37E+03	2.48E+03	2.38E+03	2.46E+03	2360.87313
	std	5.64E+01	4.77E+01	2.43E+01	3.47E+01	4.86E+01	4.10E+01	5.08E+01	2.19E+01	3.00E+01	6.03E+01	4.37E+01	49.2837131
	Rank	4	8	12	6	7	10	1	5	11	3	9	2
F22	Av	2.37E+03	2.65E+03	4.27E+03	3.78E+03	3.77E+03	3.86E+03	2.33E+03	2.34E+03	3.90E+03	2.50E+03	3.75E+03	2309.00456
	best	2.30E+03	2.31E+03	2.94E+03	2.46E+03	2.91E+03	2.98E+03	2.31E+03	2.32E+03	3.28E+03	2.42E+03	2.30E+03	2239.77198
	worst	4.29E+03	4.14E+03	5.23E+03	4.47E+03	4.37E+03	5.11E+03	2.40E+03	2.37E+03	5.09E+03	2.61E+03	4.69E+03	2364.74784
	std	3.62E+02	5.57E+02	6.48E+02	6.76E+02	3.86E+02	4.81E+02	1.90E+01	1.56E+01	4.74E+02	5.20E+01	5.98E+02	23.5312936
	Rank	4	6	12	9	8	10	2	3	11	5	7	1
F23	Av	2.66E+03	2.76E+03	2.88E+03	2.66E+03	2.68E+03	2.78E+03	2.66E+03	2.66E+03	2.79E+03	2.70E+03	2.78E+03	2.65E+03
	best	2.62E+03	2.64E+03	2.78E+03	2.64E+03	2.66E+03	2.66E+03	2.63E+03	2.63E+03	2.72E+03	2.67E+03	2.72E+03	2616.73083
	worst	2.74E+03	3.04E+03	3.17E+03	2.69E+03	2.70E+03	2.83E+03	2.70E+03	2.67E+03	2.91E+03	2.73E+03	2.84E+03	2661.27431
	std	2.62E+01	7.76E+01	7.71E+01	1.04E+01	8.91E+00	4.02E+01	1.89E+01	1.06E+01	5.37E+01	1.43E+01	2.61E+01	11.9017902
	Rank	4	8	12	5	6	10	3	2	11	7	9	1
F24	Av	2.74E+03	2.91E+03	3.06E+03	2.81E+03	2.81E+03	3.00E+03	2.77E+03	2.78E+03	2.96E+03	2.75E+03	2.93E+03	2693.62988
	best	2.50E+03	2.50E+03	2.84E+03	2.77E+03	2.79E+03	2.87E+03	2.53E+03	2.63E+03	2.87E+03	2.55E+03	2.81E+03	2500.00319
	worst	2.90E+03	3.07E+03	3.40E+03	2.84E+03	2.83E+03	3.27E+03	2.83E+03	2.82E+03	3.12E+03	2.85E+03	3.13E+03	2801.37631
	std	1.14E+02	1.25E+02	1.28E+02	2.08E+01	8.86E+00	1.04E+02	6.41E+01	3.39E+01	7.81E+01	1.02E+02	6.63E+01	112.611245
	Rank	2	8	12	6	7	11	4	5	10	3	9	1

F25	Av	2.92E+03	2.96E+03	4.83E+03	3.02E+03	3.57E+03	4.35E+03	2.96E+03	2.97E+03	4.21E+03	3.03E+03	3.99E+03	2934.98088
	best	2.60E+03	2.90E+03	3.26E+03	2.94E+03	3.07E+03	3.41E+03	2.91E+03	2.94E+03	3.44E+03	2.96E+03	3.11E+03	2899.54813
	worst	2.95E+03	3.04E+03	6.71E+03	3.12E+03	4.01E+03	4.79E+03	3.07E+03	3.04E+03	5.53E+03	3.16E+03	4.77E+03	2958.68649
	std	6.40E+01	2.60E+01	8.79E+02	5.30E+01	2.21E+02	4.85E+02	3.33E+01	2.29E+01	4.81E+02	5.29E+01	4.25E+02	22.9425882
	Rank	1	4	12	6	8	11	3	5	10	7	9	2
F26	Av	3.33E+03	4.12E+03	5.16E+03	4.00E+03	4.47E+03	4.39E+03	3.24E+03	3.32E+03	4.75E+03	3.38E+03	4.35E+03	3022.24672
	best	2.80E+03	2.80E+03	4.33E+03	3.03E+03	3.45E+03	3.31E+03	2.87E+03	2.85E+03	3.90E+03	3.10E+03	3.71E+03	2600.04706
	worst	4.28E+03	4.97E+03	6.02E+03	4.28E+03	5.05E+03	4.92E+03	4.30E+03	4.53E+03	5.56E+03	3.78E+03	5.15E+03	3337.31463
	std	5.89E+02	6.79E+02	4.81E+02	3.45E+02	3.55E+02	4.62E+02	2.65E+02	4.25E+02	4.10E+02	1.59E+02	4.37E+02	198.487956
	Rank	4	7	12	6	10	9	2	3	11	5	8	1
F27	Av	3.12E+03	3.23E+03	3.50E+03	3.11E+03	3.11E+03	3.46E+03	3.11E+03	3.13E+03	3.38E+03	3.15E+03	3.26E+03	3110.8892
	best	3.09E+03	3.11E+03	3.23E+03	3.10E+03	3.11E+03	3.26E+03	3.10E+03	3.10E+03	3.11E+03	3.12E+03	3.12E+03	3097.62531
	worst	3.20E+03	3.36E+03	4.08E+03	3.16E+03	3.13E+03	3.82E+03	3.20E+03	3.23E+03	3.77E+03	3.22E+03	3.41E+03	3164.44798
	std	3.56E+01	5.63E+01	1.79E+02	1.98E+01	4.03E+00	1.48E+02	2.65E+01	3.07E+01	1.39E+02	2.78E+01	6.52E+01	13.438089
	Rank	5	8	12	2	3	11	3	6	10	7	9	1
F28	Av	3.36E+03	3.38E+03	4.29E+03	3.25E+03	3.45E+03	3.90E+03	3.38E+03	3.48E+03	3.96E+03	3.50E+03	3.85E+03	3307.43852
	best	3.10E+03	3.10E+03	3.56E+03	3.24E+03	3.34E+03	3.71E+03	3.21E+03	3.23E+03	3.55E+03	3.28E+03	3.57E+03	3100.00637
	worst	3.45E+03	3.71E+03	5.22E+03	3.28E+03	3.56E+03	3.94E+03	3.71E+03	3.65E+03	4.40E+03	3.74E+03	4.25E+03	3593.31564
	std	1.03E+02	1.58E+02	3.47E+02	8.04E+00	6.08E+01	5.34E+01	1.14E+02	1.11E+02	2.11E+02	1.31E+02	1.91E+02	141.873412
	Rank	3	4	12	1	6	10	5	7	11	8	9	2
F29	Av	3.29E+03	3.57E+03	4.12E+03	3.37E+03	3.42E+03	3.85E+03	3.28E+03	3.35E+03	3.77E+03	3.33E+03	3.66E+03	3233.17746
	best	3.17E+03	3.22E+03	3.63E+03	3.24E+03	3.29E+03	3.53E+03	3.16E+03	3.20E+03	3.27E+03	3.24E+03	3.37E+03	3154.86848
	worst	3.45E+03	3.91E+03	4.94E+03	3.48E+03	3.56E+03	4.13E+03	3.52E+03	3.50E+03	4.22E+03	3.51E+03	3.89E+03	3294.10004
	std	8.03E+01	1.76E+02	2.86E+02	6.94E+01	7.62E+01	1.75E+02	8.90E+01	7.23E+01	2.33E+02	6.30E+01	1.33E+02	36.1976116
	Rank	3	8	12	6	7	11	2	5	10	4	9	1
F30	Av	8.56E+05	3.35E+06	1.79E+08	8.41E+06	1.22E+06	5.92E+07	2.15E+06	6.20E+06	9.73E+07	2.38E+06	3.87E+07	3.32E+07
	best	1.18E+04	3.25E+04	2.68E+07	5.69E+05	6.80E+05	1.21E+06	3.61E+04	6.82E+05	1.37E+07	3.20E+05	1.26E+07	5.02E+04
	worst	2.81E+06	1.95E+07	6.11E+08	4.48E+07	3.24E+06	1.53E+08	8.91E+06	1.44E+07	3.51E+08	6.49E+06	8.51E+07	1.56E+08
	std	7.93E+05	4.11E+06	1.30E+08	8.44E+06	6.24E+05	4.02E+07	2.02E+06	3.34E+06	7.99E+07	1.39E+06	1.80E+07	3.46E+07
	Rank	1	5	12	7	2	10	3	6	11	4	9	8



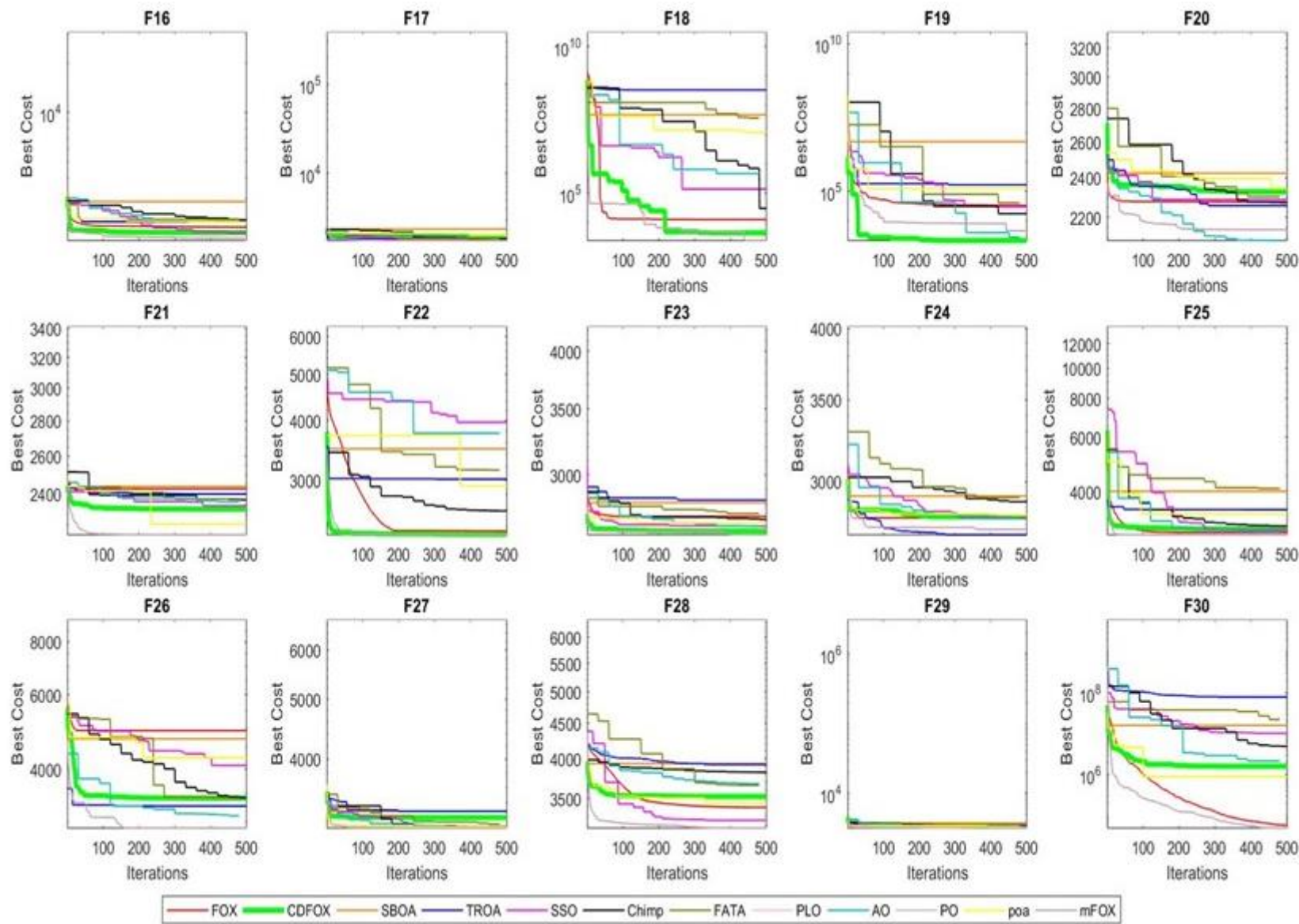


Figure 6: shows convergence behavior for CEC 2017.

Table 5: shows the processing time (seconds) of CDFOX and compared algorithms for cec2017 functions.

Benchmark Functions	CDFOX	FOX	TROA	CHWA	POA	FATA	PO	AO	SBOA	SSO	PLO	mFOX
F1	0.6096	0.3197	0.1515	1.1437	0.1189	0.0162	0.4695	0.0214	0.3167	1.0460	0.0208	0.2170
F2	0.5626	0.3046	0.1502	1.0244	0.2386	0.0092	0.4428	0.0091	0.4051	1.0982	0.0160	0.2050
F3	0.6848	0.3109	0.1638	1.1116	0.1793	0.0105	0.4545	0.0082	0.2990	0.9361	0.0172	0.2087
F4	0.6389	0.3132	0.1518	1.1354	0.1101	0.0110	0.4290	0.0128	0.3050	0.9212	0.0154	0.1999
F5	0.7038	0.2990	0.1592	1.0345	0.1177	0.0110	0.4392	0.0183	0.2938	1.2347	0.0162	0.2402
F6	0.6956	0.3763	0.1675	1.0607	0.1597	0.0115	0.5090	0.0183	0.4404	1.4022	0.0163	0.2599
F7	0.7744	0.3105	0.1518	1.2084	0.1436	0.0113	0.5215	0.0181	0.3908	1.2512	0.0167	0.2486
F8	0.7336	0.3061	0.2041	1.0191	0.1456	0.0105	0.3795	0.0204	0.3422	1.1832	0.0150	0.2323
F9	0.7060	0.3281	0.1801	1.0240	0.1387	0.0127	0.4228	0.0195	0.5170	1.1835	0.0165	0.2629
F10	0.7230	0.2862	0.1742	1.0146	0.1410	0.0200	0.4440	0.0193	0.4319	1.2080	0.0174	0.2509
F11	0.7960	0.2760	0.1652	1.0068	0.1353	0.0105	0.4201	0.0117	0.3591	1.0206	0.0158	0.2540
F12	0.6654	0.2806	0.1670	1.2208	0.1341	0.0124	0.4350	0.0198	0.3709	1.0144	0.0159	0.2407
F13	0.6845	0.2786	0.1703	1.1266	0.1174	0.0132	0.3491	0.0151	0.3584	1.0993	0.0156	0.2394
F14	0.6011	0.2812	0.1687	1.0569	0.1216	0.0128	0.3689	0.0139	0.3441	1.0794	0.0154	0.2486
F15	0.6308	0.2830	0.1626	1.1900	0.1470	0.0192	0.3615	0.0158	0.3048	1.1879	0.0148	0.2396
F16	0.6040	0.2872	0.1645	1.2124	0.1508	0.0139	0.4066	0.0201	0.3149	1.1502	0.0155	0.2514
F17	0.7323	0.3092	0.2057	1.1597	0.1707	0.0222	0.5252	0.0147	0.3786	1.1557	0.0162	0.2755
F18	0.5602	0.2832	0.1796	1.1777	0.1427	0.0128	0.4585	0.0137	0.3366	1.1304	0.0154	0.2401
F19	0.6347	0.3919	0.2638	1.3711	0.2481	0.0148	0.5987	0.0153	0.4721	1.2017	0.0181	0.3186
F20	0.7270	0.5073	0.2296	1.1390	0.1866	0.0156	0.5793	0.0283	0.4190	1.0221	0.0177	0.2773
F21	0.6569	0.5067	0.4168	1.2165	0.2163	0.0168	0.5604	0.0285	0.4039	0.9772	0.0171	0.2798
F22	0.6454	0.4861	0.2290	1.1856	0.1955	0.0176	0.7243	0.0244	0.7132	1.4270	0.0181	0.2962
F23	0.6332	0.4878	0.2168	0.9908	0.1954	0.0323	0.6180	0.0255	0.5099	1.2252	0.0200	0.3060
F24	0.6325	0.4668	0.2123	1.1581	0.1753	0.0143	0.6318	0.0178	0.4695	1.2129	0.0203	0.3096
F25	0.6380	0.4976	0.2113	1.3756	0.1883	0.0139	0.5817	0.0169	0.5754	1.1264	0.0237	0.3028
F26	0.6533	0.3585	0.2327	1.0883	0.2346	0.0174	0.6732	0.0184	0.4586	1.5593	0.0209	0.3253
F27	0.6686	0.3882	0.2440	2.0396	0.2238	0.0171	0.7817	0.0174	0.5588	1.5242	0.0189	0.3427
F28	0.6522	0.3775	0.2561	1.3375	0.1686	0.0187	0.6057	0.0177	0.4687	1.1003	0.0235	0.3244
F29	0.6599	0.3499	0.3147	1.2971	0.2041	0.0153	0.7282	0.0170	0.6157	0.8731	0.0220	0.3285
F30	0.6921	0.3901	0.3248	1.3296	0.2876	0.0222	0.6946	0.0174	0.5804	0.9659	0.0207	0.3752
SUM	20.0004	10.6420	6.1899	35.4564	5.1372	0.4570	15.6142	0.5348	12.7549	34.5177	0.5333	8.1011
Rank	10	7	5	12	4	1	9	3	8	11	2	6

