Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.080569

ARTICLE

Multistrategy Improved Aquila Optimizer for Test Case Prioritization

Jiali Chen[1,2,3], **Jiheng Zhang**[1,2,3], **Xiaojie Chen**[1,2,3], **Chong Zeng**[1,2,3], **Honghui Yi**[1,2,3] **and Heming Jia**[4,*]

[1]School of Mathematics and Information Engineering, Longyan University, Longyan, China
[2]Fujian Provincial University Key Laboratory of Big Data Mining and Applications, Longyan University, Longyan, China
[3]Guo Boling Academician Workstation of Longyan University, Longyan, China
[4]School of Information Engineering, Sanming University, Sanming, China

*Corresponding Author: Heming Jia. Email: jiaheming@fjsmu.edu.cn

Received: 12 February 2026; Accepted: 11 May 2026; Published: 15 June 2026

ABSTRACT: Traditional heuristic algorithms often fall into local optima and converge slowly when test case prioritization is addressed in regression testing, making them inadequate for complex real-world scenarios. The Aquila optimizer, a novel metaheuristic algorithm, demonstrates strong global exploration capability but still faces limitations, including insufficient exploitation capability and slow convergence. To overcome these challenges, a multi-strategy improved chaotic Cauchy inverse cumulative distribution Aquila optimizer for test case prioritization is proposed. First, a logistic–sine–cosine composite chaotic mapping is introduced during the initialization phase of the Aquila optimizer to increase population diversity. Second, the mutated random walk strategy is used to improve global exploration, further enhancing the global search ability of the Aquila optimizer. Moreover, during the narrowed exploration and narrowed exploitation phases, the Cauchy inverse cumulative distribution flight replaces the Lévy flight strategy to reallocate individual positions, strengthening individuals' optimization capability and preventing the algorithm from becoming trapped in local optima. Finally, in the later iteration stage, the specular reflection learning strategy is used to perturb the optimal individual positions and improve the Aquila optimizer's convergence accuracy and comprehensive optimization performance. Five Java projects were selected from the Defects4J benchmark datasets to conduct comparative experiments with the Aquila optimizer and seven other metaheuristic algorithms. The results demonstrate the effectiveness and superiority of the improved algorithm in test case prioritization. It achieves average improvements of approximately 4.96% in the average percentage of fault detection, 3.82% in the average percentage of block coverage, and 5.64% in the average percentage of decision coverage, enabling faster coverage of code blocks and branches. The results provide an efficient priority sorting solution for complex regression testing scenarios.

KEYWORDS: Heuristic algorithm; search-based software engineering (SBSE); Aquila optimizer (AO); test case prioritization (TCP); average percentage of fault detection (APFD); average percentage of block coverage (APBC); average percentage of decision coverage (APDC)

1 Introduction

In software engineering, regression testing serves as a critical phase in which it is ensured that newly added features and code modifications do not introduce unexpected errors [1]. It is a vital tool for software quality control and assurance and plays a pivotal role in software development. With the rapid advancement of the software industry, agile development and continuous integration have become mainstream paradigms. The rapid iteration of agile methodologies and the fast feedback loop of continuous integration have resulted in frequent changes to software requirements [2,3]. The high frequency of changes and integrations has led to

exponential growth in the scale of regression testing, with both execution frequency and costs skyrocketing, which has made fully executing all the test cases within limited timeframes increasingly difficult, thus creating a sharp contradiction between testing sufficiency and execution efficiency [4]. Maximizing the value of regression testing within limited time constraints has become a significant challenge in the field.

TCP [5] is a method that involves rearranging test case execution sequences without compromising defect detection capabilities. It maximizes testing activity benefits within limited time windows, which makes it a research hotspot that is currently attracting attention from both academia and industry [6].

In recent years, under the search-based software engineering (SBSE) paradigm, various heuristic and metaheuristic algorithms have been widely adopted to solve TCP problems [7]. From the early genetic algorithm (GA) [8], ant colony optimization (ACO) [9], and particle swarm optimization (PSO) [10] to recent novel biomimetic optimization algorithms, these approaches model TCP as a combinatorial optimization problem. By emulating natural evolution or swarm intelligence (SI), they optimize fitness functions to identify near-optimal solutions within vast test case sequences, thereby achieving a search performance that significantly surpasses that of traditional random sorting methods [11].

Although these studies have achieved certain results, the increasing complexity of application scenarios and deeper research have revealed the limitations of existing heuristic-based TCP methods in practice. First, traditional algorithms often involve blind search processes that lack targeted guidance, which can easily lead to local optima and slow convergence. Second, algorithm performance typically depends on key parameter sets, and finding optimal parameter combinations for specific projects is inherently a time-consuming process [12]. Third, most existing metaheuristic algorithms employ fixed search patterns that cannot adaptively balance exploration and exploitation across different optimization phases, limiting their effectiveness when applied to the diverse and complex nature of real-world software projects. Therefore, exploring and introducing newer and more efficient metaheuristic algorithms for TCP problems, along with targeted improvements, has become a research direction of significant theoretical and practical value [7].

The AO is a metaheuristic inspired by the hunting behavior of aquila birds [13] and has demonstrated competitive performance in multiple optimization domains, including oil production forecasting [14], image classification [15], and power system load frequency control [16], owing to its robust global exploration capability, simple parameter structure, and rapid convergence. This motivates its potential for addressing the limitations of existing metaheuristic-based TCP techniques. However, when directly applied to TCP, the original AO exhibits four key limitations: (1) the inherent mismatch between its continuous search space and the discrete permutation space of TCP; (2) low initialization diversity due to random population generation; (3) premature convergence during the narrowed exploration and exploitation phases, caused by the limitations of Lévy flight; and (4) insufficient convergence accuracy in later iterations due to the lack of effective local refinement mechanisms.

These shortcomings motivate the proposed MCIAO_TCP framework, which tailors AO for TCP through a multi-strategy enhancement. The main contributions of this study are threefold:

(1) We propose a multi-strategy improved chaotic Cauchy inverse cumulative distribution aquila optimizer (MCIAO) for TCP. The framework integrates four complementary strategies: logistic–sine–cosine composite chaotic mapping to increase initialization diversity; a mutated random walk strategy to enhance global exploration; Cauchy inverse cumulative distribution flight to replace Lévy flight for adaptive position updates; and a specular reflection learning strategy to perturb optimal individuals in later iterations for local refinement.

(2) By integrating these strategies into the optimization process, this work offers a practical and effective approach to maintaining search balance and preventing premature stagnation in TCP, a persistent challenge in SBSE, thereby advancing the application of metaheuristics in real-world testing scenarios.
(3) Comprehensive experiments on five Java projects from the Defects4J benchmark demonstrate that MCIAO achieves faster fault detection, providing an effective and reliable solution for TCP in complex regression testing scenarios.

The key innovation of MCIAO_TCP lies in the phase-aware integration of four enhancement strategies, each specifically tailored to address the limitations of the original AO at different optimization stages. While these individual strategies are established techniques, their synergistic combination creates a complementary framework where each component targets a distinct weakness of the base algorithm.

The remainder of this paper is organized as follows: Section 2 reviews related literature and critically discusses the challenges inherent in existing methods. Section 3 details the proposed MCIAO_TCP framework. Section 4 describes the experimental configuration and benchmark datasets. Section 5 presents a comprehensive analysis of the experimental results. Finally, Section 6 concludes the paper and outlines directions for future work.

2 Related Work

2.1 Definition of TCP

TCP is a methodology that reorders test cases on the basis of predefined testing objectives and specific criteria and optimizes their execution sequence to increase testing efficiency and reduce workload. This approach aims to refine the test case set in regression testing, thereby improving its effectiveness. TCP has emerged as a key research focus in software testing [7].

The TCP problem is defined as follows: Given a test suite T, its complete permutation set PT, and a sorting objective function f: PT→R, the goal is to find $T' \in PT$ such that $\forall T'' \in PT(T'' \neq T')$ and $f(T'') \leq f(T')$ [5]. This indicates that the objective function f takes an execution sequence of test cases as input and outputs a numerical value between 0 and 100. The sorting performance is determined by this value, with higher values indicating better results.

Compared with random-order testing, TCP demonstrates superior error detection speed. To evaluate the effectiveness of TCP, the sequencing results need to be assessed. In this study, three evaluation metrics for the objective function f are adopted: the average percentage of fault detection (APFD), the average percentage of block coverage (APBC), and the average percentage of decision coverage (APDC).

APFD [5] is a weighted average of the failure percentages detected throughout the test cycle and is calculated as follows:

$$APFD = 1 - \frac{TF_1 + TF_2 + \ldots + TF_m}{n * m} + \frac{1}{2n} \tag{1}$$

where n represents the total number of test cases in the test suite T and m represents the number of defects. When a specific test case execution sequence is given, TF_i denotes the position number of the first test case that can detect the i-th defect in this execution sequence. The value of APFD ranges from 0 to 100, with higher values indicating faster defect detection.

APBC [17] measures the proportion of code blocks covered by priority test cases and has the following formula:

$$APBC = 1 - \frac{TB_1 + TB_2 + \ldots + TB_m}{n * m} + \frac{1}{2n} \tag{2}$$

where n represents the number of test cases in the test suite T, which are used to verify the set B composed of m blocks, and TB_i represents the first test case in the prioritized order of T, which is used to detect the i-th block. APBC ranges from 0 to 100, with higher values indicating faster coverage speed and better coverage effectiveness.

APDC [17] is a metric that measures the proportion of priority test cases that cover various decision branches; it has the following calculation formula:

$$APDC = 1 - \frac{TD_1 + TD_2 + \ldots + TD_m}{n * m} + \frac{1}{2n} \tag{3}$$

where n similarly denotes the number of test cases in the test suite T, which are used to verify the set D consisting of m branches, and TD_i represents the first test case in the prioritized order of T, which is used to detect the i-th branch. APDC ranges from 0 to 100, where higher values indicate faster coverage speed and better coverage effectiveness.

2.2 Research Status of TCP Technology

In the field of metaheuristics, the research methods for TCP technology have evolved from simple greedy strategies to sophisticated metaheuristic algorithms. In this section, the research status of TCP technology is reviewed from three perspectives: classical metaheuristic algorithms, novel metaheuristic algorithms, and other technologies.

2.2.1 Research Status of TCP Technology Based on Classical Metaheuristic Algorithms

Classical metaheuristic algorithms such as the GA, PSO, and ACO have laid the foundation for TCP studies. A genetic algorithm-based method for time-constrained TCP in which APFD is used as the evaluation metric was proposed [8]. A search-based TCP method that uses the GA, a greedy algorithm, and the hill-climbing algorithm was introduced. The experimental results demonstrated that the GA performed optimally [17]. Two value-based TCP techniques were proposed; both techniques involve the use of the GA. The results showed that the GA outperformed existing state-of-the-art TCP technologies [18]. PSO was used to automatically prioritize test cases on the basis of modified software units. Empirical results demonstrated that the PSO effectively and efficiently ranked the test cases within test suites and placed them in new optimal positions [10]. In [19], a three-stage method was proposed to address TCP by using a multiobjective PSO (MOPSO) algorithm to optimize both fault coverage and execution time. In [20], an improved TCP method based on quantum PSO was introduced, which outperformed the bat algorithm, grey wolf optimizer, and PSO in experimental evaluations. By reordering test suites under time constraints, ACO was applied to TCP as a method for solving time-constrained prioritization issues [9]. In [21], a hybrid of ACO and GA was proposed, which generates test cases on the basis of priority assigned by test factors and then uses ACO to compute the optimal sequence with the shortest execution time and highest fault rate. For solving requirement-based TCP, an ACO-based solution was introduced in [22], and two implementation methods were presented. The experimental results show that this solution has a strong global optimization ability.

The aforementioned methods typically involve the use of a single algorithmic framework for TCP. As the non-free-lunch theorem [23] demonstrates, standalone heuristic algorithms have inherent limitations when optimization challenges are addressed.

2.2.2 Research Status of TCP Technology Based on Novel Metaheuristic Algorithms

In recent years, a series of novel metaheuristic algorithms have been introduced into TCP and have demonstrated significant potential to address increasingly complex testing environments. In [24], an approach for optimal test case prioritization based on the firefly algorithm was proposed. The overall APFD results indicate that the firefly algorithm is a promising competitor in TCP applications. In [25], an asexual reproduction repair mechanism for discretizing the continuous cuckoo search algorithm was introduced. In [26], a TCP method that is based on an improved Harris hawks optimization algorithm with two optimization targets, namely, the average defect detection rate and the overall execution time, was proposed. In [27], the gray wolf optimization algorithm (GWO) was improved by using cyclic chaotic functions for TCP. In [28], a TCP method based on an improved whale optimization algorithm (WOA) was developed, and a multidimensional directed search space was established to better apply the WOA to TCP. Single heuristic algorithms have inherent limitations in solving optimization problems. In [29], a novel hybrid metaheuristic method for prioritizing and optimizing test cases was introduced. The proposed hybrid algorithm comprehensively considers factors such as the code coverage, fault discovery rate, and execution time, thereby effectively addressing the challenges posed by large-scale test cases and dynamically evolving systems.

Fusion algorithms demonstrate advantages in increasing TCP efficiency. However, their reliance on disparate single-algorithm principles significantly increases method complexity after integration, which consequently increases the computational overhead.

2.2.3 Research Status of Other TCP Technologies

In addition to classical and novel metaheuristic algorithms, numerous studies have applied machine learning, reinforcement learning, and deep learning techniques to TCP in recent years. In [30], a systematic review of machine learning applications in test case selection and prioritization was conducted. In [31], the scalability and accuracy of TCP technology in continuous integration environments were highlighted, and the use of machine learning algorithms to adapt to evolving test data and execution patterns by many contemporary approaches was reported. In [32], a semantic-aware two-stage TCP framework that uses information retrieval techniques for initial sorting and filtering of test cases, followed by fine-grained sorting on the basis of semantic similarity using pretrained Siamese language models, was proposed. Most of the aforementioned approaches use machine learning to mitigate issues in various scenarios.

However, these approaches often require sustained feedback mechanisms and involve computationally intensive operations such as model training and parameter tuning, thereby leading to high training costs and complex implementation.