Table 6: Detail of CEC2019 Benchmark

Functions	Range	f_{min}	Dim	Function names
CEC_1	[-8192, 8192]	1	9	Storn's Chebyshev polynomial fitting problem
CEC_2	[-16384, 16384]	1	16	Inverse Hilbert matrix problem
CEC_3	[-4, 4]	1	18	Lennard-Jones minimum energy cluster
CEC_4	[-100, 100]	1	10	Restraining's function
CEC_5	[-100, 100]	1	10	Greenan's function
CEC_6	[-100, 100]	1	10	Wiregrass function
CEC_7	[-100, 100]	1	10	Modified Scheffel's function
CEC_8	[-100, 100]	1	10	Expanded Schaffer's f6 function
CEC_9	[-100, 100]	1	10	Happy cat functions.
CEC_10	[-100, 100]	1	10	Ackley function

Table 7: compares algorithms for the ten CEC2019 benchmark test functions.

Benchmark Functions	Statistical measure	CDFOX	FOX	TROA	CHWA	POA	FATA	PO	AO	SBOA	SSO	PLO	mFOX
CEC01	Av	6.844E+04	2.406E+04	5.601E+12	6.358E+09	9.172E+11	6.607E+10	1.503E+05	1.503E+11	1.650E+12	5.759E+04	1.307E+12	5.253E+04
	best	4.018E+04	0.000E+00	7.273E+11	5.398E+04	8.173E+10	2.232E+07	4.476E+04	1.031E+10	2.602E+11	4.501E+04	2.115E+11	4.039E+04
	worst	3.980E+05	2.184E+05	2.033E+13	3.689E+10	1.996E+12	3.561E+11	2.837E+05	4.034E+11	3.826E+12	7.079E+04	3.691E+12	7.415E+04
	std	6.462E+04	4.294E+04	4.194E+12	9.756E+09	4.745E+11	9.684E+10	1.561E+06	1.141E+11	9.995E+11	6.296E+03	8.789E+11	8.045E+03
	Rank	4	1	12	6	9	7	5	8	11	3	10	2
CEC02	Av	1.734E+01	1.734E+01	1.763E+04	1.741E+01	1.072E+04	5.948E+02	1.751E+01	1.452E+02	1.141E+04	1.766E+01	9.714E+03	1.734E+01
	best	1.734E+01	1.734E+01	8.454E+03	1.738E+01	5.131E+03	1.873E+01	1.736E+01	3.404E+01	4.842E+03	1.746E+01	3.983E+03	1.734E+01
	worst	1.734E+01	1.735E+01	2.432E+04	1.743E+01	1.537E+04	1.814E+03	1.479E-01	9.084E+02	2.039E+04	1.796E+01	1.786E+04	1.734E+01
	std	4.268E-06	4.595E-04	4.593E+03	1.211E-02	2.867E+03	4.921E+02	1.795E+01	1.704E+02	4.225E+03	1.109E-01	3.912E+03	5.104E-04
	Rank	1	2	12	4	10	8	5	7	11	6	9	3
CEC03	Av	1.270E+01	1.270E+01	1.271E+01	1.270E+01	1.270E+01	1.271E+01	1.270E+01	1.270E+01	1.271E+01	1.270E+01	1.270E+01	1.270E+01
	best	1.270E+01	1.270E+01	1.270E+01	1.270E+01	1.270E+01	1.270E+01	1.270E+01	1.270E+01	1.270E+01	1.270E+01	1.270E+01	1.270E+01
	worst	1.270E+01	1.270E+01	1.271E+01	1.270E+01	1.270E+01	1.271E+01	2.606E-05	1.270E+01	1.271E+01	1.270E+01	1.271E+01	1.270E+01

CEC04	std	2.058E-04	2.227E-08	1.100E-03	9.432E-06	4.738E-04	1.493E-03	1.270E+01	5.340E-05	1.184E-03	2.230E-04	8.224E-04	2.892E-08
	Rank	1	3	12	5	8	11	4	6	10	7	9	2
	Av	2.514E+01	2.266E+03	3.006E+04	4.644E+03	1.757E+04	1.706E+04	1.029E+03	7.627E+02	2.293E+04	5.646E+03	1.981E+04	7.898E+02
	best	7.960E+00	5.442E+02	1.298E+04	5.148E+02	9.842E+03	8.225E+03	2.417E+02	2.874E+02	1.101E+04	2.920E+03	8.379E+03	1.174E+02
	worst	5.472E+01	6.266E+03	5.533E+04	1.070E+04	2.233E+04	4.396E+04	8.798E+02	2.016E+03	4.160E+04	7.758E+03	3.201E+04	3.004E+03
CEC05	std	1.196E+01	1.357E+03	1.098E+04	2.862E+03	3.180E+03	7.131E+03	4.459E+03	4.609E+02	8.190E+03	1.324E+03	7.395E+03	5.556E+02
	Rank	1	5	12	6	9	8	4	2	11	7	10	3
	Av	1.175E+00	5.540E+00	7.793E+00	2.881E+00	5.902E+00	5.073E+00	1.840E+00	2.083E+00	6.564E+00	2.724E+00	5.862E+00	2.026E+00
	best	1.062E+00	3.247E+00	4.237E+00	1.806E+00	3.862E+00	3.126E+00	1.339E+00	1.624E+00	3.152E+00	2.301E+00	3.260E+00	1.421E+00
	worst	1.384E+00	7.500E+00	1.102E+01	4.004E+00	7.200E+00	6.916E+00	1.903E-01	2.722E+00	1.050E+01	3.427E+00	9.264E+00	2.901E+00
CEC06	std	8.030E-02	1.112E+00	1.639E+00	5.800E-01	6.729E-01	1.066E+00	2.184E+00	1.681E-01	1.532E+00	2.830E-01	1.258E+00	3.772E-01
	Rank	1	8	12	6	10	7	2	4	11	5	9	3
	Av	3.001E+00	4.086E+00	1.571E+01	1.145E+01	1.137E+01	1.449E+01	1.098E+01	1.218E+01	1.514E+01	1.185E+01	1.344E+01	5.777E+00
	best	1.113E+00	2.076E+00	1.371E+01	9.631E+00	1.007E+01	1.192E+01	8.382E+00	9.268E+00	1.208E+01	1.015E+01	1.095E+01	2.532E+00
	worst	4.869E+00	6.892E+00	1.758E+01	1.260E+01	1.264E+01	1.615E+01	1.048E+00	1.379E+01	1.758E+01	1.292E+01	1.467E+01	8.705E+00
CEC07	std	1.104E+00	1.478E+00	9.125E-01	6.733E-01	6.312E-01	9.004E-01	1.267E+01	1.067E+00	1.195E+00	6.888E-01	8.021E-01	1.642E+00
	Rank	1	2	12	6	5	10	4	8	11	7	9	3
	Av	3.082E+02	4.195E+02	2.283E+03	1.022E+03	8.301E+02	1.941E+03	5.406E+02	1.068E+03	1.937E+03	1.090E+03	1.318E+03	3.789E+02
	best	3.314E+01	9.822E+01	8.672E+02	6.794E+02	5.328E+02	1.247E+03	2.603E+01	5.925E+02	1.404E+03	6.764E+02	8.446E+02	5.770E+01
	worst	6.116E+02	1.263E+03	2.932E+03	1.402E+03	9.509E+02	2.398E+03	2.206E+02	1.530E+03	2.516E+03	1.331E+03	1.868E+03	8.546E+02
CEC08	std	1.565E+02	2.495E+02	4.667E+02	1.693E+02	9.495E+01	3.080E+02	9.501E+02	2.503E+02	3.351E+02	1.693E+02	2.774E+02	1.827E+02
	Rank	1	3	12	6	5	11	4	7	10	8	9	2
	Av	5.393E+00	5.871E+00	8.460E+00	6.920E+00	6.755E+00	7.974E+00	6.026E+00	6.849E+00	7.941E+00	6.397E+00	7.424E+00	5.439E+00
	best	4.567E+00	4.915E+00	7.455E+00	6.614E+00	6.167E+00	7.221E+00	4.972E+00	5.730E+00	5.648E+00	5.640E+00	5.696E+00	4.179E+00
	worst	6.148E+00	7.023E+00	9.222E+00	7.228E+00	7.163E+00	8.614E+00	5.323E-01	7.897E+00	8.701E+00	7.276E+00	8.291E+00	6.172E+00
CEC09	std	4.756E-01	3.730E-01	4.702E-01	1.553E-01	2.713E-01	4.042E-01	6.844E+00	5.520E-01	6.583E-01	3.704E-01	5.304E-01	4.887E-01
	Rank	1	3	12	8	6	11	4	7	10	5	9	2
	Av	2.422E+00	2.810E+00	6.335E+03	4.759E+02	5.389E+03	2.964E+03	1.833E+01	1.142E+02	4.837E+03	4.012E+02	4.301E+03	3.694E+00
	best	2.361E+00	2.379E+00	2.660E+03	5.377E+00	2.198E+03	1.051E+03	4.220E+00	5.018E+00	1.818E+03	7.878E+01	9.850E+02	2.883E+00
	worst	2.579E+00	3.934E+00	1.278E+04	6.882E+02	7.722E+03	4.729E+03	3.601E+01	9.507E+02	7.954E+03	9.031E+02	6.560E+03	4.898E+00
CEC10	std	5.095E-02	4.014E-01	2.034E+03	2.036E+02	1.165E+03	1.086E+03	1.977E+02	1.742E+02	1.563E+03	2.377E+02	1.316E+03	6.503E-01
	Rank	1	2	12	7	11	8	4	5	10	6	9	3
CEC10	Av	2.000E+01	1.999E+01	2.118E+01	2.050E+01	2.058E+01	2.101E+01	2.026E+01	2.062E+01	2.101E+01	2.056E+01	2.087E+01	1.994E+01

best	1.995E+01	1.998E+01	2.079E+01	2.035E+01	2.040E+01	2.064E+01	1.609E+01	2.034E+01	2.067E+01	1.893E+01	2.051E+01	1.708E+01
worst	2.001E+01	2.000E+01	2.155E+01	2.060E+01	2.070E+01	2.124E+01	8.649E-01	2.089E+01	2.125E+01	2.076E+01	2.104E+01	2.028E+01
std	1.250E-02	6.350E-03	1.904E-01	6.473E-02	7.688E-02	1.429E-01	2.073E+01	1.584E-01	1.467E-01	3.245E-01	1.223E-01	5.464E-01
Rank	3	2	12	5	7	10	4	8	10	6	9	1

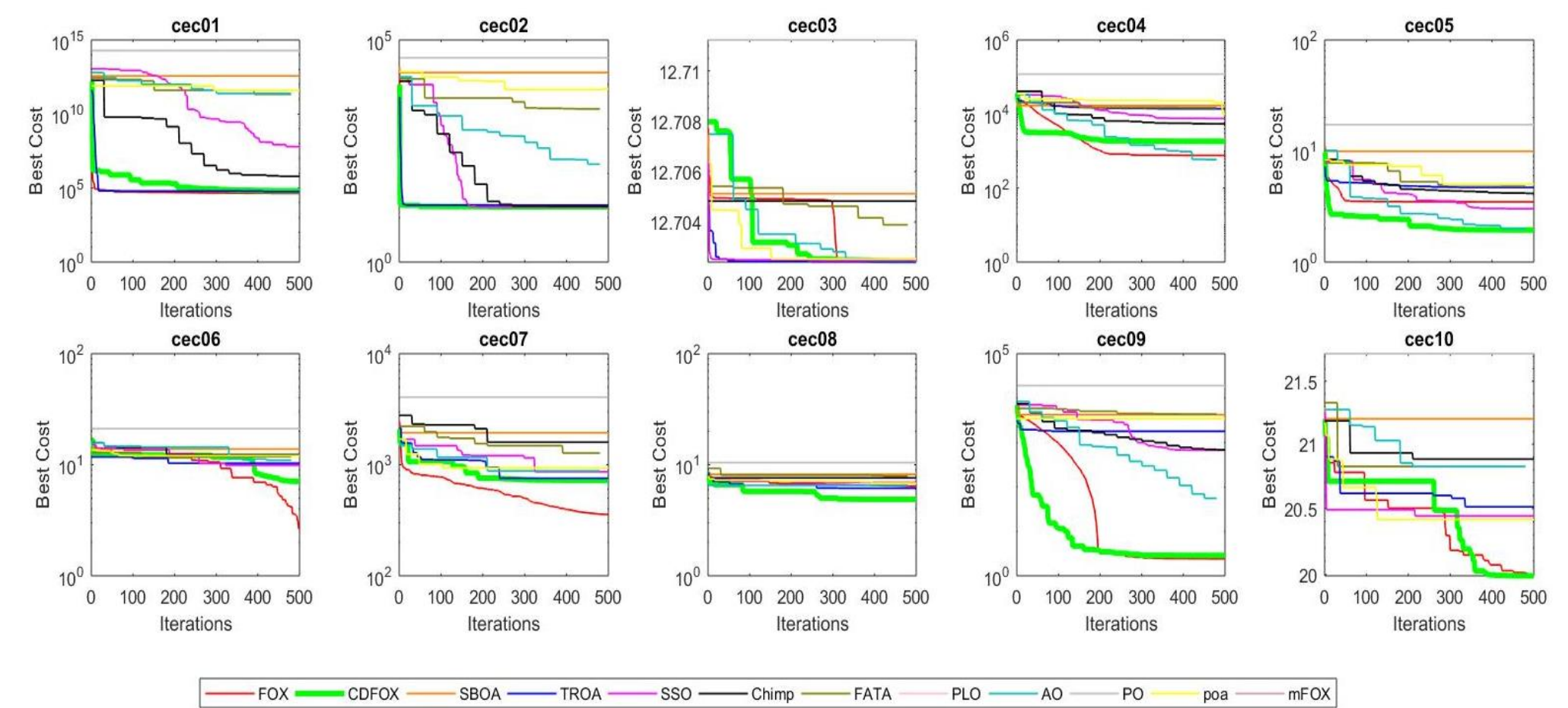


Figure 7: CEC2019 convergence behavior

Table 8: shows the processing time (seconds) of CDFOX and compared algorithms for cec2019 functions.

Benchmark Functions	CDFOX	FOX	TROA	CHWA	POA	FATA	PO	AO	SBOA	SSO	PLO	mFOX
CEC01	5.0638	5.5984	6.3221	5.5122	5.57E+00	6.101	5.4845	5.9693	10.3469	6.0157	0.1965	6.6004
CEC02	0.4116	0.4843	0.6525	0.4748	4.79E-01	0.463	0.4562	0.4515	0.1972	1.2483	0.014	0.2472
CEC03	0.4651	0.5331	0.8884	0.6272	5.43E-01	0.5172	0.5139	0.5139	0.2797	1.4321	0.0168	0.2993
CEC04	0.3851	0.4411	0.5472	0.4924	4.54E-01	0.4218	0.432	0.4439	0.2325	0.8519	0.0148	0.2526
CEC05	0.3874	0.4418	0.4586	0.4971	4.50E-01	0.4232	0.4319	0.472	0.2399	0.8543	0.0143	0.2516
CEC06	1.7598	2.0444	2.0116	2.1168	2.16E+00	1.9752	1.9911	2.1312	3.3658	2.4364	0.0676	1.9152
CEC07	0.3941	0.5345	0.4535	0.4286	5.14E-01	0.4328	0.4511	0.4651	0.2555	1.0586	0.0145	0.1983
CEC08	0.391	0.5038	0.4528	0.4341	4.73E-01	0.4302	0.4463	0.4297	0.2506	0.8758	0.0146	0.1926
CEC09	0.3952	0.4592	0.4509	0.4309	5.17E-01	0.4178	0.4155	0.4187	0.2277	0.8555	0.0141	0.1672
CEC10	0.4396	0.5289	0.8972	0.4708	5.19E-01	0.4556	0.4521	0.453	0.3065	0.8864	0.0161	0.2419
SUM	10.0927	11.5695	13.1348	11.4849	11.679	11.6378	11.0746	11.7483	15.7023	16.515	0.3833	10.3664
Rank	2	6	10	5	8	7	4	9	11	12	1	3

Table 9: Performance comparison of FOX and its modified versions

Benchmark functions	Statistical measure	CDFOX	FOX	IFOX	mFOX
CEC01	Av	6.84E+04	2.41E+04	7.04E+05	5.253E+04
	Rank	3	1	4	2
CEC02	Av	1.73E+01	1.734E+01	1.53E+01	1.734E+01
	Rank	2	3	1	4
CEC03	Av	1.27E+01	1.27E+01	7.50E+00	1.270E+01
	Rank	2	4	1	3
CEC04	Av	2.51E+01	2.27E+03	2.32E+02	7.90E+02
	Rank	1	4	2	3
CEC05	Av	1.18E+00	5.54E+00	1.70E+00	2.03E+00
	Rank	1	4	2	3
CEC06	Av	3.00E+00	4.09E+00	1.09E+01	5.78E+00
	Rank	1	2	4	3
CEC07	Av	3.082E+02	4.20E+02	1.59E+02	3.79E+02

	Rank	2	4	1	3
CEC08	Av	5.393E+00	5.87E+00	1.00E+00	5.44E+00
	Rank	2	4	1	3
CEC09	Av	2.42E+00	2.81E+00	6.30E+00	3.69E+00
	Rank	1	2	4	3
CEC10	Av	2.00E+01	2.00E+01	2.15E+01	1.99E+01
	Rank	3	2	4	1
Total rank		18	30	24	28
Rank		1	4	2	3

Table 10: Detail of classical benchmark test function

Property functions	Dim	Range	f_{min}	Function
Unimodal benchmark function	10	[-100, 100]	0	$TF1(x) = \sum_{i=1}^n x_i^2$
	10	[-10, 10]	0	$TF2(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $
	10	[-100, 100]	0	$TF_{3x=i=1nj=1} x_j^2$
	10	[-100, 100]	0	$TF4(x) = \max_i \{ x , 1 \leq i \leq n\}$
	10	[-30, 30]	0	$TF5(x) = \sum_{i=1}^{n-1} [100 (x_{i+1} - x_1^2)^2 + (x_i - 1)^2]$
	10	[-100, 100]	0	$TF6(x) = \sum_{i=1}^n ([x_i + 0.5])^2$
	10	[-1.28, 1.28]	0	$TF7(x) = \sum_{i=1}^n ix_i^4 + random[0,1]$
Multimodal benchmark function	10	[-500, 500]	569	$TF8(x) = \sum_{i=1}^n -x_i^2 \sin(\sqrt{ x_i })$
	10	[-5.12, 5.12]	0	$TF9(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$
	10	[-32, 32]	0	$TF10(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)\right) + 20 + e$
	10	[-600, 600]	0	$TF11(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$

Multimodal benchmark function with fixed dimensions	10	[-50, 50]	0	$TF12(x) = \frac{\pi}{n} \left\{ 10 \sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_i + 1)] + (y_n - 1)^2 \right\} + \sum_{i=1}^n u(x_i, 10, 100, 4)$
	10	[-50, 50]	0	$TF13(x) = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^n (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)] \right\} + \sum_{i=1}^n u(x_i, 5, 100, 4)$
	2	5.536, 65.536]	1	$f14(x) = \left \frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{j=1}^2 (x_i - a_{ij})^6} \right ^{-1}$
	10	[-5, 5]	0.00030	$f15(x) = \left[\sum_{i=1}^{11} [a_i - [(b_i^2 + b_i x_2) / (b_i^2 + b_i x_2 + x^4)]]^2 \right]$
	10	[-5, 5]	-0.316	$f16(x) = 4x_i^2 - 2.1x_1^4 + 1/3x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
	10	[-5, 5]	0.398	$f17(x) = (x_2 - 5.1x_1^2/\pi - 6)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_i + 10$
	10	[-2, 2]	3	$f18(x) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [(30 + (2x_1 + 3x_2)^2 (18 - 32x_1 + 12x_1^2 - 48x_2 + 36x_1x_2 + 27x_2^2)]$
	3	[1, 3]	-3.86	$f19(x) = (-\sum_{i=1}^4 ci \exp(-\sum_{j=1}^3 a_{ij}(x_i - p_{ij})^2)$
	6	[0, 1]	-3.	$f20(x) = (-\sum_{i=1}^4 ci \exp(-\sum_{j=1}^6 a_{ij}(x_i - p_{ij})^2)$
	4	[0, 10]]	0.1532	$f21(x) = (-\sum_{i=1}^5 [(X - a_i)(X - a_i)^T + c_i])^{-1}$
	4	[0, 10]]	0.4028	$f22(x) = (-\sum_{i=1}^7 [(X - a_i)(X - a_i)^T + c_i])^{-1}$
	4	[0, 10]]	0.5363	$f23(x) = (-\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i])^{-1}$

Table 11: compares CDFOX for twenty-three classical benchmark test functions.

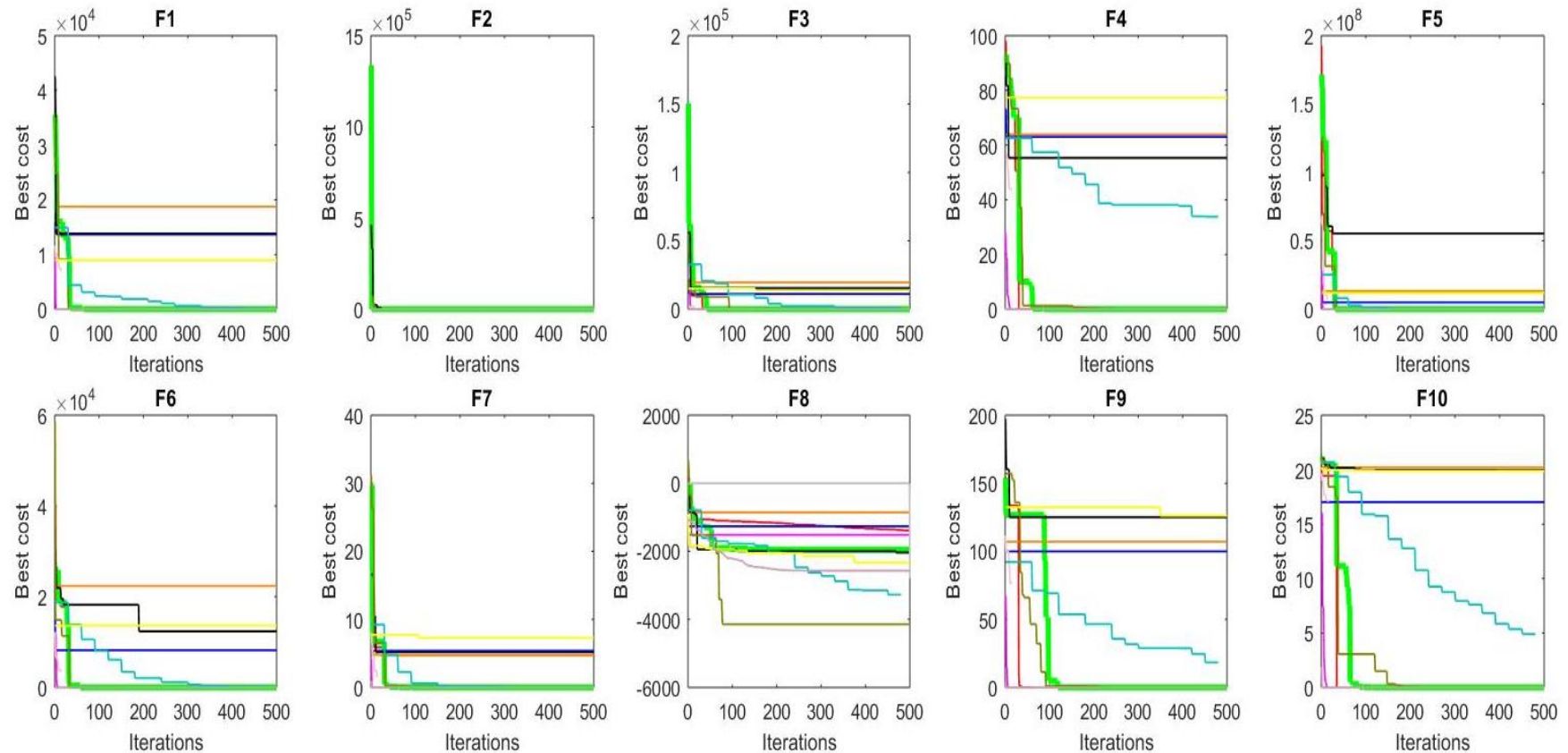
Benchmark Functions	Statistical measure	CDFOX	FOX	TROA	CHWA	POA	FATA	PO	AO	SBOA	SSO	PLO	mFOX
F1	Av	0.000E+00	0.000E+00	1.529E+04	4.189E-17	1.485E+04	4.217E+01	1.957E-05	1.529E+02	1.065E+04	6.402E-22	1.016E+04	0.000E+00
	best	0.000E+00	0.000E+00	8.599E+03	2.049E-35	6.944E+03	5.310E-03	0.000E+00	2.482E+01	4.592E+03	1.736E-35	4.976E+03	0.000E+00
	worst	0.000E+00	0.000E+00	2.034E+04	1.153E-15	1.957E+04	2.552E+02	3.616E-04	3.592E+02	1.703E+04	1.712E-20	1.468E+04	0.000E+00
	std	0.000E+00	0.000E+00	3.109E+03	2.104E-16	3.372E+03	5.417E+01	6.681E-05	9.075E+01	3.100E+03	3.135E-21	2.355E+03	0.000E+00
	Rank	1	1	12	5	11	7	6	8	10	4	9	1
F2	Av	0.000E+00	0.000E+00	5.676E+02	1.788E-12	1.510E+02	1.097E+00	4.360E-03	1.891E+00	1.136E+02	7.073E-15	5.628E+01	0.000E+00

F3	best	0.000E+00	0.000E+00	2.236E+01	6.959E-21	1.974E+01	7.755E-02	0.000E+00	9.911E-01	1.685E+01	1.824E-19	2.013E+01	0.000E+00
	worst	0.000E+00	0.000E+00	2.490E+03	3.348E-11	8.513E+02	3.442E+00	2.657E-02	4.999E+00	8.568E+02	1.762E-13	7.745E+02	0.000E+00
	std	0.000E+00	0.000E+00	7.028E+02	6.167E-12	1.633E+02	8.856E-01	6.442E-03	8.325E-01	1.757E+02	3.206E-14	1.361E+02	0.000E+00
	Rank	1	1	12	5	11	7	6	8	10	4	9	1
	Av	0.000E+00	0.000E+00	2.143E+04	2.647E-08	1.497E+04	3.541E+02	8.612E-05	2.504E+03	1.259E+04	5.515E-14	1.108E+04	0.000E+00
	best	0.000E+00	0.000E+00	1.245E+04	2.374E-14	3.689E+03	5.217E-03	0.000E+00	5.508E+02	4.223E+03	4.013E-27	6.318E+03	0.000E+00
	worst	0.000E+00	0.000E+00	3.957E+04	7.240E-07	2.130E+04	2.939E+03	8.919E-04	5.578E+03	2.494E+04	1.447E-12	1.694E+04	0.000E+00
	std	0.000E+00	0.000E+00	5.755E+03	1.319E-07	4.081E+03	6.578E+02	2.085E-04	1.130E+03	4.251E+03	2.642E-13	2.973E+03	0.000E+00
	Rank	1	1	12	5	11	7	6	8	10	4	9	1
	Av	0.000E+00	0.000E+00	6.520E+01	4.061E-06	6.918E+01	3.588E+00	2.426E-03	1.884E+01	6.108E+01	5.460E-10	5.507E+01	0.000E+00
F4	best	0.000E+00	0.000E+00	4.361E+01	1.708E-13	5.358E+01	1.250E-01	9.879E-06	6.296E+00	3.967E+01	3.060E-20	3.548E+01	0.000E+00
	worst	0.000E+00	0.000E+00	8.136E+01	4.885E-05	7.944E+01	1.196E+01	1.025E-02	2.973E+01	7.449E+01	1.254E-08	6.711E+01	0.000E+00
	std	0.000E+00	0.000E+00	1.050E+01	9.773E-06	6.094E+00	3.272E+00	3.043E-03	5.681E+00	8.206E+00	2.295E-09	7.769E+00	0.000E+00
	Rank	1	1	11	5	12	7	6	8	10	4	9	1
	Av	9.051E-06	8.614E+00	3.232E+07	8.891E+00	3.974E+07	1.986E+03	7.268E+00	2.758E+04	1.948E+07	7.906E+00	1.425E+07	0.000E+00
F5	best	6.084E-08	8.164E+00	7.336E+06	8.073E+00	1.070E+07	1.553E+00	6.214E-01	1.117E+03	1.839E+06	7.253E+00	5.395E+05	0.000E+00
	worst	1.514E-04	8.828E+00	6.154E+07	9.000E+00	7.987E+07	3.678E+04	8.912E+00	1.624E+05	6.061E+07	8.121E+00	3.221E+07	0.000E+00
	std	2.740E-05	1.530E-01	1.540E+07	2.036E-01	1.700E+07	6.784E+03	2.521E+00	2.979E+04	1.409E+07	2.745E-01	8.985E+06	0.000E+00
	Rank	2	5	11	6	12	7	3	8	10	4	9	1
	Av	6.287E-10	6.140E-05	1.592E+04	2.210E-01	1.444E+04	7.173E+01	7.005E-02	1.741E+02	1.187E+04	8.538E-01	9.660E+03	2.251E-05
F6	best	3.559E-11	1.221E-05	6.412E+03	4.290E-02	5.901E+03	1.536E-02	1.317E-03	3.688E+01	5.196E+03	5.404E-01	5.149E+03	5.758E-07
	worst	2.987E-09	1.345E-04	2.183E+04	5.229E-01	2.396E+04	4.979E+02	4.495E-01	7.788E+02	2.049E+04	1.197E+00	1.741E+04	6.435E-05
	std	6.627E-10	3.479E-05	3.888E+03	1.425E-01	4.965E+03	1.283E+02	8.616E-02	1.346E+02	4.145E+03	1.344E-01	2.865E+03	1.802E-05
	Rank	1	3	12	5	11	7	4	8	10	6	9	2
	Av	1.095E-04	1.130E-04	7.213E+00	8.274E-04	8.248E+00	5.406E-02	3.027E-02	9.960E-02	3.758E+00	6.994E-04	2.797E+00	7.894E-04
F7	best	4.413E-07	9.534E-06	2.763E+00	8.154E-05	4.163E+00	1.258E-02	3.949E-04	2.690E-02	7.921E-01	2.832E-05	4.720E-01	1.878E-04
	worst	4.203E-04	3.751E-04	1.209E+01	5.809E-03	1.482E+01	2.150E-01	2.508E-01	2.446E-01	8.268E+00	2.936E-03	5.048E+00	2.553E-03
	std	9.635E-05	8.966E-05	2.803E+00	1.059E-03	2.599E+00	4.448E-02	5.972E-02	4.891E-02	2.080E+00	6.790E-04	1.432E+00	5.455E-04
	Rank	1	2	11	5	12	7	6	8	10	3	9	4
	Av	-2.494E+03	-2.652E+03	-1.181E+03	-2.219E+03	-2.156E+03	-3.051E+03	-2.444E+03	-3.496E+03	-1.419E+03	-2.190E+03	-1.812E+03	-2.619E+03
F8	best	-4.190E+03	-3.380E+03	-1.722E+03	-2.486E+03	-2.561E+03	-4.190E+03	-3.077E+03	-3.980E+03	-2.122E+03	-2.378E+03	-2.514E+03	-3.321E+03
	worst	-2.018E+03	-2.015E+03	-5.175E+02	-2.098E+03	-1.915E+03	-1.272E+03	-1.892E+03	-3.188E+03	-9.217E+02	-2.044E+03	-1.383E+03	-2.067E+03
	std	6.923E+02	3.257E+02	3.335E+02	1.021E+02	1.381E+02	1.209E+03	2.794E+02	2.129E+02	2.635E+02	7.933E+01	2.635E+02	3.219E+02