2.3 Motivation

Despite substantial progress in TCP, a critical dichotomy persists between testing sufficiency and execution efficiency within continuous integration pipelines. Conventional metaheuristics are often constrained by rigid search dynamics and premature convergence, yielding suboptimal prioritization due to insufficient population diversity and weak escape mechanisms. Conversely, while hybrid and machine learning-enhanced approaches mitigate these performance bottlenecks, they introduce high training costs and complex implementation.

The AO presents a promising yet unexplored paradigm for TCP. However, its direct application is impeded by two fundamental barriers: (1) Structural Discontinuity: The intrinsic mismatch between AO's continuous search space and TCP's discrete permutation domain traditionally necessitates complex repair

operators to rectify invalid solutions, thereby eroding computational efficiency. (2) Algorithmic Stagnation: Even with valid encoding, standard AO suffers from low initialization diversity and a fixed exploration-exploitation transition, leading to premature convergence in the rugged landscape of test case sequences.

To bridge these gaps, we propose the MCIAO_TCP framework, which systematically addresses both aforementioned challenges. First, we employ the Smallest Position Value (SPV) [33] rule to convert continuous vectors into valid test case permutations. This approach inherently guarantees solution legality, obviating the need for computationally expensive repair operators. Second, we integrate four phase-aware strategies to dynamically enhance diversity, escape local optima, and refine solution precision. The synergistic combination of these strategies ensures that MCIAO_TCP not only adapts AO to the discrete TCP domain but also significantly outperforms existing metaheuristic-based methods.

3 MCIAO_TCP

3.1 Original AO Algorithm

The AO algorithm draws inspiration from the hunting behavior of aquila birds. The algorithm divides the optimization process into exploration and exploitation phases according to the number of iterations. Specifically, when the condition $t <= (2/3)T$ holds (where t represents the current iteration number and T represents the maximum number of iterations), the AO performs global exploration; otherwise, it switches to local exploitation. The four distinct hunting stages, along with their corresponding mathematical models, are sequentially described below [13].

3.1.1 Expanded Exploration

During this stage, the aquila hovers at high altitude and uses its acute vision to observe and identify potential prey zones. It then dives vertically to select the optimal hunting zone. The mathematical model is expressed as follows:

$$X_i(t+1) = X_{best}(t) \times \left(1 - \frac{t}{T}\right) + \left(X_M(t) - X_{best}(t) \times rand1\right) \tag{4}$$

where $X_i(t + 1)$ represents the position of the i-th individual at iteration $t + 1$; $X_{best}(t)$ represents the best solution found by iteration t (the elite individual), which indicates the approximate prey location; and $X_M(t)$ indicates the mean position of all the individuals at the t-th iteration, which can be calculated by Eq. (5).

$$X_M(t) = \frac{1}{n}\sum_{i=1}^{N} X_i(t) \tag{5}$$

In Eq. (5), $X_i(t)$ denotes the position of the i-th individual at the t-th iteration.

3.1.2 Narrowed Exploration

During this stage, the aquila hovers above the target prey and meticulously searches the designated hunting area while preparing to strike. The mathematical model that characterizes this behavior is formulated as follows:

$$X_i(t+1) = X_{best}(t) \times Levy(Dim) + X_R(t) + (y - x) \times rand2 \tag{6}$$

where $X_i(t+1)$ and $X_{best}(t)$ are defined as in Eq. (4); $Levy(Dim)$ denotes the Lévy flight distribution function, which can be calculated using Eq. (7).

$$Levy\left(Dim\right) = \left(s \times u \times \sigma\right) / |v|^{1/\beta} \tag{7}$$

In Eq. (7), s and β are constants with values of 0.01 and 1.5, respectively; u and v denote normally distributed random numbers within the interval [0, 1]; and σ can be determined via Eq. (8).

$$\sigma = \frac{\tau\left(1+\beta\right) \times \sin\left(\pi\beta/2\right)}{\tau\left(1+\beta\right) \times \beta \times 2^{(\beta-1)/2}} \tag{8}$$

In Eq. (8), τ represents the gamma function.

$X_R(t)$ represents a randomly selected individual from the current iteration, and y and x describe the spiral flight trajectory of the aquila during exploration, which can be computed using Eqs. (9) and (10), respectively.

$$y = \left(r_1 + U \times D_1\right) \times \cos\left(\theta\right) \tag{9}$$

$$x = \left(r_1 + U \times D_1\right) \times \sin\left(\theta\right) \tag{10}$$

$$\theta = 3\pi/2 - \omega \times D_1 \tag{11}$$

where r_1 has a value range of [1, 20], which indicates the predetermined number of search cycles; U and ω are constant parameters set to 0.0056 and 0.005, respectively; and D_1 denotes an integer value within the range [1, Dim].

3.1.3 Expanded Exploitation

During this stage, upon precise prey localization, the aquila executes a rapid vertical descent to conduct a preliminary attack while monitoring the response of the target. The mathematical model for this behavior is formulated as follows:

$$X_i\left(t+1\right) = \left(X_{best}\left(t\right) - X_M\left(t\right)\right) \times \alpha - rand3 + \left(\left(UB - LB\right) \times rand4 + LB\right) \times \delta \tag{12}$$

where $X_i(t+1)$, $X_{best}(t)$, and $X_M(t)$ are defined as in Eq. (4); both α and δ in the AO are assigned values of 0.1 and function as exploitation tuning parameters; and LB and UB denote the lower and upper bounds, respectively, of the defined search space of the given problem.

3.1.4 Narrowed Exploitation

During this stage, the aquila accurately hunts the prey according to its stochastic ground movements, which is modeled mathematically as follows:

$$X_i\left(t+1\right) = QF\left(t\right) \times X_{best}\left(t\right) - \left(G_1 \times X_i\left(t\right) \times rand5\right) - G_2 \times Levy\left(Dim\right) + G_1 \times rand6 \tag{13}$$

where $X_i(t+1)$ and $X_{best}(t)$ are defined as in Eq. (4), $X_i(t)$ and $Levy(Dim)$ are defined as in Eq. (5), and $QF(t)$ represents the search-strategy-balancing quality function defined in Eq. (14).

$$QF\left(t\right) = t^{\frac{2 \times rand7 - 1}{(1-T)^2}} \tag{14}$$

The parameter $G_1 = 2 \times rand\ 8 - 1$, characterizes the tracking trajectory of the aquila during the aerial pursuit of escaping prey and decreases linearly from 1 to −1 during iterative optimization. The parameter

$G_2 = 2 \times (1 - t/T)$, represents the flight slope of the aquila during the aerial pursuit of escaping prey and decreases linearly from 2 to 0 during iterative optimization.

Parameters rand1 through rand8 in the aforementioned equations denote uniformly distributed random variables sampled from the closed interval [0, 1].

3.2 Improved MCIAO Algorithm

3.2.1 Adaptation of MCIAO for TCP

Since the original AO operates in a continuous space while TCP is a combinatorial permutation problem, we adapt MCIAO using the SPV rule to bridge this gap: (1) Solution Representation: Each individual is maintained as a D-dimensional real-valued vector $\vec{X} = [x_1, x_2,..., x_D]$, where D is the number of test cases. (2) Decoding Mechanism: A valid execution sequence is generated by sorting the components of $\vec{X}$ in an ascending order, the indices of the sorted values determine the test case order. For example, given $\vec{X} = [1.5, 0.8, 2.3, 0.4]$ corresponding to $\{TC_1, TC_2, TC_3, TC_4\}$, sorting yields the index sequence [4, 2, 1, 3] (since $0.4 < 0.8 < 1.5 < 2.3$), producing the test order $[TC_4, TC_2, TC_1, TC_3]$. (3) Fitness Evaluation: The decoded permutation is evaluated using APFD, APBC, or APDC metrics, and the resulting fitness value guides the continuous position updates in the subsequent AO iterations. (4) Constraint Handling: The SPV rule inherently guarantees solution legality—ensuring each test case appears exactly once without duplicates or omissions—thus eliminating the need for explicit constraint handling.

These adaptations ensure that MCIAO effectively operates in the discrete combinatorial search space of TCP while maintaining the optimization capabilities of the original metaheuristic framework.

3.2.2 Chaotic Mapping Initialization

Like most SI algorithms, the original AO initializes the population using randomly generated data, which leads to an uneven distribution of initial individuals and compromised population diversity. This makes the algorithm prone to premature convergence in later iterations, thus ultimately degrading its search efficiency and optimization capabilities. In light of this, we build upon the cosine transform method by integrating logistic mapping and sine mapping to generate a novel logistic–sine–cosine [34] composite chaotic mapping as an improvement strategy. The mathematical formulation is expressed as follows:

$$x_{i+1} = \cos\left(\pi\left(4rx_i\left(1 - x_i\right) + \left(1 - r\right)\sin\left(\pi x_i\right) - 0.5\right)\right), r \in [0,1] \tag{15}$$

To further demonstrate the advantages of this composite chaotic mapping, Fig. 1 shows a comparison of the distributions of the three individual mappings and random initialization in the [0, 1] interval. The results conclusively show that the logistic–sine–cosine mapping has a more uniform distribution than logistic mapping, sine mapping, and random initialization do and can better cover the search space to obtain a well-distributed initial solution position. The flat frequency distribution observed in Fig. 1 provides quantitative evidence that the proposed strategy prevents initial clustering, thereby ensuring a diverse population from the outset.

3.2.3 Cauchy Inverse Cumulative Distribution Flight

In both the narrowed exploration and narrowed exploitation phases of the AO, the Lévy flight strategy was introduced for random searching because of its fixed step size. However, this approach fails to consider the dynamic environmental adaptation of aquila during predation, which makes balancing large-scale coarse-grained exploration with small-scale fine-grained exploitation challenging. Consequently, the search

process lacks directional guidance and hierarchical coordination, which potentially causes the algorithm to overlook regions that contain optimal solutions.

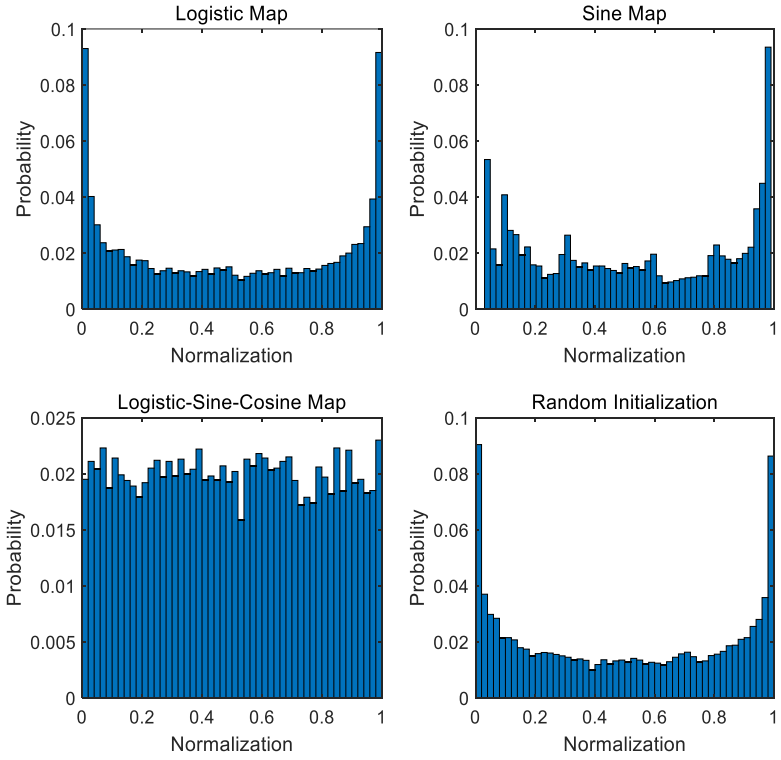

Figure 1: Distribution analysis of initialization strategies.

The Cauchy distribution is a continuous probability distribution. Under the condition that its cumulative distribution function can be calculated as an inverse function, the inverse transform method can be used to generate random numbers that follow a uniform distribution [35]. The inverse function is defined as follows:

$$step = a + b * \tan\left(pi * \left(p - 1/2\right)\right) \tag{16}$$

$$p = Randn\left(1, Dim\right) \tag{17}$$

During the narrowed exploration and narrowed exploitation phases, the random walk behavior is regulated by the location parameter a and the scale parameter b. The location parameter is set to $a = 0$ to center the perturbation around the current solution. The scale parameter $b = 0.01$ is empirically chosen to balance the heavy-tailed nature of the Cauchy distribution—which enables occasional large jumps for global exploration—with sufficiently small step sizes to support local refinement. This configuration leverages the exploratory strength of Cauchy-based perturbations while maintaining effective local search capability, thereby facilitating stable and effective convergence toward high-quality solutions. Improved equations, namely, Eqs. (18) and (19), are obtained by substituting these values into Eqs. (6) and (13).

$$X_i\left(t+1\right) = X_{best}\left(t\right) \times \left(0.01 \times \tan\left(\left(pi \times \left(p - 1/2\right)\right)\right)\right) + X_R\left(t\right) + \left(y - x\right) \times rand2 \tag{18}$$

$$X_i\left(t+1\right) = QF\left(t\right) \times X_{best}\left(t\right) - \left(G_1 \times X_i\left(t\right) \times rand5\right) - G_2 \times \left(0.01 \times \tan\left(\left(pi \times \left(p - 1/2\right)\right)\right)\right) + G_1 \times rand6 \tag{19}$$

3.2.4 Mutated Random Walk

In the exploration phase of AO, a fixed-step uniform random search is conducted, which demonstrates inadequate responsiveness to the dynamic characteristics of the solution space. As a result, the search range cannot be dynamically adjusted according to the complexity of the solution space, which ultimately causes either extensive regional omissions or redundant searches.