	Rank	5	3	12	7	9	2	6	1	11	8	10	4
F9	Av	0.000E+00	0.000E+00	1.206E+02	3.894E+00	1.223E+02	8.009E+00	1.296E-03	1.972E+01	1.013E+02	0.000E+00	8.723E+01	0.000E+00
	best	0.000E+00	0.000E+00	6.947E+01	1.421E-14	9.774E+01	5.885E-01	0.000E+00	1.030E+01	6.624E+01	0.000E+00	6.082E+01	0.000E+00
	worst	0.000E+00	0.000E+00	1.461E+02	2.432E+01	1.367E+02	2.309E+01	2.166E-02	3.045E+01	1.234E+02	0.000E+00	1.080E+02	0.000E+00
	std	0.000E+00	0.000E+00	1.516E+01	5.030E+00	9.096E+00	5.801E+00	4.465E-03	5.059E+00	1.369E+01	0.000E+00	1.293E+01	0.000E+00
F10	Rank	1	1	11	6	12	7	5	8	10	1	9	1
	Av	8.882E-16	8.882E-16	1.984E+01	1.712E+01	1.973E+01	3.771E+00	1.990E-03	5.604E+00	1.824E+01	1.469E-12	1.872E+01	8.882E-16
	best	8.882E-16	8.882E-16	1.750E+01	2.931E-14	1.808E+01	1.201E-01	8.882E-16	3.900E+00	1.385E+01	8.882E-16	1.668E+01	8.882E-16
	worst	8.882E-16	8.882E-16	2.065E+01	1.995E+01	1.995E+01	7.569E+00	1.159E-02	8.351E+00	2.017E+01	2.760E-11	1.986E+01	8.882E-16
F11	std	0.000E+00	0.000E+00	6.269E-01	6.725E+00	4.094E-01	2.067E+00	2.498E-03	1.249E+00	1.860E+00	5.102E-12	7.794E-01	0.000E+00
	Rank	2	2	12	8	11	6	5	7	9	4	10	1
	Av	0.000E+00	0.000E+00	1.432E+02	1.108E-01	1.438E+02	1.849E+00	1.583E-05	2.718E+00	1.023E+02	1.117E-10	8.079E+01	0.000E+00
	best	0.000E+00	0.000E+00	8.637E+01	0.000E+00	4.876E+01	1.281E-02	0.000E+00	1.363E+00	3.634E+01	0.000E+00	2.812E+01	0.000E+00
F12	worst	0.000E+00	0.000E+00	2.133E+02	5.981E-01	1.887E+02	6.131E+00	2.674E-04	4.802E+00	1.736E+02	3.347E-09	1.301E+02	0.000E+00
	std	0.000E+00	0.000E+00	3.312E+01	1.608E-01	2.777E+01	1.468E+00	4.991E-05	9.488E-01	3.457E+01	6.110E-10	2.381E+01	0.000E+00
	Rank	1	1	11	6	12	7	5	8	10	4	9	1
	Av	1.482E-09	6.305E-06	8.049E+07	1.447E-01	6.379E+07	2.259E+00	2.592E-02	1.897E+01	3.600E+07	1.854E-01	2.076E+07	1.042E-02
F13	best	1.048E-11	1.209E-06	1.476E+07	8.404E-03	1.232E+07	6.904E-05	6.365E-03	4.263E+00	1.667E+06	7.489E-02	2.514E+06	5.247E-06
	worst	1.061E-08	1.727E-05	1.776E+08	1.615E+00	1.623E+08	1.467E+01	6.059E-02	5.539E+01	1.587E+08	3.148E-01	8.966E+07	3.115E-01
	std	2.352E-09	3.487E-06	4.121E+07	4.002E-01	3.915E+07	3.512E+00	1.359E-02	1.313E+01	3.708E+07	6.219E-02	1.728E+07	5.686E-02
	Rank	1	2	12	5	11	7	4	8	10	6	9	3
F14	Av	2.601E-09	3.830E-05	1.560E+08	9.360E-01	1.474E+08	2.867E+00	7.629E-02	9.602E+03	1.147E+08	4.984E-01	7.046E+07	1.940E-03
	best	2.572E-11	5.699E-06	1.307E+07	7.300E-01	6.511E+06	1.348E-02	9.009E-03	5.003E+00	1.030E+07	3.346E-01	1.501E+07	1.404E-05
	worst	1.197E-08	8.930E-05	4.996E+08	9.989E-01	2.819E+08	1.450E+01	2.310E-01	1.199E+05	2.825E+08	6.770E-01	1.546E+08	2.120E-02
	std	3.168E-09	2.115E-05	9.981E+07	6.932E-02	7.841E+07	4.372E+00	5.989E-02	2.471E+04	6.578E+07	8.329E-02	3.804E+07	4.963E-03
F15	Rank	1	2	12	6	11	7	4	8	10	5	9	3
	Av	9.980E-01	9.874E+00	1.994E+02	9.981E-01	3.453E+00	5.630E+00	3.955E+00	5.767E+00	4.281E+01	3.713E+00	7.671E+00	4.170E+00
	best	9.980E-01	1.992E+00	1.441E+01	9.980E-01	9.980E-01	9.980E-01	9.980E-01	9.980E-01	2.039E+00	1.002E+00	9.980E-01	9.980E-01
	worst	3.968E+00	1.267E+01	4.667E+02	9.986E-01	6.562E+00	1.807E+01	1.076E+01	1.737E+01	2.316E+02	1.076E+01	2.206E+01	1.267E+01
F15	std	7.583E-01	3.639E+00	1.766E+02	1.650E-04	1.388E+00	4.705E+00	2.899E+00	5.103E+00	6.038E+01	3.287E+00	5.630E+00	2.820E+00
	Rank	1	10	12	2	3	7	5	8	11	4	9	6
F15	Av	3.519E-04	1.915E-03	2.004E-01	1.318E-03	3.695E-03	1.382E-02	7.127E-04	9.749E-03	7.240E-02	6.216E-04	2.698E-02	4.292E-04
	best	3.075E-04	3.083E-04	1.333E-02	1.236E-03	1.704E-03	3.589E-04	3.102E-04	5.212E-04	3.575E-03	3.513E-04	2.448E-03	3.079E-04

F16	worst	7.217E-04	2.036E-02	8.013E-01	1.446E-03	8.499E-03	1.433E-01	1.701E-03	6.015E-02	4.227E-01	1.107E-03	9.297E-02	6.557E-04
	std	9.936E-05	5.030E-03	1.946E-01	5.439E-05	2.039E-03	2.729E-02	3.992E-04	1.361E-02	7.598E-02	1.905E-04	2.117E-02	1.148E-04
	Rank	1	6	12	5	7	9	4	8	11	3	10	2
	Av	-1.032E+00	-1.032E+00	3.905E-01	-1.032E+00	-9.589E-01	-6.848E-01	-1.032E+00	-1.025E+00	-3.361E-01	-1.030E+00	-8.392E-01	-1.032E+00
	best	-1.032E+00	-1.032E+00	-1.017E+00	-1.032E+00	-1.028E+00	-1.032E+00	-1.032E+00	-1.032E+00	-1.017E+00	-1.032E+00	-1.030E+00	-1.032E+00
	worst	-1.032E+00	-1.032E+00	3.232E+00	-1.032E+00	-7.316E-01	2.228E-02	-1.032E+00	-8.597E-01	2.182E+00	-1.026E+00	-5.427E-02	-1.032E+00
	std	5.566E-15	6.401E-07	1.109E+00	1.635E-05	7.400E-02	3.744E-01	1.557E-05	3.134E-02	6.834E-01	1.351E-03	2.373E-01	4.959E-09
	Rank	1	1	12	1	8	10	1	7	11	6	9	5
	Av	3.979E-01	3.979E-01	2.011E+00	3.983E-01	4.555E-01	6.739E-01	3.979E-01	4.003E-01	1.522E+00	4.061E-01	6.134E-01	3.979E-01
	best	3.979E-01	3.979E-01	4.762E-01	3.979E-01	4.007E-01	3.979E-01	3.979E-01	3.979E-01	4.088E-01	3.979E-01	3.996E-01	3.979E-01
F17	worst	3.979E-01	3.979E-01	7.185E+00	3.996E-01	6.519E-01	9.935E-01	3.982E-01	4.294E-01	5.935E+00	4.399E-01	1.348E+00	3.979E-01
	std	6.144E-16	2.664E-07	1.639E+00	3.748E-04	5.291E-02	1.948E-01	7.429E-05	6.954E-03	1.184E+00	9.899E-03	2.583E-01	1.630E-09
	Rank	1	1	12	5	8	10	1	6	11	7	9	1
	Av	3.000E+00	8.400E+00	7.034E+01	3.000E+00	4.582E+00	1.417E+01	3.001E+00	9.524E+00	2.726E+01	3.000E+00	1.196E+01	3.000E+00
	best	3.000E+00	3.000E+00	3.646E+00	3.000E+00	3.053E+00	3.008E+00	3.000E+00	3.000E+00	3.120E+00	3.000E+00	3.029E+00	3.000E+00
F18	worst	3.000E+00	8.400E+01	4.713E+02	3.001E+00	1.635E+01	3.564E+01	3.004E+00	3.129E+01	9.957E+01	3.004E+00	3.496E+01	3.000E+00
	std	2.969E-14	2.055E+01	9.743E+01	1.273E-04	2.692E+00	1.242E+01	1.035E-03	1.176E+01	2.709E+01	7.969E-04	1.006E+01	2.165E-07
	Rank	1	7	12	1	6	10	5	8	11	1	9	4
	Av	-3.863E+00	-3.863E+00	-3.270E+00	-3.854E+00	-3.827E+00	-3.770E+00	-3.848E+00	-3.861E+00	-3.526E+00	-3.552E+00	-3.709E+00	-3.863E+00
	best	-3.863E+00	-3.863E+00	-3.753E+00	-3.862E+00	-3.856E+00	-3.861E+00	-3.863E+00	-3.863E+00	-3.799E+00	-3.842E+00	-3.860E+00	-3.863E+00
F19	worst	-3.863E+00	-3.863E+00	-2.542E+00	-3.852E+00	-3.751E+00	-3.614E+00	-3.727E+00	-3.846E+00	-2.846E+00	-3.170E+00	-3.398E+00	-3.863E+00
	std	1.676E-08	8.822E-07	3.340E-01	1.659E-03	2.913E-02	6.565E-02	3.022E-02	3.365E-03	2.266E-01	2.125E-01	1.272E-01	3.368E-05
	Rank	1	1	12	5	7	8	6	4	11	10	9	3
	Av	-3.231E+00	-3.256E+00	-1.673E+00	-2.697E+00	-2.756E+00	-2.293E+00	-3.226E+00	-3.261E+00	-2.187E+00	-2.036E+00	-2.214E+00	-3.322E+00
	best	-3.322E+00	-3.322E+00	-2.978E+00	-3.263E+00	-3.089E+00	-3.203E+00	-3.321E+00	-3.320E+00	-2.979E+00	-2.906E+00	-2.882E+00	-3.322E+00
F20	worst	-3.202E+00	-3.186E+00	-7.434E-01	-1.693E+00	-2.482E+00	-1.539E+00	-3.007E+00	-3.191E+00	-1.327E+00	-1.218E+00	-1.308E+00	-3.322E+00
	std	5.123E-02	6.247E-02	6.396E-01	4.198E-01	1.170E-01	3.755E-01	9.141E-02	5.660E-02	4.140E-01	5.044E-01	4.239E-01	3.123E-05
	Rank	4	3	12	7	6	8	5	2	10	11	9	1
	Av	-1.015E+01	-5.058E+00	-5.408E-01	-1.943E+00	-9.965E-01	-1.716E+00	-5.554E+00	-5.397E+00	-7.540E-01	-1.823E+00	-1.336E+00	-9.983E+00
	best	-1.015E+01	-5.101E+00	-1.121E+00	-5.015E+00	-2.343E+00	-5.235E+00	-1.012E+01	-1.014E+01	-2.076E+00	-3.035E+00	-7.756E+00	-1.015E+01
F21	worst	-1.015E+01	-5.055E+00	-2.367E-01	-4.973E-01	-5.478E-01	-7.812E-01	-5.036E+00	-2.354E+00	-2.612E-01	-9.548E-01	-4.789E-01	-5.055E+00
	std	1.022E-11	1.156E-02	2.453E-01	1.852E+00	4.360E-01	1.111E+00	1.531E+00	2.966E+00	4.380E-01	5.740E-01	1.343E+00	9.308E-01
	Rank	1	5	12	6	10	8	3	4	11	7	9	2

F22	Av	-1.040E+01	-5.625E+00	-5.500E-01	-3.499E+00	-1.098E+00	-2.582E+00	-6.092E+00	-5.965E+00	-9.225E-01	-2.162E+00	-1.163E+00	-9.683E+00
	best	-1.040E+01	-1.040E+01	-1.336E+00	-5.072E+00	-1.852E+00	-8.148E+00	-1.039E+01	-1.038E+01	-1.559E+00	-6.458E+00	-2.656E+00	-1.040E+01
	worst	-1.040E+01	-5.088E+00	-3.083E-01	-9.077E-01	-7.189E-01	-1.113E+00	-3.717E+00	-2.728E+00	-4.786E-01	-1.110E+00	-5.519E-01	-3.724E+00
	std	3.825E-11	1.620E+00	1.956E-01	2.004E+00	2.832E-01	1.498E+00	2.182E+00	3.391E+00	3.435E-01	1.089E+00	5.114E-01	1.894E+00
	Rank	1	5	12	6	10	7	3	4	11	8	9	2
F23	Av	-1.054E+01	-5.669E+00	-8.254E-01	-4.217E+00	-1.492E+00	-2.425E+00	-6.792E+00	-5.101E+00	-9.814E-01	-1.981E+00	-1.908E+00	-1.053E+01
	best	-1.054E+01	-1.054E+01	-1.628E+00	-5.094E+00	-6.654E+00	-9.934E+00	-1.053E+01	-1.050E+01	-1.650E+00	-3.211E+00	-6.566E+00	-1.054E+01
	worst	-1.054E+01	-5.128E+00	-5.125E-01	-9.438E-01	-8.277E-01	-1.210E+00	-2.798E+00	-2.383E+00	-5.998E-01	-1.367E+00	-9.597E-01	-1.049E+01
	std	6.671E-12	1.650E+00	2.638E-01	1.663E+00	1.030E+00	1.584E+00	2.698E+00	3.272E+00	2.637E-01	5.234E-01	1.292E+00	8.795E-03
	Rank	1	4	12	6	10	7	3	5	11	8	9	2



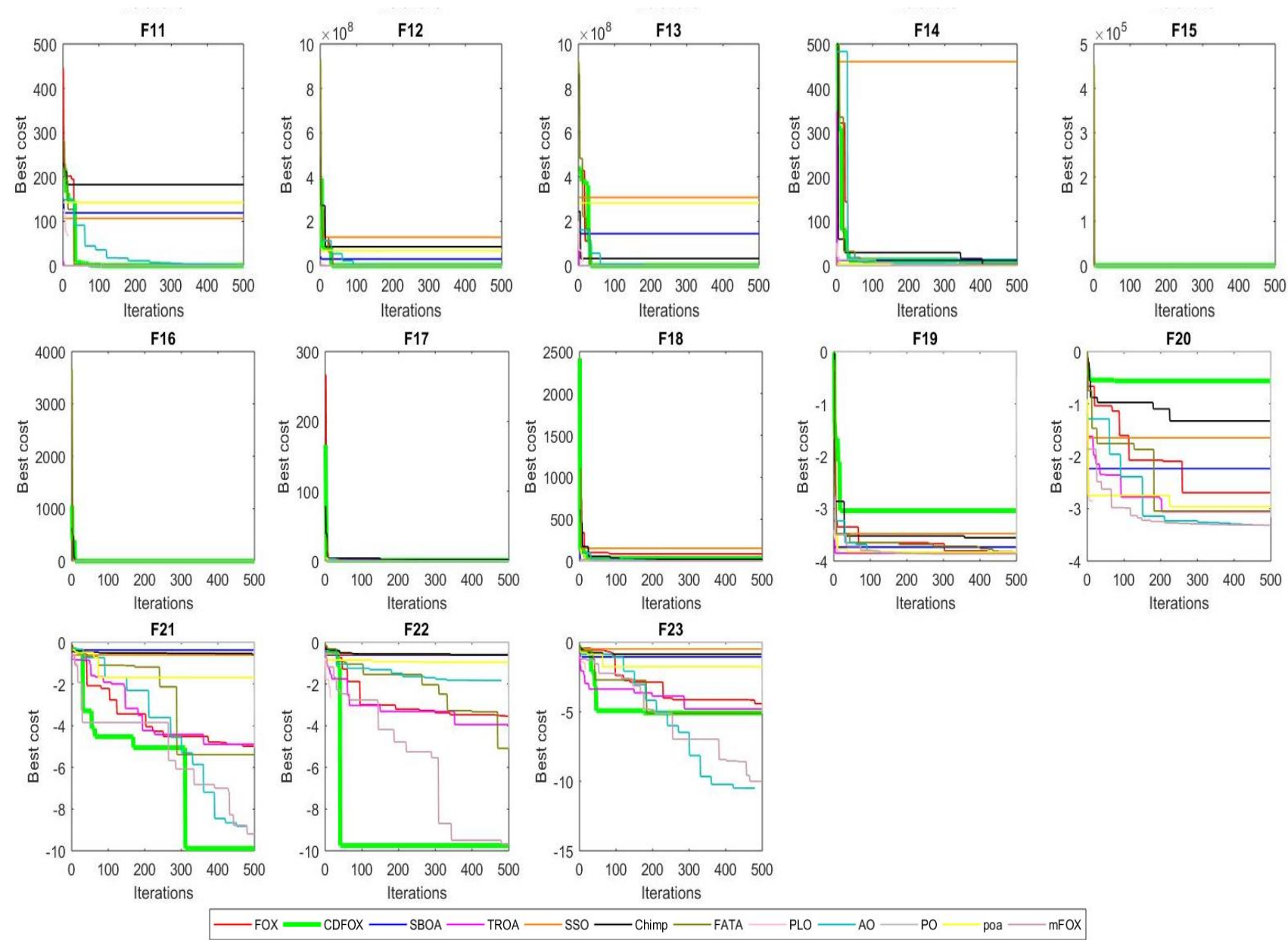


Figure 8: The classical benchmark test problems' convergence behavior

Table 12: shows the processing time (seconds) of CDFOX and compared algorithms for classical benchmark functions.

Benchmark Functions	CDFOX	FOX	TROA	CHWA	POA	FATA	PO	AO	SBOA	SSO	PLO	mFOX
F1	0.4718	0.3936	0.1924	0.9821	0.1084	0.0182	0.2949	0.0245	0.1899	0.7279	0.0152	0.1570
F2	0.4813	0.4166	0.2083	1.1530	0.1207	0.0097	0.2640	0.0136	0.2070	0.7255	0.0116	0.1608
F3	0.6063	0.5986	0.3981	1.2591	0.3795	0.0174	0.4818	0.0206	0.4916	0.8415	0.0150	0.2724
F4	0.4769	0.3432	0.1818	1.1504	0.0893	0.0108	0.2377	0.0170	0.2481	0.7020	0.0095	0.1460
F5	0.4909	0.3636	0.1987	1.2220	0.1002	0.0130	0.2789	0.0168	0.2518	0.7369	0.0100	0.1719
F6	0.4688	0.3623	0.1782	1.0854	0.0715	0.0112	0.2363	0.0184	0.2327	0.7078	0.0092	0.1458
F7	0.4863	0.5161	0.1982	1.1228	0.1078	0.0127	0.2920	0.0186	0.2694	0.8033	0.0100	0.1724
F8	0.4807	0.5235	0.2002	1.0866	0.1021	0.0121	0.2783	0.0199	0.3277	0.7452	0.0098	0.1686
F9	0.4657	0.2974	0.1741	1.0552	0.1128	0.0106	0.3351	0.0192	0.3118	0.7272	0.0091	0.1489
F10	0.4848	0.3170	0.2084	1.2399	0.1213	0.0117	0.4210	0.0239	0.2683	0.7255	0.0101	0.1648
F11	0.4981	0.3233	0.2389	1.1167	0.1172	0.0123	0.3272	0.0187	0.3670	0.7534	0.0102	0.1794
F12	0.6056	0.4473	0.3192	1.4962	0.2356	0.0196	0.5251	0.0197	0.5090	0.8605	0.0153	0.2955
F13	0.6052	0.3770	0.5899	1.5781	0.2604	0.0187	0.7071	0.0196	0.4931	0.8631	0.0146	0.2887
F14	1.0253	0.9779	0.9979	1.8076	0.8524	0.0454	2.3673	0.0349	2.3696	0.9139	0.0322	0.8530
F15	0.4804	0.2747	0.2036	1.8520	0.0794	0.0112	0.3172	0.0109	0.1941	0.7184	0.0097	0.1529
F16	0.4583	0.2243	0.1822	1.5734	0.0529	0.0103	0.2382	0.0097	0.1593	0.6982	0.0087	0.1389
F17	0.4511	0.2136	0.1609	1.3369	0.0453	0.0097	0.2274	0.0098	0.1520	0.6867	0.0086	0.1303
F18	0.4476	0.2349	0.1196	1.3866	0.0464	0.0099	0.2211	0.0103	0.1433	0.6890	0.0085	0.1282
F19	0.4148	0.3375	0.1344	0.6231	0.1029	0.0125	0.3649	0.0110	0.2523	0.3192	0.0099	0.1883
F20	0.4457	0.4874	0.1614	1.1838	0.1047	0.0132	0.4426	0.0126	0.2533	0.5001	0.0103	0.1887
F21	0.4925	0.5239	0.2587	0.8201	0.2017	0.0175	0.6483	0.0145	0.4019	0.4665	0.0129	0.2603
F22	0.5508	0.4963	0.3436	0.9497	0.2628	0.0197	0.6230	0.0169	0.5168	0.5145	0.0144	0.3086
F23	0.6161	0.4966	0.3969	1.0508	0.3265	0.0241	0.7368	0.0193	0.6578	0.6499	0.0174	0.3820
SUM	12.0052	9.5467	6.2457	28.1315	4.0019	0.3515	10.8662	0.4005	9.2678	16.0762	0.2823	5.2032
Rank	10	8	6	12	4	2	9	3	7	11	1	5

Table 13: P-value on thirty CEC2017 benchmark test functions

Bench Functions	CDFOX VS FOX	CDFOX VS TROA	CDFOX VS CHWA	CDFOX VS POA	CDFOX VS FATA	CDFOX VS PO	CDFOX VS AO	CDFOX VS SBOA	CDFOX VS SSO	CDFOX VS PLO	CDFOX VS mFOX
F1	1.761E-01	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	6.952E-01
F2	4.311E-08	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	2.602E-08
F3	2.971E-11	2.971E-11	2.971E-11	2.971E-11	2.971E-11	2.971E-11	2.971E-11	2.971E-11	2.971E-11	2.971E-11	2.971E-11
F4	6.283E-06	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.338E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	1.000E+00
F5	5.592E-01	8.148E-11	3.016E-11	3.018E-11	1.729E-07	4.981E-04	6.567E-02	3.018E-11	9.823E-01	1.463E-10	8.650E-01
F6	7.088E-08	3.020E-11	3.020E-11	4.504E-11	1.106E-04	2.170E-01	7.617E-03	1.777E-10	5.561E-04	8.993E-11	3.255E-01
F7	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	5.462E-09	7.599E-07	3.020E-11	4.504E-11	3.020E-11	3.965E-08
F8	2.919E-09	3.016E-11	3.016E-11	3.016E-11	3.016E-11	9.527E-07	1.327E-02	3.016E-11	4.072E-11	3.016E-11	7.506E-01
F9	1.065E-09	3.001E-11	2.993E-11	3.001E-11	3.001E-11	4.850E-03	1.412E-01	4.052E-11	1.996E-06	4.052E-11	1.694E-02
F10	2.014E-08	3.018E-11	2.105E-11	4.972E-11	3.018E-11	1.860E-06	8.148E-11	3.018E-11	8.988E-11	3.336E-11	9.209E-05
F11	1.383E-02	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.006E-04	1.094E-10	3.020E-11	2.390E-08	3.020E-11	1.907E-01
F12	3.501E-03	3.020E-11	3.020E-11	3.020E-11	3.020E-11	2.602E-08	2.154E-10	3.020E-11	3.020E-11	3.020E-11	4.290E-01
F13	7.013E-02	3.020E-11	3.018E-11	3.020E-11	3.020E-11	3.632E-01	7.617E-03	3.020E-11	4.504E-11	3.020E-11	1.273E-02
F14	9.941E-01	3.020E-11	3.018E-11	3.478E-01	9.069E-03	2.905E-01	3.988E-04	4.200E-10	6.952E-01	1.108E-06	1.302E-03
F15	2.002E-06	3.020E-11	3.020E-11	4.573E-09	3.020E-11	8.684E-03	1.154E-01	2.015E-08	6.203E-04	2.922E-09	3.632E-01
F16	1.106E-04	3.020E-11	3.020E-11	1.325E-04	1.070E-09	2.398E-01	6.627E-01	4.077E-11	2.813E-02	3.820E-10	5.592E-01
F17	5.600E-07	3.338E-11	3.020E-11	6.696E-11	4.975E-11	6.097E-03	3.112E-01	7.389E-11	2.772E-01	3.474E-10	4.801E-07
F18	3.778E-02	3.020E-11	3.020E-11	3.020E-11	5.494E-11	1.325E-04	4.573E-09	3.020E-11	3.020E-11	3.020E-11	4.676E-02
F19	1.120E-01	3.020E-11	3.020E-11	4.077E-11	3.020E-11	5.084E-03	6.972E-03	3.020E-11	2.681E-04	1.464E-10	2.324E-02
F20	6.787E-02	1.777E-10	3.020E-11	1.157E-07	8.841E-07	5.793E-01	7.739E-06	1.329E-10	9.705E-01	1.311E-08	2.006E-04
F21	2.921E-02	3.020E-11	3.020E-11	3.953E-01	1.695E-09	3.497E-09	1.000E+00	2.872E-10	3.871E-01	4.420E-06	1.729E-06
F22	5.573E-10	1.206E-10	3.020E-11	4.200E-10	3.820E-10	5.573E-10	5.573E-10	3.159E-10	5.573E-10	3.497E-09	2.531E-04
F23	3.197E-09	3.020E-11	3.020E-11	7.221E-06	1.206E-10	8.534E-01	9.705E-01	6.066E-11	2.015E-08	4.077E-11	3.592E-05
F24	1.102E-08	4.077E-11	1.493E-04	4.744E-06	4.975E-11	1.958E-01	1.120E-01	6.066E-11	2.839E-04	1.613E-10	1.837E-02

F25	1.547E-09	3.020E-11	8.153E-11	3.020E-11	3.020E-11	1.249E-05	1.411E-09	3.020E-11	3.020E-11	3.020E-11	9.031E-04
F26	4.214E-04	2.365E-10	5.825E-03	2.372E-07	1.192E-06	1.907E-01	1.087E-01	8.465E-09	7.243E-02	1.248E-05	7.283E-01
F27	3.646E-08	3.020E-11	4.676E-02	6.204E-01	3.020E-11	3.329E-01	1.958E-01	2.610E-10	7.295E-04	1.695E-09	6.952E-01
F28	5.106E-01	3.020E-11	9.514E-06	4.459E-04	3.020E-11	6.100E-01	2.278E-05	3.020E-11	9.833E-08	3.020E-11	1.858E-01
F29	6.010E-08	3.020E-11	7.295E-04	1.028E-06	3.020E-11	5.692E-01	6.377E-03	5.573E-10	1.501E-02	8.153E-11	4.637E-03
F30	8.684E-03	3.020E-11	8.891E-10	1.112E-01	1.329E-10	3.671E-03	2.439E-09	3.020E-11	1.311E-08	3.020E-11	3.159E-10

Table 14: P-value on ten CEC2019 benchmark test functions

Benchmark functions	CDFOX VS FOX	CDFOX VS TROA	CDFOX VS CHWA	CDFOX VS POA	CDFOX VS FATA	CDFOX VS PO	CDFOX VS AO	CDFOX VS SBOA	CDFOX VS SSO	CDFOX VS PLO	CDFOX VS mFOX
CEC01	2.261E-06	3.020E-11	8.993E-11	3.020E-11	3.020E-11	8.564E-04	3.020E-11	3.020E-11	1.335E-01	3.020E-11	8.303E-01
CEC02	3.012E-11	3.012E-11	3.012E-11	3.012E-11	3.012E-11	3.012E-11	3.012E-11	3.012E-11	3.012E-11	3.012E-11	3.012E-11
CEC03	2.990E-01	1.948E-11	1.948E-11	1.948E-11	1.948E-11	1.948E-11	1.948E-11	1.948E-11	1.948E-11	1.948E-11	1.726E-11
CEC04	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11
CEC05	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	4.077E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	2.416E-02
CEC06	6.669E-03	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	2.602E-08
CEC07	7.483E-02	3.020E-11	3.020E-11	4.077E-11	3.020E-11	6.356E-05	3.690E-11	3.690E-11	3.020E-11	3.020E-11	1.373E-01
CEC08	3.770E-04	3.020E-11	3.020E-11	3.020E-11	3.020E-11	2.959E-05	5.072E-10	5.072E-10	1.287E-09	8.153E-11	5.395E-01
CEC09	3.352E-08	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11	3.020E-11
CEC10	2.809E-02	3.014E-11	3.014E-11	3.014E-11	3.014E-11	8.473E-09	3.014E-11	3.014E-11	5.560E-10	3.014E-11	5.942E-02