The random walk strategy is a commonly used approach in search and optimization algorithms. During its execution, the system randomly selects a candidate solution from the search space according to predefined rules to participate in individual updates, thereby improving the comprehensive exploration of the solution space while preventing the population from becoming stuck in local optima [36]. Drawing inspiration from mutation mechanisms, in this study, a random walk approach, namely, the mutated random walk approach, which extends conventional random walk strategies, is introduced. This approach dynamically modulates individual movement characteristics by adjusting the step size, direction parameters, and path constraints while incorporating a mutation mechanism to generate new solutions, thereby increasing the adaptability and diversity of the population and significantly improving the global exploration performance of the algorithm.

The new individuals generated after mutation are calculated using Eq. (20):

$$X_i\left(t+1\right) = X_i\left(t\right) + rand * \left(X\left(n\left(1\right),:\right) - X\left(n\left(2\right)\right)\right) \tag{20}$$

where: $n = rand\,i([1\ \mathrm{Dim}], 2)$. Introducing the mutated random walk into Eq. (4) yields:

$$X_i\left(t+1\right) = Mutation\left(X_{best}\left(t\right) \times \left(1 - \frac{t}{T}\right) + \left(X_M\left(t\right) - X_{best}\left(t\right) \times rand1\right)\right) \tag{21}$$

The modified random walk strategy is introduced into Eq. (18):

$$X_i\left(t+1\right) = Mutation\left(X_{best}\left(t\right) \times Distribution\left(Dim, Index\_fly\right) + X_R\left(t\right) + \left(y - x\right) \times rand2\right) \tag{22}$$

The mutated random walk operates entirely in the continuous domain. After updating the continuous position vector using Eqs. (21) or (22), the SPV rule described in Section 3.2.1 is applied to convert the continuous vector into a valid permutation of test cases. This two-step mechanism guarantees that any movement in the continuous space—regardless of its magnitude or direction—always yields a legitimate test case sequence without requiring additional repair operators or constraint handling.

3.2.5 Specular Reflection Learning

The inspiration for specular reflection learning stems from the reflection of light, and its model is illustrated in Fig. 2 [37]. This method involves leveraging the principle of symmetry to generate boundary-constrained opposite solutions along multiple mirror directions. By replacing original individual positions with these opposite solutions, it effectively guides individuals to migrate toward high-resource regions.

The AO predominantly uses the current optimal individual to guide the search process during late iterations, which tends to decrease the population diversity and may lead to premature convergence to a local optimum. To overcome this limitation, the specular reflection learning strategy can be used to perturb the position of the best individual in the later stages. This mechanism not only enriches the diversity of the population but also better emulates the natural habitat selection behavior of aquila birds, thereby enabling broader solution space exploration while robustly preventing entrapment in local optima and increasing

convergence accuracy. According to the principle of symmetry in specular reflection learning, the opposite point $\breve{x}$ of x can be calculated using Eq. (23).

$$\breve{x} = (0.5\lambda + 0.5) \times (Z_L + Z_U) - \lambda x \tag{23}$$

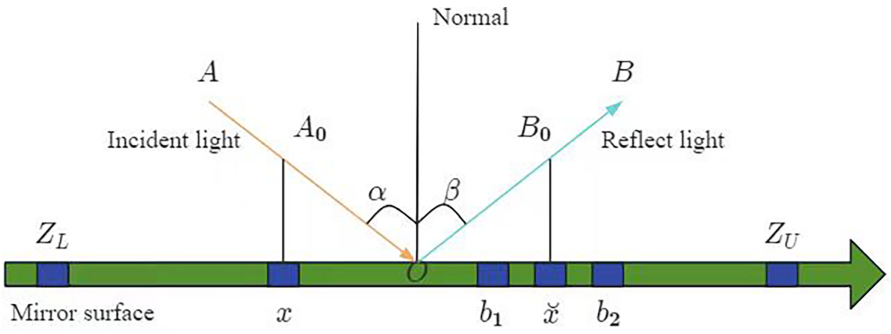

Figure 2: Schematic diagram of specular reflection learning.

By integrating specular reflection learning with AO, Eq. (23) can be transformed into Eq. (24).

$$X_{\mathrm{i}}(t+1) = (0.5 \times rand + 0.5) \times (UB + LB) - \mathrm{rand} \times X_i(t) \tag{24}$$

In the context of TCP, specular reflection learning operates in the continuous domain. The opposite solution generated by Eq. (24), which is computed as a symmetric point with respect to the current best solution, is mapped to a test case permutation via the SPV rule (Section 3.2.1), yielding a sequence that is structurally distinct from the existing candidates. This mechanism introduces targeted diversity during later iterations when the population tends to concentrate around local optima.

3.2.6 Implementation Procedure

A schematic diagram of the MCIAO_TCP is shown in Fig. 3. The process begins by extracting relevant software programs from the datasets for testing. A test suite is then developed to generate test cases. TCP sequencing is performed using a search-based optimization approach, where the proposed improved algorithm MCIAO is used to formulate strategies while incorporating evaluation metrics such as APFD, APBC, and APDC. The final step yields prioritized test cases.

The MCIAO algorithm is developed by applying the improvement strategies described in Sections 3.2.2–3.2.5 to the AO; its workflow is illustrated in detail in Algorithm 1.

3.3 Time Complexity Analysis of the MCIAO

We let N represent the population size, D represent the problem dimension, and MT represent the maximum number of iterations. The time complexity of the AO consists of three parts: the initialization phase O(N × D), position updating O(N × MT × D), and fitness function calculation O(N × MT). The overall complexity is $f(T_1) = O(N \times D) + O(N \times MT \times D) + O(N \times MT) = O(N \times MT \times D)$.

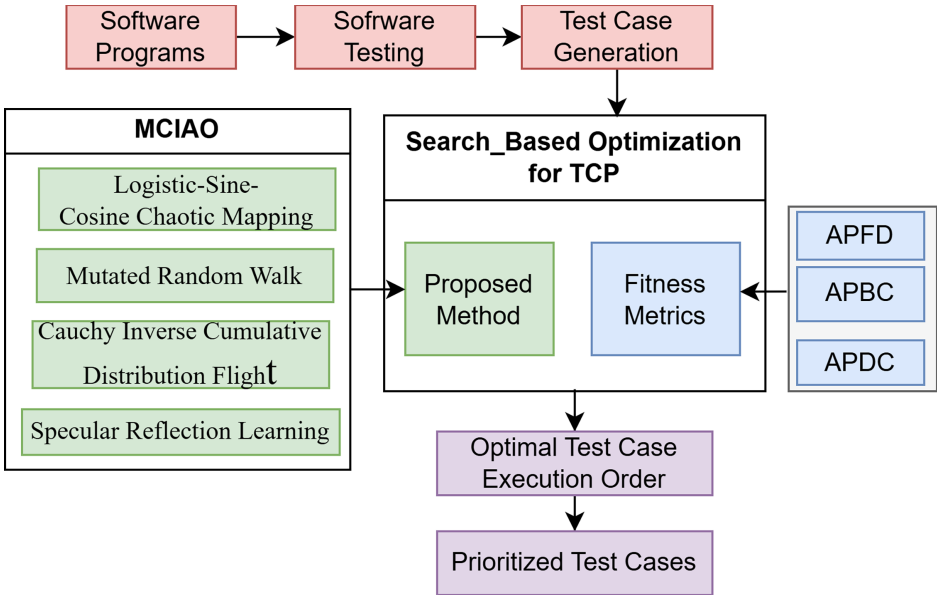

Figure 3: Overall architecture of the proposed MCIAO_TCP.

Algorithm 1: MCIAO algorithm

Initialize the parameters of the AO (N, T, α, δ, etc.)
Initialize the population using logistic–sine–cosine chaotic mapping
While ($t < T$)
  Calculate the fitness values
  Determine the optimal aquila individual X_{best}
  For each individual i do
    If $t < 2/3T$
      If $rand < 0.5$
        Expanded exploration using Eq. (21)
      Else
        Narrowed exploration using Eq. (22)
      End If
    Else
      If $rand < 0.5$
        Expanded exploitation using Eq. (12)
      Else
        Narrowed exploitation using Eq. (19)
      End If
    End If
  End For
  Generate the opposite solution $\breve{x}$ of X_{best} using Eq. (24)
  Evaluate fitness of $\breve{x}$
  If $\breve{x} < X_{best}$ Then
    $X_{best} \leftarrow \breve{x}$
  End If

(Continued)

Algorithm 1 (continued)

26: $t = t + 1$
27: End While
28: Return the best solution X_{best}

For the MCIAO, we let t_1 represent the time required to introduce the chaotic mapping. Afterward, the time complexity of the initialization phase after the chaotic mapping is introduced becomes $f(T_1) = O(N \times D + t_1)$. We let t_2 represent the time required to generate new solutions through a mutated random walk and t_3 represent the time required to generate new solutions using Cauchy inverse cumulative distribution flight. Then, the time complexity of the search phase for incorporating the improved strategies for position update and fitness function evaluation is $f(T_2) = O(N \times MT \times D + t_2) + O(N \times MT) + O(N \times MT \times D + t_3)$. We let t_4 represent the time required to generate opposite solutions using the specular reflection learning strategy. The time complexity of generating opposite solutions with specular reflection learning is $f(T_3) = O(N \times D + t_4)$. Therefore, the overall time complexity of the MCIAO is $f(T_2) = f(T_1) + f(T_2) + f(T_3) = O(N \times D + t_1) + (O(N \times MT \times D + t_2) + O(N \times MT) + O(N \times MT \times D + t_3)) + O(N \times D + t_4) = O(N \times MT \times D)$, which remains consistent with f(T1).

While a formal proof of global convergence is beyond the scope of this applied study, the design of MCIAO is grounded in well-established principles of stochastic optimization. Specifically: logistic–sine–cosine chaotic mapping enhances population diversity; mutated random walk and the Cauchy inverse cumulative distribution flight jointly enable adaptive, long-range jumps that improve global exploration; and specular reflection learning perturbs the best solution in later stages to help escape local optima. Together, these components promote a sustained balance between exploration and exploitation—a key factor for effective metaheuristic performance.

The $O(N \times MT \times D)$ complexity accounts only for algorithmic operations. The evaluation cost (APFD/APBC/APDC) is identical per iteration across methods and based on precomputed matrices. MCIAO's faster convergence reduces the number of such evaluations, resulting in lower effective runtime. Metaheuristic overhead remains negligible compared to test execution time.

4 Experimental Design and Datasets Selection

4.1 Research Questions

In this study, the CEC2017 and CEC2020 benchmark test functions are used for numerical simulation experiments, with five real datasets selected for analysis. The primary objective is to address three research questions to validate the effectiveness of the proposed algorithm.

RQ1: How does the improved algorithm perform in terms of convergence speed and stability?

To address this issue, we conduct performance analysis experiments for MCIAO on the CEC2017 and CEC2020 benchmark functions to evaluate the convergence speed, stability, and accuracy of the algorithms, and the results are presented in Section 5.1.

RQ2: How do the four proposed improvement strategies individually affect the performance of the algorithm?

To address this issue, as reported in Section 5.2, an ablation experiment is conducted to compare the performance of the MCIAO, original AO, and each AO algorithm using a single improved strategy to validate the effectiveness of the hybrid improved algorithm.

RQ3: Does the improved algorithm significantly outperform the baseline algorithm in terms of the APFD, APBC, and APDC metrics?

To address this issue, as reported in Section 5.3, comparative experiments are conducted to evaluate the MCIAO algorithm against eight other SI optimization algorithms across various metrics to demonstrate the superiority of the improved algorithm.

4.2 Experimental Setup and Datasets

The simulation experiment is conducted in MATLAB 2024a on a Windows 11 OS system with an 11th Gen Intel® Core™ i7-11390H processor (3.40 GHz) and 16 GB of RAM.

The Defects4J database has been widely used to evaluate TCP-related technologies. To validate the effectiveness and reliability of MCIAO, we select five Java projects from the Defects4J datasets that contain Java code defect data for testing. To ensure fair comparison, we prioritize datasets with comparable evaluation metrics. Detailed information on these five open-source datasets is presented in Table 1.

Table 1: Statistical information about the five projects.

Project Name	Identifier	Number of Defects	Number of Tests
Jfreechart	Chart	26	2233
Closure-compiler	Closure	174	9372
Commons-lang	Lang	61	4355
Commons-math	Math	106	8629
Joda-time	Time	26	4055

4.3 Comparison Algorithms and Parameter Settings

In this study, comparative experiments are conducted using AO, GOOSE [38], BKA [39], SSA [40], HHO [41], WOA [42], GWO [43], PSO [44] along with LSHADE [45] and iCSPM [46] on the numerical benchmark functions, while only the first eight algorithms (AO to PSO) are used on the real-world TCP problem, to validate the effectiveness of MCIAO. To ensure simulation fairness, all the comparison algorithms are conducted under uniform parameter settings, namely, maximum iteration count T = 500, population size N = 50, and dimensional sizes Dim = 10 and 100, with each algorithm performing 30 independent runs. The specific parameter configurations for each comparison algorithm are detailed in Table 2.

Table 2: Algorithm parameter settings.

Algorithm	Parameter Settings
MCIAO	C = 10
AO	C = 10
GOOSE	C = 1
BKA	p = 0.9
SSA	C_1 = [1, 0] and a = [2, 0]
HHO	J = [0, 2]
WOA	l = [−1, 1] and b = 1
GWO	a = [2 0]
PSO	C_1 = 2 and C_2 = 2
LSHAD	NP_init = N and NP_min = 4
iCSPM	pm_eta = 20

The experimental results and detailed analysis demonstrating the effectiveness of the proposed algorithm are presented in Section 5.

5 Experimental Results and Analysis

In this section, we conduct comprehensive experiments to evaluate the performance of MCAIO. Specifically, we first validate its theoretical search mechanisms using numerical benchmark functions, and then empirically assess its practical effectiveness in real-world TCP.

5.1 RQ1: MCIAO Performance Analysis on Numerical Benchmark Functions

5.1.1 Convergence Accuracy Analysis

The performance of the MCIAO is evaluated on the CEC2017 and CEC2020 benchmark functions. Four metrics are defined as follows: Mean (average fitness) reflects convergence accuracy and optimization capability; Std (standard deviation) indicates robustness and stability; Min (best achieved fitness) represents the potential to reach near-global optima; and Epoch (number of iterations to convergence) measures convergence speed and computational efficiency. Table 3 reports the complete numerical results. To improve readability, only the summarized winning counts are discussed here, while the full results are provided in Appendix A.1, Table A1, and Appendix A.2, Table A2.