Table 15: P-value for 23 classical benchmark test functions

Benchmark functions	CDFOX VS FOX	CDFOX VS TROA	CDFOX VS CHWA	CDFOX VS POA	CDFOX VS FATA	CDFOX VS PO	CDFOX VS AO	CDFOX VS SBOA	CDFOX VS SSO	CDFOX VS PLO	CDFOX VS mFOX
F1	NaN	1.21E-12	1.21E-12	1.24E-12	1.21E-12	1.66E-11	1.21E-12	1.21E-12	1.21E-12	1.20E-12	NaN
F2	NaN	1.21E-12	1.21E-12	1.24E-12	1.21E-12	1.66E-11	1.21E-12	1.21E-12	1.21E-12	1.21E-12	NaN
F3	NaN	1.21E-12	1.21E-12	1.24E-12	1.21E-12	4.57E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	NaN
F4	NaN	1.21E-12	1.21E-12	1.24E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	NaN
F5	3.02E-11	3.02E-11	3.02E-11	7.66E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	2.92E-11	3.02E-11	3.020E-11
F6	3.02E-11	3.02E-11	3.02E-11	7.65E-11	3.02E-11	3.02E-11	3.01E-11	3.02E-11	3.01E-11	3.02E-11	3.018E-11
F7	7.45E-01	3.02E-11	5.00E-09	8.32E-10	3.02E-11	3.34E-11	3.02E-11	3.02E-11	1.29E-09	3.02E-11	1.464E-10
F8	2.31E-02	2.11E-11	4.00E-01	7.26E-01	4.34E-01	1.30E-01	1.17E-07	1.05E-10	5.17E-01	1.76E-07	1.610E-02
F9	NaN	1.21E-12	1.21E-12	1.20E-10	1.21E-12	4.57E-12	1.21E-12	1.21E-12	NaN	1.21E-12	NaN
F10	NaN	1.21E-12	7.57E-13	1.09E-11	1.21E-12	4.57E-12	1.21E-12	1.19E-12	1.72E-10	1.19E-12	NaN
F11	NaN	1.21E-12	4.57E-12	1.21E-10	1.21E-12	1.93E-10	1.21E-12	1.21E-12	1.61E-01	1.21E-12	NaN
F12	3.02E-11	3.02E-11	3.02E-11	7.66E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.01E-11	3.02E-11	3.020E-11
F13	3.02E-11	3.02E-11	3.02E-11	7.64E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.018E-11
F14	1.24E-11	7.82E-12	2.67E-04	1.28E-06	4.85E-08	2.80E-09	1.71E-08	1.07E-11	2.19E-08	2.09E-10	6.329E-07
F15	4.64E-05	3.02E-11	3.02E-11	9.82E-10	1.46E-10	4.69E-08	6.07E-11	3.02E-11	5.97E-09	3.02E-11	2.006E-04
F16	1.21E-12	1.21E-12	1.21E-12	1.31E-11	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.665E-08
F17	1.21E-12	1.21E-12	1.21E-12	3.20E-09	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	2.783E-03
F18	1.21E-12	1.21E-12	1.21E-12	3.20E-09	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.656E-11
F19	2.17E-11	1.59E-11	1.59E-11	2.11E-11	1.59E-11	1.59E-11	1.59E-11	1.59E-11	1.59E-11	1.59E-11	2.949E-11
F20	8.49E-02	2.98E-11	2.84E-10	3.02E-10	4.45E-11	1.22E-01	4.29E-01	2.98E-11	2.98E-11	2.98E-11	3.973E-04
F21	1.21E-12	1.21E-12	1.21E-12	3.65E-13	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.211E-12
F22	1.21E-12	1.21E-12	1.21E-12	3.65E-13	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.212E-12
F23	1.21E-12	1.21E-12	1.21E-12	3.65E-13	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.212E-12

Table 16: Solving the design of pressure vessels

Pressure Vessel Design										
Algorithms	X1	X2	X3	X4	Statistical measure					
					Avg	Best	Worst	STD	Rank	P_value
CDFOX	1.23E+00	6.05E-01	6.36E+01	1.90E+01	6.47E+03	5.88E+03	8.35E+03	6.59E+02	1	
PO	8.48E-01	4.42E-01	4.13E+01	1.89E+02	6.85E+03	6.04E+03	7.79E+03	4.42E+02	3	3.67E-03
CHOA	1.28E+00	6.53E-01	6.58E+01	1.26E+01	7.89E+03	6.32E+03	8.48E+03	3.80E+02	4	4.18E-09
AO	1.23E+00	6.74E-01	6.14E+01	5.14E+01	8.30E+03	6.73E+03	1.20E+04	1.17E+03	5	9.26E-09
SSO	8.53E-01	5.75E-01	4.31E+01	1.92E+02	8.82E+03	7.35E+03	1.11E+04	9.84E+02	6	3.82E-10
FOX	1.85E+00	9.08E-01	9.56E+01	8.88E+01	1.39E+04	5.89E+03	3.54E+04	9.13E+03	7	1.25E-07
FATA	1.65E+00	1.65E+00	5.92E+01	5.92E+01	1.56E+04	7.84E+03	3.31E+04	3.68E+03	8	3.02E-11
POA	1.45E+00	1.07E+00	5.61E+01	2.00E+02	1.78E+04	7.84E+03	3.14E+04	5.88E+03	9	3.34E-11
PLO	2.52E+00	1.57E+00	4.14E+01	1.85E+02	2.35E+04	1.06E+04	3.66E+04	8.12E+03	10	3.02E-11
SBOA	4.86E+00	9.69E-01	1.51E+02	1.00E+01	2.82E+04	8.49E+03	7.09E+04	1.19E+04	11	3.02E-11
TROA	2.77E+00	8.04E-01	7.93E+01	6.71E+01	4.53E+04	1.37E+04	1.00E+05	1.89E+04	12	3.02E-11
mFOX	1.26075977	0.6205685	65.3203326	10.2604239	6.58E+03	5.95E+03	7.44E+03	444.7156	2	0.06145191

Table 17: Solving Geartrain Design.

Geartrain Design										
Algorithms	T_D	T_B	T_A	T_F	Statistical measure					
					Avg	Best	STD	worst	Rank	P_value
CDFOX	12.00024	12	16.89988	59.05861	9.77E-24	2.79E-28	6.70E-23	1.82E-23	1	
FOX	25.23491	13.53643	55.29957	42.81344	1.98E-15	1.17E-17	1.37E-14	3.29E-15	2	3.02E-11
CHOA	12	14.34499	19.88516	60	5.83E-11	2.54E-14	3.88E-10	9.41E-11	4	3.02E-11
PO	25.75047	20.16476	59.97993	60	2.03E-10	2.60E-15	3.56E-09	6.71E-10	5	3.02E-11
POA	31.36522	12	60	43.47908	9.17E-09	4.09E-12	6.72E-08	1.41E-08	6	3.02E-11
AO	14.16631	26.73641	48.63019	54.22221	6.96E-08	7.63E-13	5.91E-07	1.36E-07	7	3.02E-11
SSO	12	12	19.38056	51.55525	7.61E-08	2.45E-10	4.32E-07	1.17E-07	8	3.02E-11
PLO	12.10306	31.11677	53.26812	49.71877	4.71E-06	1.66E-09	4.65E-05	9.02E-06	9	3.02E-11
FATA	18.51171	15.45861	54.48496	35.66614	2.65E-04	1.09E-09	2.95E-03	6.41E-04	10	3.02E-11

SBOA	4.859572	0.968524	151.3935	10	2.08E-03	5.44E-08	3.56E-02	7.07E-03	11	3.02E-11
TROA	20.37942	59.89424	39.52383	12.61789	4.76E-03	4.50E-07	3.05E-02	7.28E-03	12	3.02E-11
mFOX	12	41.3105972	57.30503658	59.95781732	2.11E-14	1.01E-17	2.30E-13	4.39E-14	3	3.02E-11

Table 18: Solving Cantilever Beam Design.

Cantilever Beams											
Algorithms	X1	X2	X3	X4	X5	Statistical measure					
						Avg	Best	worst	STD	Rank	P_value
CDFOX	5.98E+00	5.31E+00	4.51E+00	3.52E+00	2.16E+00	1.34E+00	1.34E+00	1.34E+00	1.00E-04	1	
PO	6.20E+00	5.51E+00	4.26E+00	3.74E+00	1.92E+00	1.36E+00	1.34E+00	1.39E+00	1.25E-02	3	1.21E-12
CHOA	6.41E+00	5.50E+00	4.79E+00	3.00E+00	2.35E+00	1.36E+00	1.35E+00	1.39E+00	1.10E-02	4	1.21E-12
FOX	6.04E+00	5.29E+00	4.49E+00	3.50E+00	2.15E+00	1.38E+00	1.34E+00	2.62E+00	2.34E-01	5	1.00E+00
SSO	5.43E+00	8.01E+00	4.56E+00	3.17E+00	2.60E+00	1.51E+00	1.39E+00	1.65E+00	6.61E-02	6	1.21E-12
AO	4.86E+00	8.27E+00	8.35E+00	2.78E+00	3.13E+00	1.54E+00	1.35E+00	2.13E+00	1.87E-01	7	1.21E-12
FATA	6.24E+00	7.55E+00	8.43E+00	4.98E+00	4.65E+00	1.95E+00	1.53E+00	3.07E+00	4.16E-01	8	1.21E-12
POA	1.15E+01	2.10E+01	7.46E+00	3.45E+00	1.08E+01	7.05E+00	3.38E+00	9.21E+00	1.15E+00	9	1.21E-12
PLO	8.81E+00	1.98E+01	1.37E+01	4.19E+01	1.57E+01	7.38E+00	4.89E+00	1.10E+01	1.71E+00	10	1.21E-12
SBOA	2.42E+01	1.84E+01	7.77E+00	4.18E+00	2.14E+01	7.99E+00	4.46E+00	1.03E+01	1.56E+00	11	1.21E-12
TROA	3.87E+00	1.09E+01	2.11E+01	8.43E+01	8.77E+01	8.04E+00	4.60E+00	1.10E+01	1.53E+00	12	1.21E-12
mFOX	6.1626	5.3016	4.3393	3.4459	2.2528	1.3408	1.34	1.3434	8.84E-04	2	1.21E-12

4.1. CEC2017 benchmark test function

The main goal of this work is to confirm by extensive testing the efficiency of the CDFOX algorithm on highly complicated mathematical test functions. This focuses on CDFOX's ability to optimize handling hard optimization problems and prove its advantages in exploitation, exploration, and avoiding local optima. The CEC2017 benchmark set, which includes 30 functions categorized into four classes—unimodal (F1-F3), multimodal (F4-F11), hybrid (F12-F20), and composition functions (F21-F30)—has been chosen for this purpose. All these functions have been explained in a detailed manner in Table 3.

In Table 4, the results of comparing all of the algorithms on the CEC2017 benchmark are shown, respectively. There are many metrics that are used to evaluate each algorithm, including the average (AVG), the standard deviation (STD), the best, the worst, and the rank. Each participant's average performance is used to determine the ranking of participants.

CDFOX shows outstanding performance in solving problems of the CEC2017 benchmark set, proving itself one of the most competitive optimization algorithms. It takes the first place in solving a wide number of test problems: F1-F4, F7, F9-F12, F15-F25, F30, etc. This shows the dominance of the method in solving both unimodal and multimodal problems, proving its good balance of exploration and exploitation abilities. Besides, it takes the second place in solving problems F8, F13, F24, etc. It also shows good performance in solving more complicated problems, such as F23 and F28. Moreover, it shows stable convergence properties, as it solves problems repeatedly with high-quality optimized results. If compared with more established optimization algorithms, such as TROA, CHOA, SOA, etc., it shows better performance in solving problems, proving its effectiveness.

In contrast, although CFOX does not achieve Rank 1 in several test problems (e.g., F8, F13, F14, F17, F18, F19, F20, F22, F23, F24, F26, F27, and F29), CDFOX proves to be superior to other approaches. The primary reason for this performance is that CDFOX succeeds in making a balance between exploration and exploitation, which enables the algorithm to avoid premature convergence and effectively search the solution space. As a consequence, CDFOX becomes a rival to state-of-the-art approaches and, in many situations, exceeds their efficiency.

Figure 6 represents the convergence curves of the CDFOX algorithm on all 30 CEC2017 benchmark functions. These curves clearly depict the fast and stable algorithm of CDFOX towards optimal solutions. When compared with other algorithms, CDFOX has maintained a high rate of convergence with most of them and steadily reaches the optimum in most of the test problems. Such performance is an indication of the strength of the new CDFOX for solving tough optimization problems.

Figure 6 represents the convergence curves of the CDFOX algorithm on all 30 CEC2017 benchmark functions. These curves clearly depict the fast and stable algorithm of CDFOX towards optimal solutions. When compared with other algorithms, CDFOX has maintained a high

rate of convergence with most of them and steadily reaches the optimum in most of the test problems. Such performance is an indication of the strength of the new CDFOX for solving tough optimization problems. The rapid decline observed in several functions, such as F10, F11, F18, F25, and F29, and particularly in F4, F5, F7, F10, F19, F22, and F23, is mainly due to the landscape characteristics of these problems. These functions often contain relatively smooth regions or strong gradients near the global optimum. From the figure, CDFOX generally achieves lower objective values and faster convergence on many functions. This behavior is mainly due to the improvements introduced in the algorithm. First, the chaotic map mechanism enhances the exploration phase by generating diverse search patterns, which helps the algorithm explore the search space more effectively and avoid premature convergence. Second, the search-agent power strategy a^4 strengthens the movement capability of the agents, allowing them to adjust their positions more efficiently when approaching promising regions. Overall, the results demonstrate that the combination of chaotic exploration and the search-agent power mechanism gear train design balances exploration and exploitation, enabling CDFOX to achieve stable and competitive optimization performance across different types of benchmark functions.

Table 5 provides the processing times (in seconds) for CDFOX and other competing algorithms on each of the CEC 2017 test functions. Overall, the ranking of CDFOX, in terms of computational efficiency, is quite satisfactory. FATA secured the top position, whereas PLO secured the second position, and CDFOX occupied the tenth position. The performance comparison shows that CHWA performs the slowest, followed by SSO, and CDFOX is relatively efficient.

4.2. CEC2019 benchmark test function

In this experiment, the performance and competence of the CDFOX algorithm are tested and evaluated on a set of functions called CEC2019. In Table 6, there are ten functions with specific dimensions and search intervals that belong to CEC2019 [59]. The CEC2019 benchmark is a set of ten single-objective optimization functions that are far more complex than the classical functions discussed in this work. These are considerably harder problems with multilocal optima and, therefore, are quite suitable for testing an algorithm's precision, robustness, convergence rate, and local optima escaping capability.

CDFOX can effectively avoid the local minima and search the whole space for the best answers. Thus, it achieves great accuracy by avoiding entrapment in less-than-ideal regions due to a proper balance between exploration and exploitation by the random number. On every benchmark function in CEC02-CEC09, the performance of CDFOX is very good compared to other algorithms present in Table 7. This proves that CDFOX attained rank 1 across 8 benchmark functions, highlighting its excellent performance compared to the other algorithms. However, on CEC01 and CEC10, the ranks achieved by the CDFOX algorithm are 3rd and 4th, while FOX attained the first rank

on CEC01 and mFOX attained the first rank on CEC10. All these top-ranking results in a continuous manner highlight the strong capability of finding better solutions by CDFOX with a high degree of accuracy. It confirms that the proposed algorithm is strongly capable of optimization and robust across a wider spectrum of problem types. CDFOX outperforms eleven other benchmark algorithms on 8 test functions, and this is due to its good-quality exploration.

The convergence curves of the optimal run are presented in Figure 7. Additionally, enable the evaluation of the effectiveness of the CDFOX algorithm in the optimization of the entire set of benchmark functions presented in the CEC2019 set. The convergence curves prove the competitiveness and reliability of the CDFOX algorithm in terms of the rate of convergence compared to the performance of the other algorithms. The algorithm is moving forward in the optimal direction in the various test cases, and this is a reflection of the reliability of the algorithm in the handling of complex optimization problems.

The superior convergence of CDFOX on most CEC2019 functions can be attributed to the following three mechanisms. Firstly, the replacement of uniform random perturbations with the normal distribution-based exploration allows more targeted local search around promising regions, while at the same time enabling the possibility of large movements to avoid local optima. Secondly, the chaotic map initialization ensures the diversity of the search agents, thereby reducing the possibility of premature convergence. Thirdly, the power factor of 4, which scales the search performance, allows the algorithm to take larger steps when the search process is not progressing well, thus improving the exploration ability of the algorithm, while at the same time reducing the step sizes when better solutions are obtained, thus improving the exploitation ability of the algorithm.

The processing time in seconds for CDFOX and all compared algorithms for each CEC 2019 benchmark function is shown in Table 8. The overall rank for CDFOX based on total computation time for all ten functions is found to be 2, showing good computation efficiency for this function. It has outperformed 10 out of the total 11 compared algorithms, such as FOX, TROA, ChOA, POA, FATA, PO, AO, SBOA, and SSO, while being outperformed by only one, PLO, which has the first position in this table. The SSO function is the slowest, followed by SBOA, while CDFOX has shown better efficiency.

Table 9 shows a comparison of the performance of the CDFOX algorithm with other algorithms, namely FOX, IFOX, and mFOX, on the CEC 2019 benchmark functions. The results show that the CDFOX algorithm performs better than the other algorithms on most functions. Specifically, the CDFOX and IFOX algorithms obtain the best rank (Rank 1) for four functions, while the CDFOX algorithm attains the second rank (Rank 2) for the CEC02, CEC03, CEC07, and CEC08 functions, which shows the overall performance of the algorithm. The total rank shows that the CDFOX algorithm attains the lowest cumulative rank (18), while the IFOX [39], mFOX [20], and FOX algorithms attain a cumulative rank of 24, 28, and 30, respectively, which

shows the overall performance of the algorithms in solving the CEC 2019 functions.

4.3. Classical benchmark test function

Several conventional benchmark functions were used to assess the CDFOX algorithm's efficacy [50]. Unimodal and multimodal are the two main groups into which these functions fall. Unimodal functions (F1 to F7) with a single global optimum are particularly useful when assessing the exploitation capabilities of an optimization technique. On the other hand, multimodal functions (F7 to F13), which have several local optima, are more appropriate for assessing the algorithm's capacity for exploration [51]. A subclass of multimodal functions, fixed-dimension multimodal functions (F14 to F23) are useful for evaluating how well optimization methods balance exploration and exploitation, as they exhibit several local optima within a uniform search space, hence facilitating the evaluation of an algorithm's capacity to conduct extensive searches while honing effective solutions. More detail is shown in table 10.

Classical test functions were used to compare how well the proposed CDFOX algorithm worked with the algorithms in Table 10. The scores were averaged (AVG), and the standard deviation (STD), minimum value, maximum value, and ranking value were used to compare them. Table 10 lists the selected benchmark functions, classifying them as either unimodal or multimodal.

Table 11 makes it clear that the CDFOX algorithm consistently outperforms the other algorithms. CDFOX and the mFOX algorithm achieved first and second ranks, while original FOX predominantly secured third place across most unimodal functions (F1–F7). Specifically, they ranked first in four functions (F1, F2, F3, and F4), and CDFOX gets a superior rank in other functions (F6 and F7). The classical benchmark functions F1 to F7 are unimodal and mainly assess the exploitation capability of an algorithm. CDFOX demonstrates excellent performance.

Functions F8 to F13 are multimodal and pose exploration challenges due to numerous local optima. Among these functions, CDFOX ranked first in five (F9, F11, F12, and F13) and achieved second place in F10. CDFOX performs exceptionally well in these multimodal functions. These features make CDFOX highly effective for problems. CDFOX was the best in eight functions within the fixed-dimension multimodal group (F14–F23): F14–F19 and F21–F23. When compared to the original FOX optimizer, CDFOX performed better in 13 functions and equally well in seven functions (F1–F4, F9–F11, F16, F17, and F19).

In contrast, CDFOX did not achieve Rank 1 in several benchmark problems like F5, F17, and F20; CDFOX still gives an impressive performance. The results show that CDFOX has good convergence behavior and maintains a balance between exploration and exploitation.

Figure 8 presents the convergence curves of CDFOX against eleven other algorithms for benchmark functions F1–F23. Each curve shows the best objective value at each iteration. CDFOX tends to show good performance on most unimodal functions (F1–F7) with reasonable speed to reach good solution values. However, on high-dimensional

multimodal functions (F8–F13), its improvement trend is generally steady but not always the fastest. On fixed-dimensional multimodal functions, CDFOX tends to behave competitively and stably, even though several algorithms occasionally outperform it. The rapid decline observed in functions F1, F2, F3, F4, F5, F6, F16, F17, F18, F21, and F22 is mainly due to both the landscape characteristics of these problems and the design of CDFOX. The algorithm uses a search performance power mechanism, a^4 , which increases the agent's movement capability and accelerates position updates when a promising region is detected. In addition, the chaotic map improves exploration by generating diverse search patterns that help the algorithm quickly locate promising areas. In general, the results show that CDFOX is a consistent and competitive optimizer, as it reliably converges across different functions.

The processing time of CDFOX and some other rival algorithms using the classical benchmark test functions can be seen in Table 12 below. In general, the processing performance of CDFOX appears to have been fairly good. The top position was taken by PLO, followed by FATA in second place and CDFOX in tenth place. CHWA is revealed to be the slowest method, and SSO is the second slowest.

4.4. Statistical analysis

In this section, statistical testing is conducted using p-values to examine the performance of the algorithm with various benchmark functions such as classical benchmark test functions, CEC2017, and CEC2019.

4.4.1. CEC2017 benchmark test function

In Table 13, it can be found that the Wilcoxon rank-sum test comparison of CDFOX with eleven different algorithms on 30 test functions of CEC2017 shows that most of the p-value outcomes were found to be statistically significant with $p\text{-value} < 0.05$. CDFOX performed excellently on the 30 test functions. It obtained better p-values for all functions when compared with TROA, CHOA, FATA, SBOA, and PLO. CDFOX did not achieve better p-values for a few functions against the remaining algorithms, as highlighted in Table 13. When compared with FOX, CDFOX outperformed it on all functions except F1, F5, F14, F19, and F28. Against POA, CDFOX was better on all functions, except F14, F21, F27, and F30. Compared with PO, CDFOX outperformed it on all functions except F6, F16, F20, F23, F24, F26, F27, F28, and F29. Against AO, CDFOX achieved better p-values on all functions except F15, F16, F17, F21, and F23. Finally, compared with SSO, CDFOX performed better on all functions except F5, F14, F20, and F21.

4.4.2. CEC2019 benchmark test function

Table 14 shows the results of the rank-sum Wilcoxon test to identify differences in performances among other algorithms and the CDFOX algorithm on 10 test CEC2019 functions. The differences were found to be statistically significant because of low values of $p < 0.05$ if considering differences between CDFOX and other algorithms. On

every test function, it is evident that CDFOX performs better than the FOX algorithm. Table 14 demonstrates that CDFOX greatly outperforms other algorithms based on a p-value comparison. It can easily be determined from Table 14 that CDFOX performs significantly better than other algorithms as far as performance evaluation is concerned.

4.4.3. Classical benchmark test function

The rank-sum The Wilcoxon test was employed to statistically analyze the differences in performances among different optimization techniques; if the p-value is ≤ 0.05 , it reveals significant differences and ensures that these differences in performances were not due to random phenomena but rather because of actual algorithm performances. It can be revealed from Table 15 that for more than half of the CDFOX techniques with other techniques on 23 classical benchmark problems, the p-values were well below 0.05, ensuring that CDFOX has statistically significant superior or distinctive performances. It can be inferred that CDFOX faced non-significant differences only on F1–F4, F9, and F20 with FOX; however, it performed better on other functions. It can similarly be inferred that CDFOX faced non-significant differences only on F8 with CHWA, POA, and FATA; however, it performed significantly superior on other functions. In comparison with PO, CDFOX faced non-significant differences only on F8 and F20; however, it performed significantly superior on other functions. In comparison with AO, CDFOX faced non-significant differences only on F20; however, it performed significantly superior on other functions. In comparison with SSO, CDFOX faced non-significant differences only on F7 and F9; however, it performed significantly superior on other functions. For other optimization algorithms, CDFOX produced promising p-values for all test problems, thereby ensuring the efficacy of CDFOX over other algorithms with utmost significance. It can thus be concluded that when compared with eleven prominent algorithms of optimization, CDFOX performed significantly superior to all test problems except for those with p-values more than 0.05, which are marked with asterisks.

4.4.4. Limitation of CDFOX

CDFOX cannot achieve Rank 1 in several functions within the CEC2017, CEC2019, and classical benchmark functions, suggesting that the algorithm has certain performance weaknesses due to particular function characteristics. In the case of CEC2017, which includes functions such as F5, F6, F13, F14, F17–F24, and F26–F29, the functions are usually multimodal, non-convex, continuous, and shifted with additional properties, namely hybridity and compositionality. The nature of these functions causes non-separable search landscapes containing multiple local optima. In the case of CEC2019, including CEC01 and CEC10, there are functions with non-convex and shifted properties (CEC01) and more complex multimodal, hybrid, and composite characteristics (CEC10). These properties lead to inconsistent optimization results. Even in classical benchmark functions, CDFOX shows weak performance:

F2 is unimodal and non-convex, F8 is multimodal and non-convex, and F20 is multimodal, non-convex, and composite. This shows that CDFOX struggles with composite properties, even in relatively simple problems.

Generally, CDFOX has obvious weaknesses when optimizing multimodal, non-convex, shifted, hybrid, and composite functions, whose properties cause complex search landscapes, and in some cases also struggles with unimodal functions, as these characteristics create complex or deceptive search landscapes.

4.5. Real-world application

Three engineering design problems are covered in this section with real scenarios that will be chosen whereby the efficiency of the CDFOX algorithm can be tested. These problems cover the design of gear trains, pressure vessels, and cantilever beam design problems. For comparison purposes, the CDFOX algorithm will be tested among eleven other well-known metaheuristics. Every algorithm will run separately for 30 trials, considering stochastic processes with population dimensions of 30 and maximum numbers of 500 iterations per trial. The efficiency of every algorithm will then be measured using statistical measures like mean, standard deviation, best solution value, worst solution value, and P-value of objective value.

However, the FOX algorithm has a number of limitations, including the fact that the algorithm converges prematurely, which may cause suboptimal solutions; the algorithm is sensitive to the parameters chosen for the algorithm, which may cause instability; the algorithm does not adequately explore the search space; and the algorithm gets stuck in local optimal solutions. The disadvantages limit the performance and efficiency of the algorithm for use in real-life applications.

However, the CDFOX algorithm has been able to improve the disadvantages of the FOX algorithm by introducing the following strategies. Firstly, the chaotic map initialization strategy promotes diversity for the population. Secondly, the power strategy for the search agents (a^4), also known as a^4 strategy, enhances the movement strategy for the agents. Thirdly, the use of the Chebyshev distance strategy enhances the convergence accuracy. With the improvement strategies for the disadvantages of the FOX algorithm, the CDFOX algorithm has been able to improve the performance and efficiency of the algorithm for use in solving complex real-life problems compared to the original FOX algorithm.

4.5.1. Solving the Pressure Vessel Design Problem

The traditional engineering problem of pressure vessel design has been solved by several researchers using different methods [18]. The main objective is to ensure that a cylindrical pressure vessel meets specific safety and pressure standards while minimizing the total construction cost. The aim is to reduce manufacturing expenses without compromising safety or performance requirements. The design depends on four critical variables: the cylindrical section's length (L), head thickness (T_h), inner radius (R), and shell

thickness (T_s). Figure 9 is the pressure vessel design problem, and Eq. 17 sets its objective function. Table 16 lists the ranges of the design variables. CDFOX performs best and attains the highest mean performance in this very constrained engineering design problem. According to a statistical study, CDFOX achieves the greatest results across important performance criteria, outperforming all rival algorithms.

$$\text{Let } \bar{x} = [X_1 X_2 X_3 X_4] = [T_s T_h R L] \quad (17)$$

minimize: $f(\bar{x})$

$$= 0.6224 x_1 x_2 x_3 + 1.7781 x_2 x_3^2 + 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3,$$

$$\begin{aligned} \text{the variable limits are} \quad & 0 \leq x_1 \leq 99 \\ & 0 \leq x_2 \leq 9 \\ & 0 \leq x_3 \leq 200 \end{aligned}$$

They are subject to the following constraint:

$$g_1(\bar{x}) = -x_1 + 0.0193x_3 \leq 0$$

$$g_2(\bar{x}) = -x_3 + 0.00954x_3 \leq 0$$

$$g_3(\bar{x}) = -\pi x_3^2 x_4 - \frac{4}{3} \pi x_3^3 + 1,296,000 \leq 0$$

$$g_4(\bar{x}) = x_4 + 240 \leq 0$$

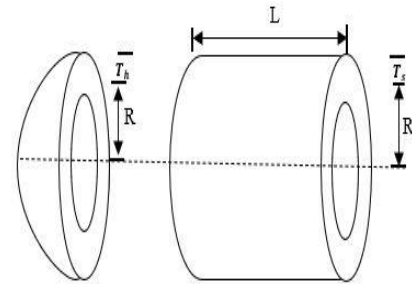


Figure 9: pressure vessel design problem.

4.5.2. Solving the Geartrain Design Problem

Designing gear trains still remains one of the greatly preferred and popular problems solved with the aid of mechanical engineering theories [51]. The basic concept of solving this problem is to identify optimal values of gears corresponding to which there would exist a minimal ratio of faster and slower rotation speeds of output and input sides, respectively. There exist four basic design variables, T_A , T_B , T_D , and T_F , which should be optimized; these symbols introduce the number of teeth of four interrelated gears. These variables should not lose their meaning and should always keep positive integer values ranging from 12 to 60 with specific basic side constraints applied considering real mechanical functionalities. The mathematical model of the gear train system used while performing optimizations was given in Eq. 18, and how these gears were related was shown in Figure 10. For testing the performance of the proposed work, Table 17 was used to introduce an extensive comparison between the CDFOX algorithm and eleven different proven metaheuristic algorithms, which revealed that CDFOX efficiently performed well when compared with other optimized algorithms, ranking at the top position multiple times.

$$\text{Let } \bar{x} = [X_1 X_2 X_3 X_4] = [T_D T_B T_A T_F],$$

$$\text{minimize: } f(\bar{x}) = \left(\left(\frac{1}{6.931} \right) \right) - \left(\left(\frac{x_2 x_3}{x_1 x_4} \right) \right)^2 \quad (18)$$

The variable range: $2 \leq x_i \leq 60, i = 1, 2$

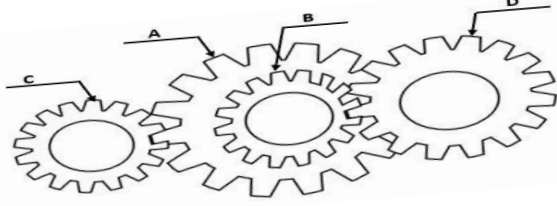


Figure 10: Geartrain Design problem.