Table 3: Summary of best-performing counts on CEC2017 and CEC 2020.

Benchmark	Best Mean	Best Min	Best Std.	Fastest Epoch
CEC2017	22/29	20/29	16/29	7/29
CEC2020	7/10	5/10	2/10	7/10

The results from Table 3 show that on CEC2017, MCIAO achieves the best (lowest) Mean on 22 out of 29 functions (75.9%), the best Min on 20 out of 29 functions (69.0%), and the lowest Std on 16 out of 29 functions (55.2%), indicating consistently high solution quality and robustness across independent runs. In terms of convergence speed, MCIAO attains the fastest Epoch on 7 out of 29 functions (24.1%); on the remaining functions, its Epoch values remain competitive (mostly well below the maximum iteration limit of 500). On CEC2020, MCIAO achieves the best (lowest) Mean on 7 out of 10 functions (70%), including F_1, F_3, F_5, F_6, F_7, F_9, and F_{10}, and the best Min on 5 functions (50%), notably on F_1, F_3, F_6, F_7, and F_9, indicating its strong capability to locate high-quality solutions. In terms of convergence speed, MCIAO attains the fastest Epoch on 9 out of 10 functions (90%), with only F_6 where GWO converges slightly faster. Regarding stability, MCIAO achieves the lowest Std on F_1 and F_9; on the remaining functions, its Std values are generally comparable to the best competitors. Overall, MCIAO exhibits a strong balance between solution accuracy, stability, and convergence efficiency on both test suites.

5.1.2 Convergence Analysis

To visually compare the MCIAO with the reference algorithms, we plot partial convergence curves, as shown in Figs. 4 and 5, using simulation data from Appendix A.1, Table A1, and Appendix A.2, Table A2.

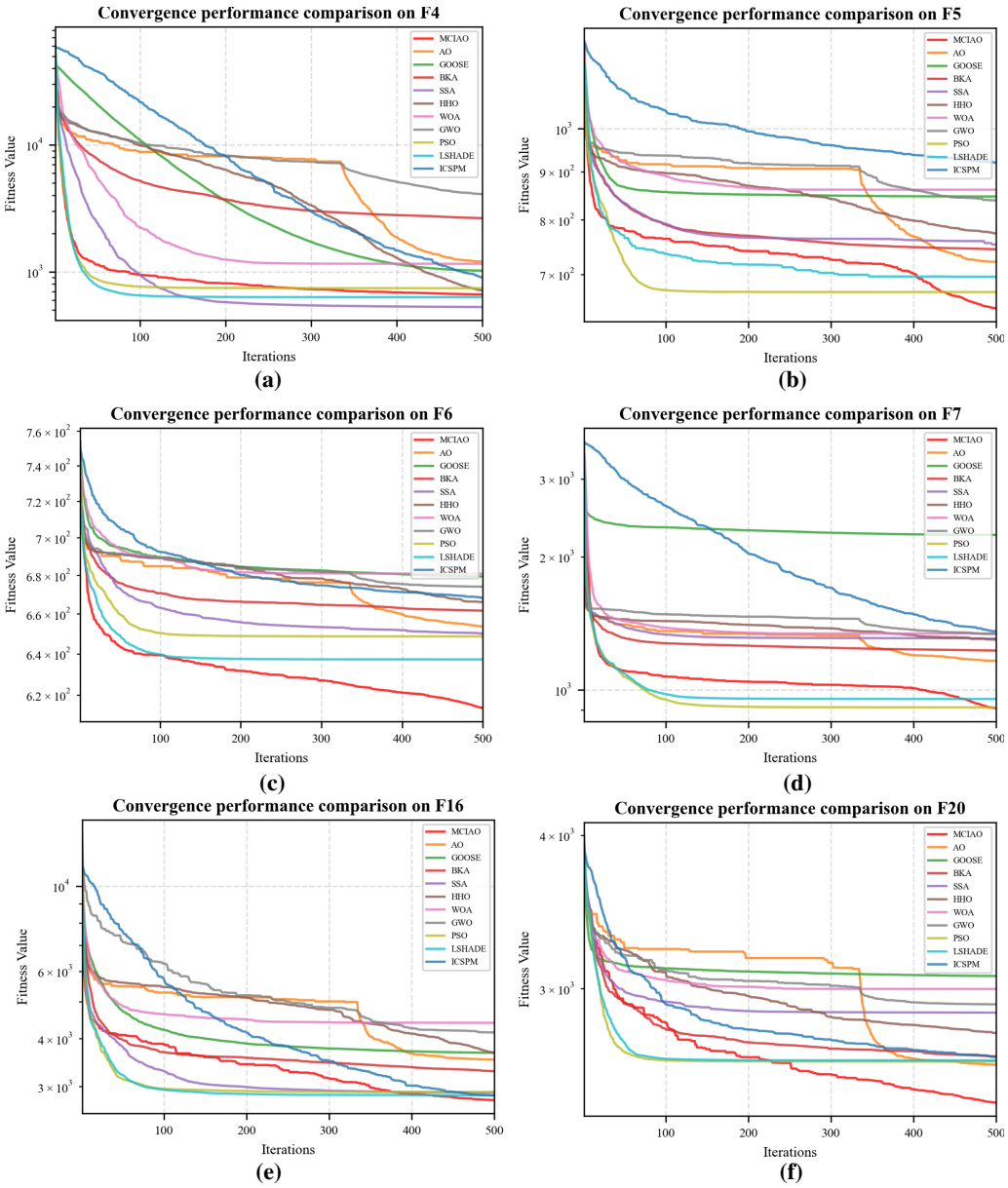

Figure 4: Partial convergence performance comparison on CEC2017 functions. (**a**) F_4; (**b**) F_5; (**c**) F_6; (**d**) F_7; (**e**) F_{16}; (**f**) F_{20}.

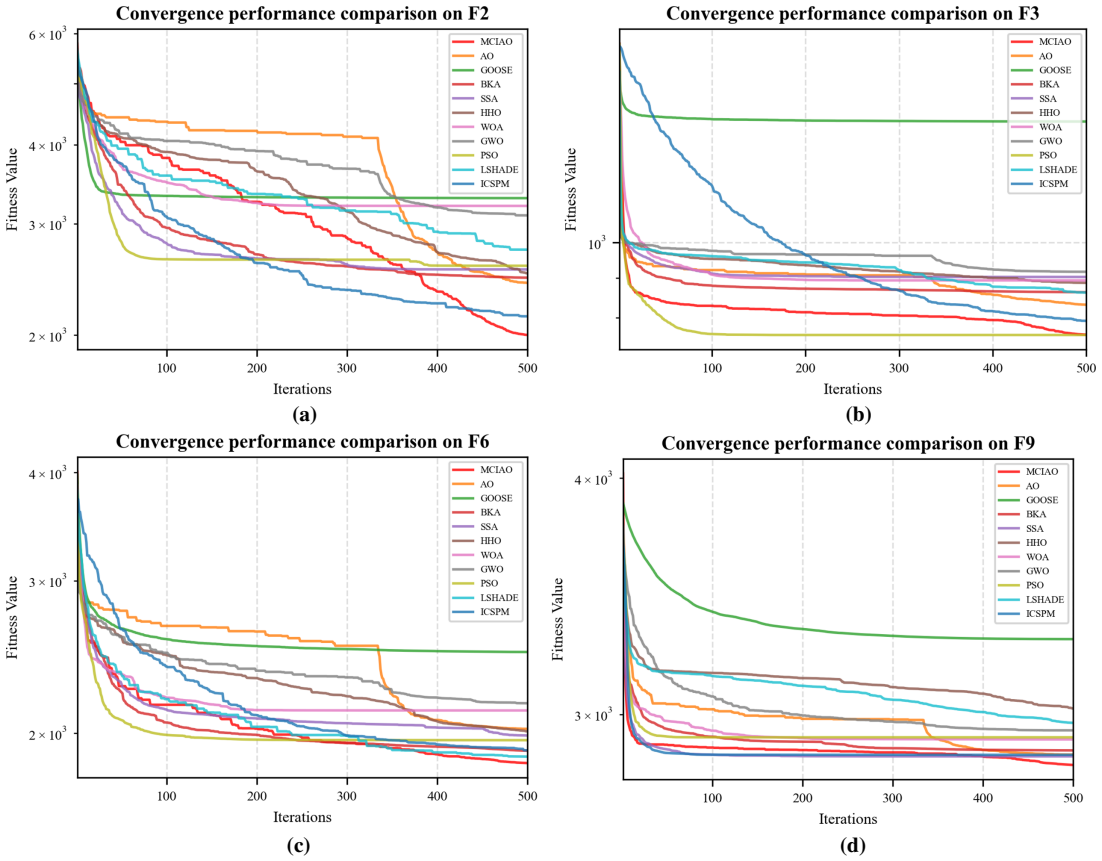

Figure 5: Partial convergence performance comparison on CEC2020 functions. (**a**) F_2; (**b**) F_3; (**c**) F_6; (**d**) F_9.

As shown in Fig. 4, MCIAO achieves the fastest convergence speed and highest solution accuracy among all the tested functions F_4–F_7, F_{16}, and F_{20}. PSO exhibits rapid initial convergence but premature stagnation, often trapped in local optima. BKA and SSA show moderate performance with somewhat lower final precision. WOA and GWO demonstrate average convergence behavior, with visible degradation on complex landscapes like F_{16} and F_{20}. LSHADE stagnates prematurely on F_7 and F_{20}, while iCSPM converges slowly across all test cases. GOOSE performs poorly, failing completely on F_7. Overall, MCIAO demonstrates superior exploration–exploitation balance, leading to robust optimization performance.

As shown in Fig. 5, which presents results on selected functions including multimodal functions F_2 and F_3, hybrid function F_6, and composition function F_9, MCIAO consistently achieves the fastest convergence and lowest final fitness among all algorithms. While PSO exhibits rapid initial convergence on F_2 and F_6, it suffers from premature stagnation on F_3 and F_9, demonstrating inconsistent performance. WOA, GWO, and SSA show moderate convergence but significantly lower final accuracy. LSHADE and iCSPM, despite occasional competitiveness, generally converge slower or stagnate earlier than MCIAO. GOOSE performs poorest, failing to converge on F_9. MCIAO's robust exploration–exploitation balance ensures its adaptability across diverse landscapes.

5.1.3 Wilcoxon Rank-Sum Test

The Wilcoxon rank-sum test was employed to investigate whether the observed differences between MCIAO and the competing algorithms are statistically reliable. In this study, the significance level was fixed at

0.05. When the calculated p-value is smaller than 0.05, the corresponding difference is regarded as statistically significant; otherwise, the two methods are considered to exhibit comparable performance from a statistical perspective. Tables 4 and 5 summarize the statistical comparison results obtained from the Wilcoxon rank-sum test on CEC2017 and CEC2020 benchmark suites. For clarity, only the numbers of statistically significant and non-significant cases are reported in the main manuscript, while the detailed p-values are provided in Appendix B.1 Table A3, and Appendix B.2 Table A4.

Table 4: Summary of Wilcoxon rank-sum test results on CEC2017.

Algorithm	Significant Difference ($p < 0.05$)	Non-Significant Difference ($p \geq 0.05$)
AO	26	3
GOOSE	23	6
BKA	27	2
SSA	26	3
HHO	21	8
WOA	19	10
GWO	28	1
PSO	29	0
LSHADE	29	0
iCSPM	25	4

Table 5: Summary of Wilcoxon rank-sum test results on CEC2020.

Algorithm	Significant Difference ($p < 0.05$)	Non-Significant Difference ($p \geq 0.05$)
AO	9	1
GOOSE	9	1
BKA	8	2
SSA	7	3
HHO	7	3
WOA	5	5
GWO	9	1
PSO	9	1
LSHADE	10	0
iCSPM	10	0

The results confirm that MCIAO exhibits statistically significant superiority over the majority of competitors. Specifically, MCIAO significantly outperforms PSO and LSHADE on all 29 functions (100%), GWO on 28 functions (96.6%), BKA on 27 functions (93.1%), AO and SSA on 26 functions (89.7%), iCSPM on 25 functions (86.2%), GOOSE on 23 functions (79.3%), HHO on 21 functions (72.4%), and WOA on 19 functions (65.5%). These results demonstrate that MCIAO achieves consistently better solution quality with high statistical confidence across the majority of test functions.

The results demonstrate that MCIAO exhibits statistically significant superiority over the majority of competing algorithms. Specifically, MCIAO achieved significantly better results ($p < 0.05$) in 10 out of 10 functions against LSHADE and iCSPM, indicating complete dominance over these advanced continuous optimizers. It also outperformed PSO, GWO, AO, and GOOSE on 9 out of 10 functions (90%), and BKA

on 8 functions (80%). Strong statistical differences were also observed against SSA and HHO (7 out of 10 functions each). In comparison with WOA, which emerged as the most competitive counterpart, MCIAO showed significant superiority in 5 out of 10 functions, while the remaining cases ($p \geq 0.05$) suggest comparable performance in specific complex landscapes. Notably, there were no instances where any competitor significantly outperformed MCIAO. These results confirm the consistent statistical advantage and robustness of MCIAO on the CEC2020 benchmark suite.

5.1.4 Execution Time Comparison

To evaluate the actual computational efficiency, we measured the average wall-clock execution time (in seconds) over 30 independent runs on four representative CEC2020 functions: F_2 (multimodal), F_3 (multimodal), F_6 (hybrid), and F_9 (composition). All experiments were conducted under identical hardware and software conditions. Table 6 reports the results.

Table 6: Average execution time (seconds) on four CEC2020 functions.