4.5.3. Solving the Design of the Cantilever Beam Problem

The goal of the well-known structural optimization problem in engineering known as Cantilever Beam Design (CBD) is to reduce the weight of a cantilever beam composed of five hollow rectangular blocks [18]. Each of the five design variables represents the thickness of a specific beam segment, as illustrated in Figure 11. The objective, defined in Eq. 19, is to minimize the weight of the beam without affecting its integrity when the loads are imposed. Table 18 presents result that are compared with those from eleven other metaheuristic algorithms. Statistical evaluation confirms CDFOX's superior performance, achieving Rank 1 for reliability, consistency, and optimization quality, making it a strong solution for real-world structural design problems.

$$\text{onsider } \bar{x} = [X_1 X_2 X_3 X_4 X_5] = [T_s T_h RL],$$

$$\text{minimize: } f(\bar{x})$$

$$= 0.6224 (x_1 + x_2 + x_3 + x_4), \quad (19)$$

They are subject to:

$$G(\bar{x}) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0$$

The variable range: $0.01 \leq x_i \leq 1.00, i \in 1 \dots$

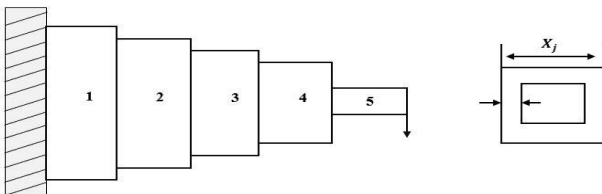


Figure 11: Cantilever Beam Design Problem

5. Conclusion

Modified CDFOX was proposed by adjusting the original FOX optimizer. CDFOX's primary enhancements are significantly improving the ability of exploitation and exploration, leading to more efficient and stable optimization performance. Additionally grasp its performance fully, CDFOX was experimented with on 63 benchmark functions, including 23 basic test functions, 10 CEC2019

functions, and 30 CEC2017 functions. In addition, the algorithm was used for solving three engineering design problems in real life: pressure vessel and gear train design problems and cantilever beam design. Experiment results clearly demonstrate that CDFOX significantly dominates a wide array of state-of-the-art optimization algorithms: FOX, TROA, ChOA, POA, FATA, PO, AO, SSO, SBOA, and PLO. Its superior performance on different problem sets makes CDFOX a powerful and effective optimization device.

In the future, several lines of research are very promising to further push the capability of CDFOX. One of them is to derive CDFOX variants with the addition of discretization techniques and multi-objective optimization strategies such that their applicability can be extended to more general classes of problems. In addition. Future research would further explore the modification of the internal operation of CDFOX or the blending of CDFOX with other heuristics and metaheuristics to improve performance further, particularly in dynamic or high-dimensional environments, and improve CDFOX's exploration–exploitation balance for multimodal, non-convex, shifted, hybrid, and composite functions by using adaptive parameters, diversity strategies, and hybridization to better escape local optima.

Funding source

None

Conflict 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.

Competing interests

The authors declare that there are no competing interests.

Ethical and informed consent for data used

Not applicable.

Data Availability

The datasets used and/or analyzed during the current study are publicly available benchmark functions from the CEC2017 and classical benchmark test function and CEC2019 competition suites. All experimental results generated during this study are included in this published article.

6. References

- [1] K. SM, "Cuckoo Search and Firefly Algorithms-Biologically Inspired Algorithms," *Open Access Biostat. Bioinforma.*, vol. 1, pp. 1–4, 2018, doi: 10.31031/oabb.2018.01.000514.
- [2] S. Zhang, "Real-life Application of Optimization Problem," *Theor. Nat. Sci.*, vol. 5, no. 1, pp. 53–57, 2023, doi: 10.54254/2753-8818/5/20230271.

- [3] D. Liñán Romero, G. Contreras-Zarazua, E. Sánchez Ramírez, J. Segovia-Hernandez, and L. Ricardez-Sandoval, "A Hybrid Deterministic-Stochastic Algorithm for the Optimal Design of Process Flowsheets with Ordered Discrete Decisions," *Comput. Chem. Eng.*, vol. 180, p. 108501, Nov. 2023, doi: 10.1016/j.compchemeng.2023.108501.
- [4] W. G. Alshanti, I. M. Batiha, M. A. Hammad, and R. Khalil, "A novel analytical approach for solving partial differential equations via a tensor product theory of Banach spaces," *Partial Differ. Equations Appl. Math.*, vol. 8, no. June, p. 100531, 2023, doi: 10.1016/j.padiff.2023.100531.
- [5] M. Dehghani, E. Trojovská, and P. Trojovský, "A new human-based metaheuristic algorithm for solving optimization problems on the base of simulation of driving training process," *Sci. Rep.*, vol. 12, no. 1, pp. 1–21, 2022, doi: 10.1038/s41598-022-14225-7.
- [6] Z. Marszałek, "Parallelization of modified merge sort algorithm," *Symmetry (Basel)*, vol. 9, no. 9, 2017, doi: 10.3390/sym9090176.
- [7] S. Saha, M. K. Bhaumik, and S. Das, "A New Modified Linear Search Algorithm," *Int. J. Math. Trends Technol.*, vol. 65, no. 12, pp. 148–152, 2019, doi: 10.14445/22315373/ijmtt-v65i12p516.
- [8] Z. Montazeri, T. Niknam, J. Aghaei, O. P. Malik, M. Dehghani, and G. Dhiman, "Golf Optimization Algorithm: A New Game-Based Metaheuristic Algorithm and Its Application to Energy Commitment Problem Considering Resilience," *Biomimetics*, vol. 8, no. 5, 2023, doi: 10.3390/biomimetics8050386.
- [9] I. A. Falahah *et al.*, "Frilled Lizard Optimization: A Novel Bio-Inspired Optimizer for Solving Engineering Applications," *Comput. Mater. Contin.*, vol. 79, no. 3, pp. 3631–3678, 2024, doi: 10.32604/cmc.2024.053189.
- [10] C. Audet and W. Hare, "Genetic Algorithms," *Springer Ser. Oper. Res. Financ. Eng.*, pp. 57–73, 2017, doi: 10.1007/978-3-319-68913-5_4.
- [11] A. Slowik, "Particle Swarm Optimization," *Ind. Electron. Handb. - Five Vol. Set*, no. May 2011, 2011, doi: 10.1007/978-3-319-46173-1_2.
- [12] R. S. Prado, R. C. P. Silva, F. G. Guimarães, and O. M. Neto, "A New Differential Evolution Based Metaheuristic for Discrete Optimization," *Int. J. Nat. Comput. Res.*, vol. 1, no. 2, pp. 15–32, 2010, doi: 10.4018/jncr.2010040102.
- [13] E. Trojovska, M. Dehghani, and P. Trojovsky, "Zebra Optimization Algorithm: A New Bio-Inspired Optimization Algorithm for Solving Optimization Algorithm," *IEEE Access*, vol. 10, pp. 49445–49473, 2022, doi: 10.1109/ACCESS.2022.3172789.
- [14] M. C. Su, J. H. Chen, A. M. Utami, S. C. Lin, and H. H. Wei, "Dove Swarm Optimization Algorithm," *IEEE Access*, vol. 10, pp. 46690–46696, 2022, doi: 10.1109/ACCESS.2022.3170112.
- [15] T. S. L. V. Ayyarao *et al.*, "War Strategy Optimization Algorithm: A New Effective Metaheuristic Algorithm for Global Optimization," *IEEE Access*, vol. 10, pp. 25073–25105, 2022, doi: 10.1109/ACCESS.2022.3153493.
- [16] O. N. Oyelade, A. E. S. Ezugwu, T. I. A. Mohamed, and L. Abualigah, "Ebola Optimization Search Algorithm: A New Nature-Inspired Metaheuristic Optimization Algorithm," *IEEE Access*, vol. 10, pp. 16150–16177, 2022, doi: 10.1109/ACCESS.2022.3147821.
- [17] J. Tang, G. Liu, and Q. Pan, "A Review on Representative Swarm Intelligence Algorithms for Solving Optimization Problems: Applications and Trends," *IEEE/CAA J. Autom. Sin.*, vol. 8, pp. 1627–1643, Oct. 2021, doi: 10.1109/JAS.2021.1004129.
- [18] Z. K. A. Dler Hasan, Hardi M. Mohammed, "Modified FOX Optimizer for Solving optimization problems," 2025, [Online]. Available: doi:10.48550/arXiv.2502.20027
- [19] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft Comput.*, vol. 22, no. 2, pp. 387–408, 2018, doi: 10.1007/s00500-016-2474-6.
- [20] H. Givi and M. Hubalovska, "Skill Optimization Algorithm: A New Human-Based Metaheuristic Technique," *Comput. Mater. Contin.*, vol. 74, no. 1, pp. 179–202, 2023, doi: 10.32604/cmc.2023.030379.
- [21] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems," *CAD Comput. Aided Des.*, vol. 43, no. 3, pp. 303–315, 2011, doi: 10.1016/j.cad.2010.12.015.
- [22] L. Sharma, V. L. Chellapilla, and P. Chellapilla, "Socio-inspired evolutionary algorithms: a unified framework and survey," *Soft Comput.*, vol. 27, no. 19, pp. 14127–14156, 2023, doi: 10.1007/s00500-023-07929-z.
- [23] B. A. Umit Canl, *, "Physics Based Metaheuristic Algorithms for Global Optimization," 2015, doi: http://creativecommons.org/licenses/by-nc/4.0/.
- [24] A. Kaveh and T. Bakhshpoori, "Metaheuristics: Outlines, MATLAB Codes and Examples," *Metaheuristics: Outlines, MATLAB Codes and Examples*, pp. 1–190, 2019, doi: 10.1007/978-3-030-04067-3.
- [25] H. Eskandar, A. Sadollah, A. Bahreinejad, and M. Hamdi, "Water cycle algorithm - A novel metaheuristic optimization method for solving constrained engineering optimization problems," *Comput. Struct.*, vol. 110–111, pp. 151–166, 2012, doi: 10.1016/j.compstruc.2012.07.010.
- [26] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, 2017, doi: 10.1016/j.advengsoft.2017.07.002.
- [27] J. Lian *et al.*, "Parrot optimizer: Algorithm and applications to medical problems," *Comput. Biol. Med.*, vol. 172, 2024, doi: 10.1016/j.combiomed.2024.108064.
- [28] Y. Fang, S. Na, M. W. Mahoney, and M. Kolar, "Fully Stochastic Trust-Region Sequential Quadratic Programming for Equality-Constrained Optimization Problems," *SIAM J. Optim.*, vol. 34, no. 2, pp. 2007–2037, 2024, doi: 10.1137/22M1537862.
- [29] T. Strutyńska, "Social and economic problems of regional TV news in the conditions of the Russian-Ukrainian War (on the example of the TV channel '3-Studio' in Ivano-Frankivsk)," *Commun. Commun. Technol.*, vol. 1813, no. 24, pp. 48–56, 2024, doi: 10.15421/292406.
- [30] G. ATALI, I. PEHLIVAN, B. GÜREVİN, and H. I. SEKER, "Chaos in metaheuristic based artificial intelligence algorithms: A short review," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 29, no. 3, pp. 1354–1367, 2021, doi: 10.3906/elk-2102-5.
- [31] X. L. Y. S. Ke Xue*, "Escaping Local Optima in Global Placement", doi: https://arxiv.org/pdf/2402.18311.
- [32] M. Banaie-Dezfouli, M. H. Nadimi-Shahraki, and Z. Beheshti, "R-GWO: Representative-based grey wolf optimizer for solving engineering problems," *Appl. Soft Comput.*, vol. 106, p. 107328, 2021, doi: 10.1016/j.asoc.2021.107328.

- [33] T. A. R. *Hardi M. Mohammed1, “FOX: A Fox-inspired Optimization Algorithm,” 2022, [Online]. Available: <https://doi.org/10.21203/rs.3.rs-1939478/v1>
- [34] G. Hu, K. Song, X. Li, and Y. Wang, “DEMFFA: a multi-strategy modified Fennec Fox algorithm with mixed improved differential evolutionary variation strategies,” *J. Big Data*, vol. 11, no. 1, 2024, doi: 10.1186/s40537-024-00917-6.
- [35] Z. Fu, J. An, Q. Yang, H. Yuan, Y. Sun, and H. Ebrahimiyan, “Skin cancer detection using Kernel Fuzzy C-means and Developed Red Fox Optimization algorithm,” *Biomed. Signal Process. Control*, vol. 71, p. 103160, 2022, doi: <https://doi.org/10.1016/j.bspc.2021.103160>.
- [36] Z. Zhang, X. Wang, and L. Cao, “FOX Optimization Algorithm Based on Adaptive Spiral Flight and Multi-Strategy Fusion,” *Biomimetics*, vol. 9, no. 9, p. 524, 2024, doi: 10.3390/biomimetics9090524.
- [37] M. A. Jumaah, Y. H. Ali, and T. A. Rashid, “Improved FOX Optimization Algorithm,” pp. 1–34, 2025, [Online]. Available: <http://arxiv.org/abs/2504.09574>
- [38] A. K. Fedaa, 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, p. e24192, 2024, doi: 10.1016/j.heliyon.2024.e24192.
- [39] D. P. Nikezić, D. S. Radivojević, N. S. Mirkov, I. M. Lazović, and T. A. Miljočić, “Symmetric U-Net Model Tuned by FOX Metaheuristic Algorithm for Global Prediction of High Aerosol Concentrations,” 2024.
- [40] Z. Huo, S. Liu, and H. Ebrahimiyan, “Aircraft Energy Management System Using Chaos Red Fox Optimization Algorithm,” *J. Electr. Eng. Technol.*, vol. 17, Aug. 2021, doi: 10.1007/s42835-021-00884-5.
- [41] L. Xu and M. Mohammadi, “Brain tumor diagnosis from MRI based on Mobilenetv2 optimized by contracted fox optimization algorithm,” *Heliyon*, vol. 10, no. 1, 2024, doi: 10.1016/j.heliyon.2023.e23866.
- [42] T. T. & A. F. K. Fahrettin Burak Demir, “A chaotic optimization method based on logistic-sine map for numerical function optimization,” pp. 14227–14239, 2022, [Online]. Available: <https://doi.org/10.1007/s00521-020-04815-9>
- [43] A. E. Haryati, “Implementation of Minkowski-Chebyshev Distance in Fuzzy Subtractive Clustering,” vol. 2, no. 2, pp. 82–87, 2021, doi: 10.20885/EKSAKTA.vol2.iss2.art1.
- [44] V. S. D. M. Sahu, P. Samal, and C. K. Panigrahi, “Tyrannosaurus optimization algorithm: A new nature-inspired meta-heuristic algorithm for solving optimal control problems,” *e-Prime - Adv. Electr. Eng. Electron. Energy*, vol. 5, no. August, p. 100243, 2023, doi: 10.1016/j.prime.2023.100243.
- [45] O. Al-Baik *et al.*, “Pufferfish Optimization Algorithm: A New Bio-Inspired Metaheuristic Algorithm for Solving Optimization Problems,” *Biomimetics*, vol. 9, no. 2, 2024, doi: 10.3390/biomimetics9020065.
- [46] A. Qi, D. Zhao, A. A. Heidari, L. Liu, Y. Chen, and H. Chen, “FATA: An efficient optimization method based on geophysics,” *Neurocomputing*, vol. 607, 2024, doi: 10.1016/j.neucom.2024.128289.
- [47] C. Yuan *et al.*, “Artemisinin optimization based on malaria therapy: Algorithm and applications to medical image segmentation,” *Displays*, vol. 84, no. May, 2024, doi: 10.1016/j.displa.2024.102740.
- [48] H. A. Shehadeh, I. Ahmedy, and M. Y. I. Idris, “Sperm swarm optimization algorithm for optimizing wireless sensor network challenges,” *ACM Int. Conf. Proceeding Ser.*, no. February, pp. 53–59, 2018, doi: 10.1145/3193092.3193100.
- [49] Y. Fu, D. Liu, J. Chen, and L. He, *Secretary bird optimization algorithm: a new metaheuristic for solving global optimization problems*, vol. 57, no. 5. Springer Netherlands, 2024. doi: 10.1007/s10462-024-10729-y.
- [50] K. K. Mishra, S. Tiwari, and A. K. Misra, “A bio inspired algorithm for solving optimization problems,” *2011 2nd Int. Conf. Comput. Commun. Technol. ICCCT-2011*, pp. 653–659, 2011, doi: 10.1109/ICCCT.2011.6075211.
- [51] P. Sarangi and P. Mohapatra, “Chaotic-Based Mountain Gazelle Optimizer for Solving Optimization Problems,” *Int. J. Comput. Intell. Syst.*, vol. 17, no. 1, 2024, doi: 10.1007/s44196-024-00444-5.