	MCIAO	AO	GOOSE	BKA	SSA	HHO	WOA	GWO	PSO	LSHADE	iCSPM
F_2	1.24E−01	1.18E−01	9.06E−02	1.09E−01	1.51E−01	1.94E−01	1.09E−01	1.05E−01	**7.24E−02**	1.01E−01	1.90E−01
F_3	8.67E−02	6.14E−02	**5.83E−02**	9.52E−02	1.49E−01	1.20E−01	9.94E−02	7.81E−02	7.48E−02	7.09E−02	1.84E−01
F_6	**3.49E−02**	3.62E−02	6.94E−02	7.90E−02	7.38E−02	9.47E−02	5.42E−02	6.90E−02	4.48E−02	7.63E−02	1.24E−01
F_9	1.68E−01	1.41E−01	1.03E−01	1.71E−01	1.38E−01	2.12E−01	1.10E−01	1.19E−01	**8.96E−02**	1.08E−01	2.09E−01
Average	1.04E−01	8.91E−02	8.03E−02	1.14E−01	1.28E−01	1.55E−01	9.29E−02	9.29E−02	**7.04E−02**	8.92E−02	1.77E−01

Note: The best results are highlighted in bold.

As shown, MCIAO achieves the fastest execution time on the hybrid function F_6 (0.035 s) and maintains competitive overall efficiency. Although it is slightly slower than PSO and GOOSE on some functions, this trade-off is compensated by its better solution quality and faster convergence (Sections 5.1.1–5.1.3). Moreover, MCIAO outperforms BKA, SSA, HHO, and iCSPM in execution time by 9%–72%. These results confirm that MCIAO offers superior optimization performance without compromising practical efficiency, making it suitable for real-world applications requiring both accuracy and speed.

5.2 RQ2: Ablation Experiment

5.2.1 Ablation Experiment

To evaluate the effects of different improvement strategies, in this study, comparative experiments involving the MCIAO, the AO, and the following four variants are conducted: AO-1 (improved solely with the logistic–sine–cosine chaotic mapping strategy), AO-2 (improved only with the Cauchy inverse cumulative distribution flight strategy), AO-3 (modified exclusively using the mutated random walk strategy), and AO-4 (optimized only with the specular reflection learning strategy). Table 7 summarizes the number of benchmark functions on which each variant achieves the best Mean and Std values. The detailed numerical results of the ablation experiments are provided in Appendix C.1, Table A5.

Table 7: Summary of ablation result on CEC2020.

Metrics	MCIAO	AO-1	AO-2	AO-3	AO-4	AO
Best Mean	9/10	0	0	0	0	1
Best Std.	8/10	0	1	0	0	1

As shown in Table 7, the MCIAO achieves the highest convergence accuracy, with the best average performance on 9 of the 10 functions. In contrast, AO-3 performs the worst on multiple functions (e.g., F_1, F_5, and F_7), which indicates its tendency to fall into local optima; AO-1, AO-2, AO-4, and AO perform similarly on some functions but are outperformed overall by the MCIAO. In terms of stability, the MCIAO demonstrates the smallest Std for most functions, which remains within a reasonable range without abnormal fluctuations, thus indicating its superior stability and reduced likelihood of extreme deviations. The Stds of AO-3 on F_1, F_5, and F_7 are extremely large, thus indicating highly unstable performance. The stability of AO-1, AO-2, AO-4, and the AO varies with the function, but overall, it is inferior to that of the MCIAO.

In summary, the ablation results in Table 7 show that each component of MCIAO addresses a specific aspect of the blind-search limitation: AO-1 mitigates initial blindness by ensuring diverse starting points; AO-2 and AO-3 jointly mitigate exploratory blindness through adaptive, long-range jumps that replace undirected movements; and AO-4 prevents convergence blindness in later stages by perturbing solutions near local optima. The full MCIAO algorithm, which integrates all four strategies, achieves the best performance, confirming the synergistic effect of these components.

To more intuitively compare the performance of the hybrid strategy–improved algorithm and single strategy–improved algorithms, we plot partial convergence curves using simulation data from Appendix C.1, Table A5, as shown in Fig. 6.

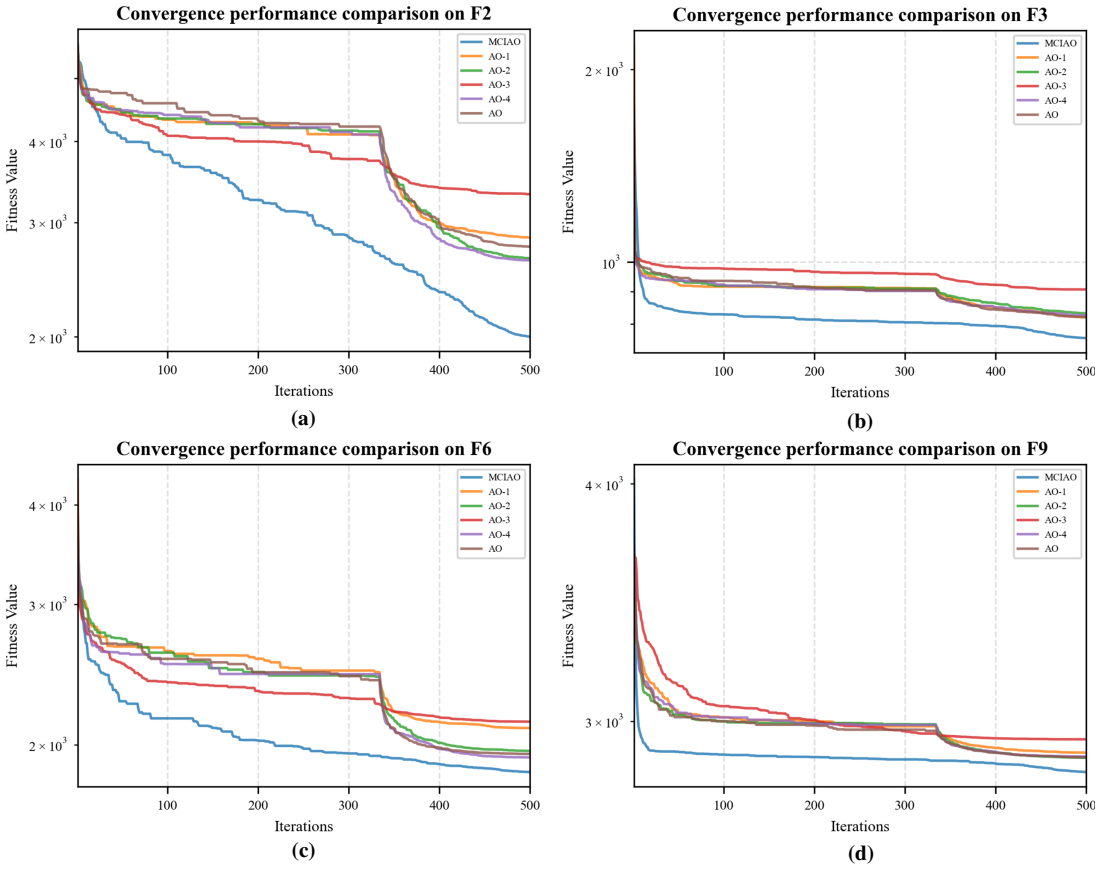

Figure 6: Partial convergence performance comparison on CEC2020 functions. (**a**) F_2; (**b**) F_3; (**c**) F_6; (**d**) F_9.

As shown in Fig. 6, for the multimodal function and hybrid functions tested (F_2, F_3, F_6, F_9,), the MCIAO achieves the fastest convergence and the lowest final fitness value, significantly outperforming all others. The

AO-3 and AO-4 are the closest competitors but still fall short, especially in later convergence stages. The AO and AO-1 algorithms consistently show the slowest convergence and poorest accuracy. Furthermore, the AO generally has slower convergence and lower accuracy and often becomes trapped in local optima or has suboptimal search efficiency. Both the individual and hybrid improvement approaches demonstrate measurable performance improvements over the original algorithm.

5.2.2 Convergence Driver Score Analysis

To provide deeper insights into the search behavior of MCIAO, we adopted the Convergence Driver Score (CDS) from the EvoMapX framework [47]. CDS quantifies the contribution of each search operator to the overall convergence by tracking fitness improvements during optimization. Four representative operators were evaluated on two CEC2020 functions (F_3 and F_9). Table 8 reports the average CDS and normalized contributions.

Table 8: Convergence driver score of MCIAO on CEC2020 benchmark functions F_3 and F_9.

Search Phase	Operator(s)	CDS (F_3)	CDS (F_9)	Average CDS	Relative Contribution (%)
Expanded Exploration	Mutated Random Walk	0.220	0.170	0.195	25.5%
Narrowed Exploration	Cauchy Inverse Cumulative Distribution Flight combined with Mutated Random Walk	0.121	0.259	0.190	24.8%
Narrowed Exploitation	Cauchy Inverse Cumulative Distribution Flight	0.143	0.248	0.195	25.5%
Later iteration stage	Specular Reflection Learning	0.259	0.109	0.184	24.1%
	Total	0.743	0.786	0.764	100%

As shown in Table 8, all four operators contribute almost equally (about 24%–26%), indicating a well-balanced exploration-exploitation strategy. The Cauchy-based operators together account for approximately 50% of the convergence events, confirming their central role in search guidance. Mutated random walk ensures global exploration, while specular reflection learning provides non-negligible local refinement in later iterations. This balanced operator attribution explains the robust optimization performance of MCIAO across diverse function landscapes.

5.3 RQ3: Comparative Experiment on Test Case Prioritization

This section evaluates the performance of MCIAO on real-world test case prioritization tasks. To verify whether the improved algorithm significantly outperforms the baseline algorithm in terms of various metrics, comparative experiments are conducted. The MCIAO is evaluated against eight other algorithms across APFD, APBC, and APDC, and the superiority of the improved approach is demonstrated.

5.3.1 APFD

In the boxplots in Fig. 7, the performance of the five open-source datasets selected in Section 4.2 are compared across several evaluation metrics, with APFD as the key metric.

Figure 7: Performance evaluation of various algorithms in terms of APFD. (**a**) Chart; (**b**) closure; (**c**) lang; (**d**) math; (**e**) time.

As shown in Fig. 7, the MCIAO consistently demonstrates superior performance across all five projects. In each case, it achieves the highest median APFD values–reaching up to 98.5% in Chart and 99% in Time–accompanied by compact data distributions, minimal outliers, and exceptional stability. Specifically, the median APFD of MCIAO substantially exceeds the overall average of all algorithms in every project: by approximately 2.5% in Chart (96% average), 7% in Closure (83.5% average), 5% in Lang (90% average), 8% in Math (83% average), and 2.3% in Time (96.7% average). These results underscore the robustness and consistent leading performance of MCIAO in comparison to the other methods evaluated.

5.3.2 APBC

In the boxplots in Fig. 8, the performance of all the algorithms on the five open-source datasets selected in Section 4.2 is compared in terms of the APBC evaluation metric.

Figure 8: Performance evaluation of various algorithms in terms of APBC. (**a**) Chart; (**b**) closure; (**c**) lang; (**d**) math; (**e**) time.

As shown in Fig. 8, the MCIAO consistently achieves the best overall performance across all projects in terms of APBC metrics, leading in both median value and stability. It is followed closely by GWO and WOA in most projects, which also exhibit stable and high performance. Notably, in the Closure project, MCIAO performs comparably to BKA and GWO, though BKA shows greater fluctuation and lower stability. In all cases, HHO consistently yields the poorest results. Furthermore, the APBC value of MCIAO exceeds the overall average of all algorithms in each project: by approximately 3% in Chart (95% average), 2.4% in Closure (87.6% average), 5.2% in Lang (89.8% average), 6.3% in Math (83.7% average), and 2.2% in Time (95.8% average), underscoring its robust and superior effectiveness compared to the other evaluated algorithms.

5.3.3 APDC

In the boxplots in Fig. 9, the performance on the five open-source datasets selected in Section 4.2 is compared across several evaluation metrics, with the APDC score as the key performance indicator.

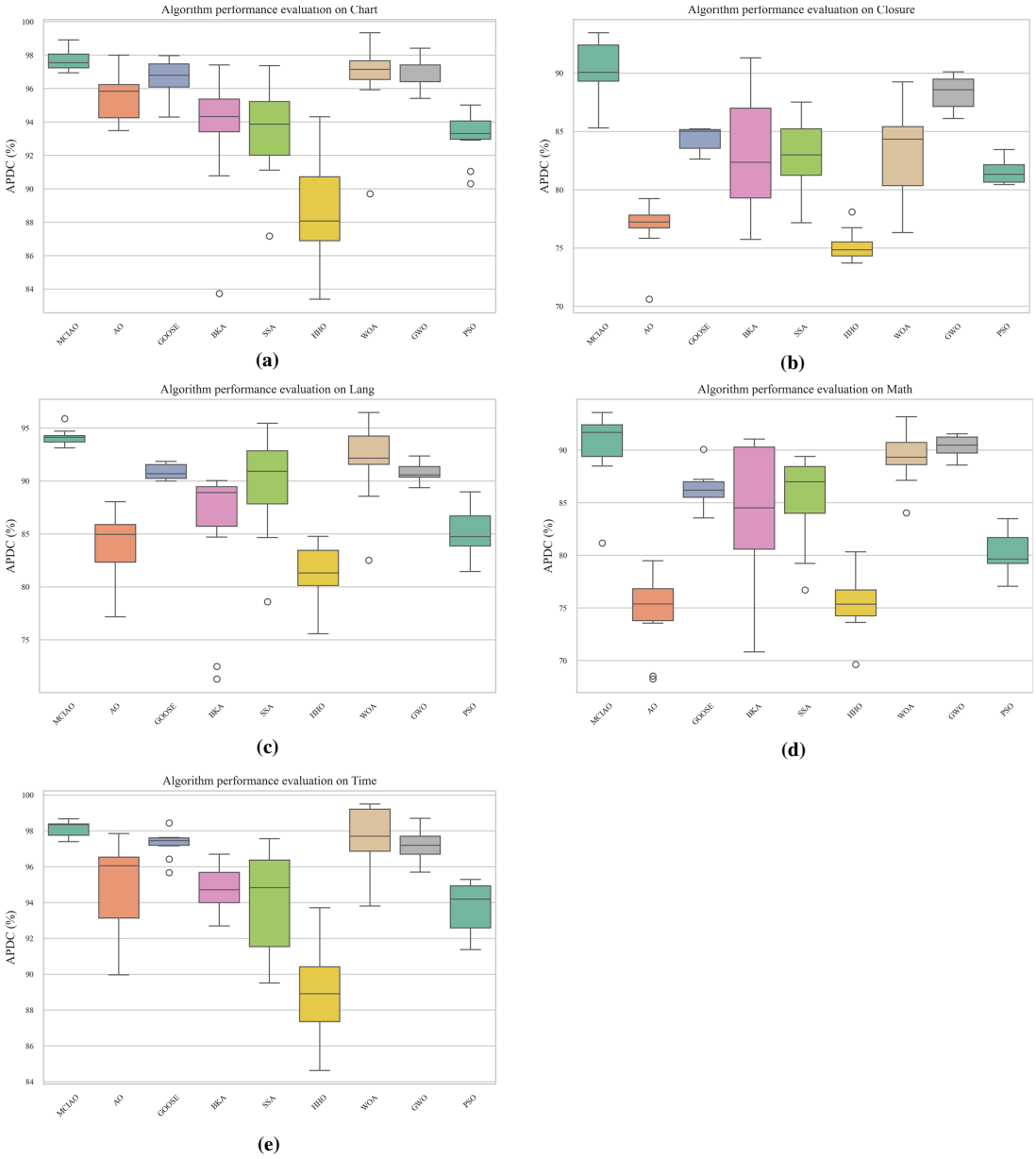

Figure 9: Performance evaluation of various algorithms in terms of APDC. (**a**) Chart; (**b**) closure; (**c**) lang; (**d**) math; (**e**) time.

As shown in Fig. 9, the MCIAO demonstrates the best overall APDC performance across all projects, consistently achieving the highest median values and superior stability. It is followed by GWO and WOA in the Chart project, while HHO consistently yields the lowest results. The APDC value of MCIAO exceeds the overall algorithm average in each project: by approximately 2.9% in Chart (94.6% average), 7.8% in Closure (82.2% average), 5.9% in Lang (88.1% average), 6.5% in Math (83.9% average), and 5.1% in Time (92.9% average), further confirming its robust and leading effectiveness.

Although code structure and complexity may affect algorithm performance across projects, the MCIAO consistently outperforms the other algorithms, which demonstrates its robustness and adaptability to software projects with diverse characteristics.

5.3.4 Convergence and Correlation Analysis on TCP

To further evaluate MCIAO on discrete optimization, we conducted TCP experiments using APFD, APBC, and APDC. Fig. 10 presents the convergence curves of all compared algorithms on the Chart and Time instances for each metric.

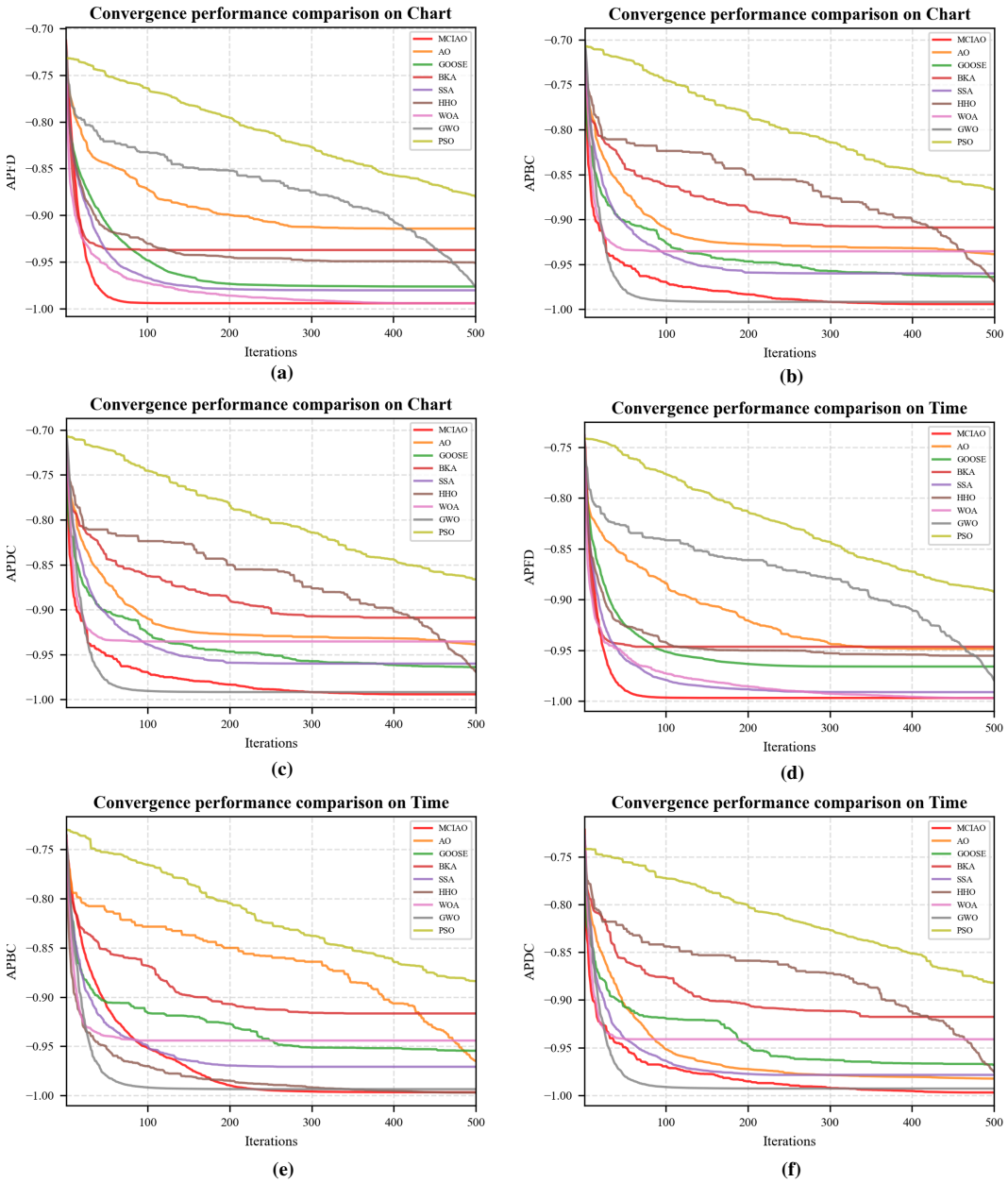

Figure 10: Convergence performance comparison on TCP. (**a**) APFD on chart; (**b**) APDC on chart; (**c**) APBC on chart; (**d**) APFD on time; (**e**) APDC on time; (**f**) APBC on time.

As shown in Fig. 10, MCIAO consistently achieves the fastest convergence and the highest final values across all three metrics and both projects. On Chart, MCIAO reaches APFD above 0.95 within 200 iterations, significantly outperforming PSO, GWO, and others. On Time, MCIAO converges to APDC ≈ 0.98 after only

150 iterations. PSO shows rapid early progress but premature stagnation; BKA and SSA exhibit moderate convergence but lower final accuracy; GOOSE and HHO perform poorly, often staying below 0.85.

The convergence curves in Fig. 10 reveal a strong practical correlation among APFD, APBC, and APDC across all studied TCP instances. The relative rankings of all algorithms remain highly consistent across the three metrics, and test sequences achieving higher structural coverage (APBC and APDC) consistently yield superior fault detection (APFD), demonstrating that coverage improvements serve as a reliable proxy for fault detection effectiveness. This strong empirical correlation validates the practical efficacy of MCIAO and confirms that optimizing for coverage metrics directly translates to improved fault detection performance.

Overall, the TCP experiments confirm that MCIAO maintains its convergence speed and solution quality on discrete optimization tasks.

5.4 Sensitivity Analysis of Cauchy Scale Parameter B

To justify the choice of b = 0.01 in the Cauchy inverse cumulative distribution flight, we conducted a sensitivity analysis on six representative CEC2017 functions (F_4–F_7, F_{16}, F_{20}) and four CEC2020 functions (F_2, F_3, F_6, F_9), comparing b = 0.01 (default), b = 0.1, and b = 0.001. Table 9 reports the Mean and Std of the final fitness values over 30 independent runs.

Table 9: Sensitivity analysis of the Cauchy scale parameter b.

Functions		Metrics	b = 0.01	b = 0.1	b = 0.001
CEC2017	F_4	Mean	**6.218E+02**	7.807E+02	8.376E+02
		Std	**2.713E+01**	4.742E+01	8.263E+01
	F_5	Mean	**6.124E+02**	7.695E+02	6.242E+02
		Std	**4.556E+00**	2.011E+01	6.102E+01
	F_6	Mean	**9.020E+02**	1.054E+03	1.153E+03
		Std	**5.048E+01**	8.114E+01	9.017E+01
	F_7	Mean	**9.005E+02**	9.571E+02	1.160E+03
		Std	**2.305E+01**	5.429E+01	5.062E+01
	F_{16}	Mean	2.071E+03	**2.068E+03**	2.320E+03
		Std	**1.970E+02**	2.432E+02	2.467E+02
	F_{20}	Mean	**2.405E+03**	2.525E+03	2.605E+03
		Std	3.057E+01	6.631E+01	**2.950E+01**
CEC2020	F_2	Mean	**2.197E+03**	2.275E+03	2.238E+03
		Std	3.127E+02	3.299E+02	**3.045E+02**
	F_3	Mean	**7.513E+02**	7.926E+02	8.168E+02
		Std	**1.239E+01**	5.262E+01	2.693E+01
	F_6	Mean	**1.862E+03**	1.910E+03	1.900E+03
		Std	**1.463E+02**	1.601E+02	1.582E+02
	F_9	Mean	2.824E+03	2.890E+03	**2.818E+03**
		Std	**2.044E+01**	2.868E+01	3.198E+01

Note: The best results are highlighted in bold.

As shown in Table 9, b = 0.01 yields the best Mean values on 5 out of 6 CEC2017 functions and 3 out of 4 CEC2020 functions. The only exceptions are F_{16} (where b = 0.1 gives a marginally lower Mean by 0.14%) and F9 (where b = 0.001 gives a marginally lower Mean by 0.22%). In terms of Std, b = 0.01 also demonstrates superior stability across most test cases. These results empirically justify b = 0.01 as the optimal default scale parameter, providing the best trade-off between exploration capability and convergence accuracy for the proposed algorithm.

6 Conclusions and Future Work

In this study, we proposed MCIAO_TCP, a multi-strategy improved aquila optimizer for TCP. By integrating logistic–sine–cosine chaotic mapping, mutated random walk, Cauchy inverse cumulative distribution flight, and specular reflection learning, the algorithm enhances population diversity, global exploration, and local refinement precision. Experiments on five Defects4J Java projects show that MCIAO_TCP achieves statistically significant, albeit modest, average improvements of 4.96%, 3.82%, and 5.64% in APFD, APBC, and APDC, respectively.

However, this work has several limitations. First, the evaluation is confined to five medium-sized Java projects, which may not fully represent the diversity and scale of real-world software systems. Second, our evaluation does not include comparisons with the most recent TCP approaches, which limits the contextualization of our results. Third, the study focuses on single-objective TCP, whereas real-world regression testing often involves multiple conflicting criteria (e.g., execution time, fault severity). Finally, the current implementation assumes precomputed coverage/fault matrices, which may not be feasible for dependency-heavy or non-Java test suites.

Despite these limitations, MCIAO retains the same asymptotic complexity as the original AO, with only constant-factor overhead. The marginal increase in algorithmic complexity is justified by the consistent and statistically significant gains. Its faster convergence reduces the number of fitness evaluations, making it suitable for CI/CD pipelines where test execution time dominates. The modest but consistent gains in APFD/APBC/APDC are practically valuable, as even small gains can accelerate feedback cycles.

Future work will address these gaps through four concrete directions: First, we will conduct comparative studies with state-of-the-art TCP approaches. Second, we will extend the evaluation to larger and more diverse projects, including C/C++ and Python systems, and explicitly address the challenges of dependency-heavy test suites through enhanced static and dynamic dependency modeling. Third, we will develop a multi-objective framework that jointly optimizes fault detection rate, execution time, and resource consumption for more balanced and practical test scheduling.

Through these efforts, we aim to bridge the gap between theoretical optimization and industrial practice, contributing robust, scalable, and cost-effective solutions for test case prioritization in modern continuous integration and delivery pipelines.

Acknowledgement: Not applicable.

Funding Statement: This research was funded by Natural Science Foundation of Fujian Province, grant numbers 2023J01975, 2026J0011041, and 2026J0011042. Educational research projects of young and middle-aged teachers in Fujian Province, grant number JAT220362. Industry-University-Research Project of Longyan Nonferrous Metals Research Institute, grant number PT202502.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization and methodology, Jiali Chen and Heming Jia; experimental setup, Honghui Yi; programming, Jiheng Zhang; validation, Chong Zeng and Xiaojie Chen; formal analysis, Jiali Chen and Xiaojie Chen; investigation and data curation, Jiheng Zhang and Jiali

Chen; writing—original draft preparation, Jiali Chen; writing—review and editing, Jiali Chen; supervision, Jiali Chen and Heming Jia. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A Detailed Numerical Results on CEC2017 and CEC2020 Benchmark Suites in Manuscript

Appendix A.1 Test Results on the CEC2017 Benchmark Functions

Table A1: Test results on the CEC2017 benchmark functions.

	Metrics	MCIAO	AO	GOOSE	BKA	SSA	HHO	WOA	GWO	PSO	LSHADE	iCSPM
F_1	Mean	**9.25E+04**	4.47E+09	1.06E+09	1.29E+10	2.60E+09	4.05E+08	4.88E+09	2.54E+10	9.33E+08	2.84E+08	2.93E+09
	Std.	**9.08E+04**	1.46E+09	1.36E+09	1.15E+10	1.51E+09	2.26E+08	1.60E+09	4.68E+09	1.48E+09	2.36E+08	1.02E+09
	Min	**6.25E+04**	2.87E+09	7.02E+04	2.15E+09	8.50E+08	1.53E+08	2.78E+09	1.85E+10	4.15E+05	1.38E+07	1.71E+09
	Epoch	**4.94E+02**	4.97E+02	5.00E+02	5.00E+02	4.98E+02	5.00E+02	4.98E+02	5.00E+02	5.00E+02	5.00E+02	4.97E+02
F_2	Mean	**3.20E+04**	6.68E+04	1.37E+05	6.12E+04	5.82E+04	5.54E+04	2.78E+05	8.67E+04	5.83E+04	9.86E+04	1.41E+05
	Std.	**6.01E+03**	7.40E+03	3.89E+04	1.50E+04	7.02E+03	1.16E+04	7.53E+04	9.04E+03	2.52E+04	1.80E+04	2.02E+04
	Min	**2.12E+04**	5.45E+04	8.54E+04	3.95E+04	4.65E+04	3.35E+04	1.58E+05	7.25E+04	2.82E+04	6.72E+04	9.64E+04
	Epoch	4.77E+02	4.88E+02	5.00E+02	5.00E+02	4.97E+02	5.00E+02	5.00E+02	5.00E+02	4.99E+02	4.88E+02	**4.70E+02**
F_3	Mean	**5.12E+02**	9.96E+02	8.52E+02	1.92E+03	6.17E+02	7.08E+02	1.36E+03	3.89E+03	6.31E+02	5.94E+02	8.94E+02
	Std.	**2.65E+01**	1.97E+02	3.73E+02	3.63E+03	9.85E+01	7.87E+01	3.23E+02	9.16E+02	1.41E+02	4.71E+01	9.65E+01
	Min	**4.62E+02**	6.25E+02	5.01E+02	5.21E+02	4.68E+02	5.62E+02	7.52E+02	2.15E+03	4.72E+02	5.28E+02	7.77E+02
	Epoch	**4.96E+02**	4.96E+02	5.00E+02	5.00E+02	4.98E+02	5.00E+02	4.99E+02	5.00E+02	5.00E+02	5.00E+02	4.95E+02
F_4	Mean	**6.22E+02**	7.34E+02	8.84E+02	7.52E+02	7.77E+02	7.68E+02	8.72E+02	8.49E+02	6.68E+02	6.96E+02	8.83E+02
	Std.	2.71E+01	3.24E+01	8.48E+01	4.32E+01	4.31E+01	3.85E+01	6.90E+01	3.21E+01	3.62E+01	**1.36E+01**	3.85E+01
	Min	**5.72E+02**	6.73E+02	7.21E+02	6.71E+02	6.92E+02	6.95E+02	7.42E+02	7.88E+02	5.98E+02	6.72E+02	8.03E+02
	Epoch	4.90E+02	4.94E+02	5.00E+02	4.99E+02	4.97E+02	5.00E+02	4.95E+02	4.92E+02	5.00E+02	**4.58E+02**	4.79E+02
F_5	Mean	**6.12E+02**	6.59E+02	6.76E+02	6.62E+02	6.58E+02	6.68E+02	6.77E+02	6.76E+02	6.41E+02	6.17E+02	6.68E+02
	Std.	4.56E+00	7.23E+00	9.45E+00	9.84E+00	7.82E+00	6.90E+00	1.38E+01	5.42E+00	1.27E+01	**5.25E−01**	7.44E+00
	Min	6.03E+02	6.44E+02	6.57E+02	6.43E+02	6.43E+02	6.54E+02	6.50E+02	6.66E+02	6.16E+02	**6.00E+02**	6.50E+02
	Epoch	4.89E+02	4.82E+02	5.00E+02	4.99E+02	4.97E+02	4.99E+02	4.97E+02	5.00E+02	5.00E+02	4.99E+02	**4.78E+02**
F_6	Mean	**9.02E+02**	1.15E+03	2.51E+03	1.21E+03	1.28E+03	1.32E+03	1.33E+03	1.35E+03	9.49E+02	9.40E+02	1.35E+03
	Std.	5.05E+01	6.93E+01	7.71E+02	6.81E+01	7.15E+01	7.88E+01	8.97E+01	4.65E+01	8.08E+01	**2.23E+01**	5.61E+01
	Min	**8.01E+02**	1.02E+03	9.66E+02	1.07E+03	1.14E+03	1.16E+03	1.15E+03	1.26E+03	8.11E+02	8.90E+02	1.22E+03
	Epoch	4.85E+02	4.91E+02	5.00E+02	5.00E+02	4.98E+02	4.97E+02	4.93E+02	**4.66E+02**	5.00E+02	4.97E+02	4.89E+02
F_7	Mean	**9.00E+02**	9.86E+02	1.09E+03	1.01E+03	9.83E+02	9.83E+02	1.07E+03	1.08E+03	9.39E+02	9.86E+02	1.13E+03
	Std.	2.31E+01	2.75E+01	6.70E+01	7.04E+01	2.84E+01	2.44E+01	6.52E+01	2.89E+01	3.14E+01	**2.06E+01**	3.38E+01
	Min	**8.54E+02**	9.31E+02	9.58E+02	8.65E+02	9.26E+02	9.34E+02	9.42E+02	1.02E+03	8.76E+02	9.39E+02	1.06E+03
	Epoch	4.90E+02	4.93E+02	5.00E+02	5.00E+02	4.96E+02	5.00E+02	4.95E+02	5.00E+02	5.00E+02	**4.19E+02**	4.78E+02
F_8	Mean	2.94E+03	8.00E+03	7.85E+03	6.16E+03	5.43E+03	8.89E+03	1.27E+04	7.88E+03	4.30E+03	**9.88E+02**	1.43E+04
	Std.	**1.18E+02**	1.13E+03	2.50E+03	1.65E+03	1.51E+03	1.01E+03	4.20E+03	8.92E+02	1.49E+03	1.30E+02	2.35E+03
	Min	2.70E+03	5.82E+03	2.84E+03	3.12E+03	2.81E+03	6.89E+03	4.27E+03	6.12E+03	2.78E+03	**9.04E+02**	1.04E+04
	Epoch	4.85E+02	4.86E+02	5.00E+02	5.00E+02	4.99E+02	4.93E+02	4.93E+02	**4.55E+02**	5.00E+02	4.89E+02	4.58E+02
F_9	Mean	**4.96E+03**	6.33E+03	5.91E+03	5.29E+03	5.43E+03	6.19E+03	7.70E+03	7.29E+03	5.02E+03	8.59E+03	5.59E+03
	Std.	1.28E+03	7.45E+02	8.14E+02	7.08E+02	5.91E+02	7.75E+02	6.27E+02	4.66E+02	7.56E+02	**3.70E+02**	3.81E+02
	Min	**2.41E+03**	4.84E+03	4.27E+03	3.87E+03	4.25E+03	4.64E+03	6.44E+03	6.36E+03	3.51E+03	7.36E+03	4.88E+03
	Epoch	4.81E+02	4.91E+02	5.00E+02	4.99E+02	4.96E+02	4.99E+02	4.92E+02	4.68E+02	5.00E+02	**3.67E+02**	4.77E+02

(Continued)

Table A1 (continued)

	Metrics	MCIAO	AO	GOOSE	BKA	SSA	HHO	WOA	GWO	PSO	LSHADE	iCSPM
F_{10}	Mean	**1.30E+03**	4.51E+03	1.92E+03	1.75E+03	2.48E+03	1.59E+03	1.05E+04	8.22E+03	1.36E+03	1.37E+03	3.09E+03
	Std.	**6.73E+01**	1.82E+03	8.90E+02	1.65E+03	1.22E+03	2.01E+02	3.57E+03	2.97E+03	1.70E+02	7.92E+01	7.10E+02
	Min	**1.17E+03**	1.42E+03	1.24E+03	1.19E+03	1.24E+03	1.19E+03	3.37E+03	2.28E+03	1.17E+03	1.17E+03	1.93E+03
	Epoch	4.86E+02	4.94E+02	5.00E+02	5.00E+02	5.00E+02	5.00E+02	4.99E+02	5.00E+02	5.00E+02	**4.76E+02**	4.97E+02
F_{11}	Mean	**2.57E+06**	3.07E+08	3.25E+07	3.57E+08	1.92E+08	6.50E+07	5.90E+08	2.45E+09	2.19E+08	2.67E+06	1.03E+08
	Std.	**2.01E+06**	2.30E+08	5.40E+07	1.17E+09	3.81E+08	4.52E+07	3.31E+08	8.82E+08	4.84E+08	2.55E+06	4.00E+07
	Min	3.52E+05	5.45E+06	6.25E+05	6.45E+06	7.42E+06	8.25E+06	1.45E+07	6.85E+08	**3.43E+05**	3.70E+05	4.25E+07
	Epoch	**4.94E+02**	4.96E+02	5.00E+02	5.00E+02	4.98E+02	5.00E+02	4.98E+02	5.00E+02	5.00E+02	4.99E+02	4.96E+02
F_{12}	Mean	**2.33E+04**	2.38E+07	1.17E+05	1.07E+08	4.99E+07	1.08E+06	8.56E+06	5.53E+08	9.45E+05	7.32E+04	1.64E+07
	Std.	**1.62E+04**	6.62E+07	6.15E+04	3.38E+08	9.74E+07	5.59E+05	6.17E+06	4.14E+08	1.81E+06	3.99E+04	1.18E+07
	Min	**1.50E+04**	1.25E+06	3.52E+04	4.25E+06	2.51E+06	3.52E+05	1.43E+06	1.85E+07	1.43E+05	1.70E+04	1.85E+06
	Epoch	4.92E+02	4.90E+02	5.00E+02	5.00E+02	4.97E+02	5.00E+02	4.94E+02	5.00E+02	5.00E+02	**4.81E+02**	4.98E+02
F_{13}	Mean	6.63E+05	1.26E+06	9.72E+04	4.38E+04	1.10E+05	1.06E+06	2.87E+06	1.58E+06	6.04E+04	**1.29E+04**	2.16E+05
	Std.	7.81E+05	1.05E+06	1.15E+05	1.50E+05	9.78E+04	1.11E+06	5.05E+06	1.28E+06	7.81E+04	**1.98E+04**	1.22E+05
	Min	**5.12E+02**	1.43E+05	8.51E+03	1.25E+03	1.25E+04	8.52E+04	2.15E+05	1.25E+05	8.55E+02	1.77E+03	3.45E+04
	Epoch	**4.51E+02**	4.75E+02	5.00E+02	4.99E+02	4.97E+02	5.00E+02	4.98E+02	4.99E+02	5.00E+02	4.66E+02	4.92E+02
F_{14}	Mean	**8.64E+03**	1.98E+05	7.24E+04	2.86E+06	2.53E+06	1.33E+05	1.23E+07	1.22E+08	1.45E+04	1.66E+04	2.56E+05
	Std.	**7.75E+03**	1.19E+05	6.68E+04	1.52E+07	6.46E+06	7.82E+04	3.01E+07	9.98E+07	1.15E+04	1.04E+04	1.41E+05
	Min	**3.01E+03**	4.52E+04	1.25E+04	1.43E+05	8.54E+04	3.52E+04	1.43E+06	8.54E+06	5.25E+03	3.53E+03	2.44E+04
	Epoch	4.92E+02	4.78E+02	5.00E+02	4.99E+02	4.97E+02	4.97E+02	4.97E+02	5.00E+02	5.00E+02	**4.73E+02**	4.98E+02
F_{15}	Mean	2.68E+03	3.53E+03	3.71E+03	3.12E+03	3.08E+03	3.72E+03	4.30E+03	3.97E+03	**2.68E+03**	3.15E+03	3.00E+03
	Std.	3.42E+02	3.04E+02	6.15E+02	3.71E+02	3.65E+02	5.79E+02	5.02E+02	4.05E+02	3.15E+02	3.48E+02	**1.97E+02**
	Min	**1.85E+03**	2.95E+03	2.54E+03	2.45E+03	2.39E+03	2.65E+03	3.25E+03	3.13E+03	2.01E+03	2.59E+03	2.49E+03
	Epoch	4.82E+02	4.83E+02	5.00E+02	4.99E+02	4.93E+02	5.00E+02	4.88E+02	4.92E+02	4.99E+02	**3.75E+02**	4.94E+02
F_{16}	Mean	**2.07E+03**	2.46E+03	2.83E+03	2.45E+03	2.59E+03	2.68E+03	2.71E+03	2.74E+03	2.38E+03	2.15E+03	2.30E+03
	Std.	**1.97E+02**	2.95E+02	3.73E+02	2.85E+02	3.16E+02	3.25E+02	2.97E+02	3.99E+02	2.53E+02	2.51E+02	2.14E+02
	Min	**1.75E+03**	2.02E+03	2.35E+03	2.12E+03	2.20E+03	2.29E+03	2.34E+03	2.25E+03	1.99E+03	1.90E+03	2.11E+03
	Epoch	4.72E+02	4.80E+02	5.00E+02	4.94E+02	4.96E+02	4.99E+02	4.86E+02	4.83E+02	5.00E+02	**4.08E+02**	4.88E+02
F_{17}	Mean	1.83E+06	7.08E+06	1.01E+06	2.39E+05	9.76E+05	3.07E+06	1.09E+07	1.07E+07	**2.15E+05**	1.15E+06	9.29E+05
	Std.	1.56E+06	8.45E+06	7.44E+05	4.95E+05	9.39E+05	3.77E+06	1.48E+07	8.72E+06	**1.87E+05**	7.07E+05	6.00E+05
	Min	5.64E+03	1.25E+04	6.52E+03	8.51E+03	**4.52E+03**	1.25E+04	3.25E+04	4.25E+04	4.68E+03	2.46E+05	1.52E+05
	Epoch	4.70E+02	4.86E+02	5.00E+02	4.99E+02	4.97E+02	5.00E+02	4.97E+02	4.99E+02	4.99E+02	**4.65E+02**	4.91E+02
F_{18}	Mean	1.75E+06	5.45E+06	1.39E+06	2.42E+05	9.56E+03	3.77E+06	1.67E+07	3.02E+07	1.71E+04	**7.86E+03**	6.39E+05
	Std.	1.51E+06	4.70E+06	8.33E+05	2.86E+05	1.27E+04	7.80E+06	1.52E+07	1.61E+07	2.64E+04	**5.39E+03**	3.44E+05
	Min	1.55E+04	1.52E+05	6.52E+05	1.55E+05	2.45E+04	2.15E+05	5.41E+05	8.54E+05	1.41E+04	**2.50E+03**	9.62E+04
	Epoch	4.81E+02	4.88E+02	5.00E+02	4.99E+02	4.95E+02	5.00E+02	4.93E+02	5.00E+02	5.00E+02	**4.74E+02**	4.96E+02
F_{19}	Mean	**2.46E+03**	2.64E+03	3.04E+03	2.59E+03	2.82E+03	2.77E+03	2.91E+03	2.86E+03	2.66E+03	2.64E+03	2.75E+03
	Std.	1.80E+02	**1.61E+02**	2.91E+02	1.90E+02	2.72E+02	2.16E+02	2.38E+02	1.99E+02	2.22E+02	1.83E+02	1.93E+02
	Min	**2.11E+03**	2.34E+03	2.52E+03	2.25E+03	2.39E+03	2.41E+03	2.51E+03	2.50E+03	2.29E+03	2.36E+03	2.45E+03
	Epoch	**4.66E+02**	4.86E+02	4.99E+02	4.99E+02	4.97E+02	4.99E+02	4.84E+02	4.85E+02	5.00E+02	4.68E+02	4.72E+02
F_{20}	Mean	**2.40E+03**	2.51E+03	2.68E+03	2.54E+03	2.54E+03	2.60E+03	2.65E+03	2.62E+03	2.46E+03	2.48E+03	2.60E+03
	Std.	**3.06E+01**	5.37E+01	6.48E+01	5.46E+01	5.90E+01	5.63E+01	6.80E+01	4.54E+01	4.16E+01	3.66E+01	1.09E+02
	Min	**2.32E+03**	2.41E+03	2.51E+03	2.42E+03	2.42E+03	2.49E+03	2.51E+03	2.52E+03	2.37E+03	2.45E+03	2.32E+03
	Epoch	4.85E+02	4.92E+02	5.00E+02	5.00E+02	4.98E+02	5.00E+02	4.97E+02	4.85E+02	5.00E+02	**3.67E+02**	4.84E+02
F_{21}	Mean	3.67E+03	6.09E+03	7.61E+03	6.14E+03	5.92E+03	7.72E+03	8.13E+03	6.97E+03	4.95E+03	**2.92E+03**	6.86E+03
	Std.	9.61E+02	2.47E+03	**7.38E+02**	1.51E+03	2.35E+03	1.01E+03	2.00E+03	1.77E+03	1.95E+03	1.84E+03	9.52E+02
	Min	**1.93E+03**	2.85E+03	6.25E+03	3.51E+03	2.15E+03	5.84E+03	4.51E+03	3.85E+03	2.15E+03	2.33E+03	3.54E+03
	Epoch	4.87E+02	4.95E+02	4.99E+02	4.99E+02	4.97E+02	5.00E+02	4.94E+02	**4.15E+02**	5.00E+02	4.84E+02	4.81E+02

(Continued)

Table A1 (continued)

	Metrics	MCIAO	AO	GOOSE	BKA	SSA	HHO	WOA	GWO	PSO	LSHADE	iCSPM
F_{22}	Mean	**2.79E+03**	3.03E+03	3.51E+03	3.09E+03	2.96E+03	3.23E+03	3.14E+03	3.19E+03	3.03E+03	2.84E+03	3.27E+03
	Std.	**6.55E+01**	7.79E+01	1.76E+02	9.62E+01	1.03E+02	1.32E+02	1.03E+02	7.85E+01	1.18E+02	7.25E+01	9.14E+01
	Min	**2.61E+03**	2.85E+03	3.12E+03	2.87E+03	2.73E+03	2.95E+03	2.89E+03	3.01E+03	2.79E+03	2.76E+03	3.12E+03
	Epoch	4.70E+02	4.85E+02	5.00E+02	5.00E+02	4.96E+02	5.00E+02	4.97E+02	**4.49E+02**	5.00E+02	4.95E+02	4.88E+02
F_{23}	Mean	**2.98E+03**	3.14E+03	3.70E+03	3.25E+03	3.16E+03	3.52E+03	3.26E+03	3.31E+03	3.17E+03	3.03E+03	3.26E+03
	Std.	7.52E+01	**5.61E+01**	1.13E+02	9.90E+01	1.05E+02	1.73E+02	9.62E+01	7.49E+01	1.22E+02	8.14E+01	1.67E+02
	Min	**2.75E+03**	3.01E+03	3.42E+03	3.03E+03	2.91E+03	3.15E+03	3.05E+03	3.15E+03	2.89E+03	2.97E+03	2.91E+03
	Epoch	4.84E+02	4.88E+02	5.00E+02	5.00E+02	4.97E+02	5.00E+02	4.99E+02	**4.19E+02**	4.96E+02	4.76E+02	4.93E+02
F_{24}	Mean	**2.90E+03**	3.08E+03	3.01E+03	3.10E+03	3.02E+03	3.00E+03	3.23E+03	3.60E+03	2.97E+03	2.97E+03	3.11E+03
	Std.	**1.80E+01**	6.79E+01	7.87E+01	1.00E+02	5.19E+01	3.33E+01	9.72E+01	1.74E+02	4.41E+01	4.10E+01	3.67E+01
	Min	2.90E+03	2.91E+03	**2.87E+03**	2.89E+03	2.91E+03	2.92E+03	3.01E+03	3.15E+03	2.88E+03	2.91E+03	3.05E+03
	Epoch	**4.95E+02**	4.96E+02	5.00E+02	5.00E+02	4.99E+02	5.00E+02	4.97E+02	5.00E+02	5.00E+02	4.99E+02	4.96E+02
F_{25}	Mean	**5.00E+03**	6.53E+03	1.14E+04	8.30E+03	6.55E+03	8.26E+03	8.53E+03	8.47E+03	5.68E+03	5.05E+03	5.98E+03
	Std.	**4.63E+02**	1.26E+03	2.22E+03	1.26E+03	1.34E+03	1.26E+03	1.44E+03	1.05E+03	1.32E+03	8.04E+02	1.33E+03
	Min	4.13E+03	4.13E+03	6.54E+03	5.41E+03	3.85E+03	5.41E+03	5.54E+03	6.25E+03	4.25E+03	**2.99E+03**	4.32E+03
	Epoch	4.87E+02	4.93E+02	5.00E+02	5.00E+02	4.98E+02	5.00E+02	4.97E+02	4.96E+02	5.00E+02	**4.43E+02**	4.94E+02
F_{26}	Mean	3.27E+03	3.43E+03	4.14E+03	3.42E+03	3.32E+03	3.56E+03	3.44E+03	3.55E+03	3.35E+03	**3.23E+03**	3.40E+03
	Std.	3.43E+01	6.71E+01	4.60E+02	1.22E+02	7.03E+01	2.01E+02	1.20E+02	1.01E+02	6.68E+01	3.69E+01	4.08E+01
	Min	3.20E+03	3.29E+03	3.21E+03	3.20E+03	3.19E+03	**3.15E+03**	3.19E+03	3.32E+03	3.21E+03	3.21E+03	3.32E+03
	Epoch	**4.76E+02**	4.85E+02	4.99E+02	4.99E+02	4.94E+02	5.00E+02	4.88E+02	5.00E+02	4.94E+02	4.98E+02	4.94E+02
F_{27}	Mean	**3.26E+03**	3.87E+03	3.56E+03	3.68E+03	3.49E+03	3.51E+03	3.98E+03	5.05E+03	3.39E+03	3.41E+03	3.63E+03
	Std.	**2.48E+01**	2.27E+02	3.29E+02	5.20E+02	2.09E+02	1.03E+02	2.92E+02	2.56E+02	1.83E+02	8.07E+01	1.07E+02
	Min	**3.09E+03**	3.41E+03	3.21E+03	3.21E+03	3.09E+03	3.29E+03	3.39E+03	4.51E+03	3.15E+03	3.28E+03	3.43E+03
	Epoch	**4.92E+02**	4.97E+02	5.00E+02	5.00E+02	4.99E+02	5.00E+02	4.98E+02	5.00E+02	5.00E+02	4.99E+02	4.97E+02
F_{28}	Mean	**3.93E+03**	4.89E+03	5.35E+03	4.85E+03	4.28E+03	4.89E+03	5.35E+03	5.27E+03	4.28E+03	3.96E+03	4.37E+03
	Std.	1.73E+02	4.02E+02	5.53E+02	5.76E+02	3.15E+02	3.99E+02	5.74E+02	4.22E+02	3.52E+02	1.80E+02	**1.47E+02**
	Min	**3.58E+03**	4.09E+03	4.24E+03	3.72E+03	3.65E+03	4.08E+03	4.21E+03	4.42E+03	3.61E+03	3.74E+03	4.13E+03
	Epoch	4.74E+02	4.84E+02	5.00E+02	4.91E+02	4.97E+02	5.00E+02	4.87E+02	5.00E+02	5.00E+02	**4.16E+02**	4.89E+02
F_{29}	Mean	**2.35E+04**	3.26E+07	5.86E+06	1.26E+07	1.22E+07	7.99E+06	6.86E+07	1.21E+08	2.77E+05	9.98E+04	3.60E+06
	Std.	**9.99E+03**	2.19E+07	4.44E+06	4.08E+07	9.61E+06	8.05E+06	6.16E+07	8.89E+07	3.47E+05	7.11E+04	1.99E+06
	Min	1.99E+04	1.03E+06	1.25E+05	2.15E+05	1.52E+06	3.25E+05	2.51E+06	5.12E+06	2.55E+04	**1.65E+04**	1.61E+06
	Epoch	4.87E+02	4.89E+02	5.00E+02	4.99E+02	4.97E+02	5.00E+02	4.98E+02	5.00E+02	4.99E+02	**4.79E+02**	4.96E+02

Note: The best results are highlighted in bold.

Appendix A.2 Test Results on the CEC2020 Benchmark Functions

Table A2: Test results on the CEC2020 benchmark functions.

	Metrics	MCIAO	AO	GOOSE	BKA	SSA	HHO	WOA	GWO	PSO	LSHADE	iCSPM
F_1	Mean	**4.99E+03**	3.69E+08	5.48E+07	5.93E+08	1.84E+08	4.37E+06	2.82E+08	4.60E+09	1.81E+07	9.13E+04	2.34E+07
	Std.	**6.85E+03**	2.38E+08	1.92E+08	2.04E+09	3.65E+08	1.46E+06	2.15E+08	1.51E+09	9.93E+07	2.71E+05	5.97E+06
	Min	**4.03E+03**	1.25E+06	6.52E+04	1.25E+06	2.45E+05	1.54E+06	1.45E+06	2.15E+09	1.25E+04	9.21E+03	1.22E+07
	Epoch	**4.91E+02**	4.96E+02	5.00E+02	5.00E+02	4.98E+02	5.00E+02	4.97E+02	5.00E+02	5.00E+02	5.00E+02	5.00E+02
F_2	Mean	2.20E+03	2.76E+03	3.38E+03	2.55E+03	2.67E+03	2.64E+03	3.28E+03	3.20E+03	2.30E+03	2.91E+03	**2.07E+03**
	Std.	3.13E+02	4.75E+02	6.27E+02	4.55E+02	4.57E+02	5.03E+02	5.45E+02	3.87E+02	4.89E+02	4.09E+02	**1.86E+02**
	Min	1.65E+03	1.95E+03	2.15E+03	1.75E+03	1.85E+03	1.79E+03	2.25E+03	2.51E+03	1.69E+03	2.03E+03	**1.62E+03**
	Epoch	**4.16E+02**	4.85E+02	4.99E+02	4.99E+02	4.97E+02	4.99E+02	4.90E+02	4.84E+02	4.25E+02	4.84E+02	4.25E+02

(Continued)

Table A2 (continued)

	Metrics	MCIAO	AO	GOOSE	BKA	SSA	HHO	WOA	GWO	PSO	LSHADE	iCSPM
F_3	Mean	**7.51E+02**	8.26E+02	1.39E+03	8.56E+02	8.96E+02	8.82E+02	8.90E+02	9.13E+02	7.65E+02	7.62E+02	7.87E+02
	Std.	1.24E+01	2.87E+01	2.72E+02	4.55E+01	3.88E+01	2.66E+01	5.10E+01	3.14E+01	2.02E+01	**8.91E+00**	1.46E+01
	Min	**7.32E+02**	7.81E+02	8.54E+02	7.81E+02	8.25E+02	8.35E+02	8.12E+02	8.61E+02	7.35E+02	7.34E+02	7.52E+02
	Epoch	**4.01E+02**	4.89E+02	5.00E+02	4.99E+02	4.99E+02	4.99E+02	4.90E+02	4.35E+02	4.22E+02	4.35E+02	4.02E+02
F_4	Mean	1.91E+03	1.98E+03	2.01E+03	5.64E+03	1.92E+03	1.92E+03	2.38E+03	1.30E+04	**1.91E+03**	1.91E+03	1.91E+03
	Std.	2.98E+00	7.80E+01	9.83E+01	1.42E+04	4.06E+01	7.22E+00	1.09E+03	2.36E+04	1.60E+01	3.56E+00	**2.38E+00**
	Min	1.90E+03	1.85E+03	1.81E+03	2.45E+00	1.85E+03	1.91E+03	1.24E+03	**1.25E+00**	1.90E+03	1.90E+03	1.91E+03
	Epoch	**4.73E+02**	4.78E+02	5.00E+02	5.00E+02	4.97E+02	5.00E+02	4.93E+02	4.81E+02	5.00E+02	4.81E+02	5.00E+02
F_5	Mean	**1.03E+04**	1.63E+06	2.02E+05	5.68E+05	2.00E+05	7.29E+05	3.97E+06	4.74E+06	8.57E+04	1.38E+04	3.59E+05
	Std.	2.21E+04	1.65E+06	2.31E+05	7.30E+05	1.98E+05	5.45E+05	3.23E+06	9.07E+06	1.19E+05	**2.53E+03**	2.07E+05
	Min	5.13E+03	1.25E+04	8.54E+03	1.25E+04	**1.25E+03**	3.52E+04	1.25E+05	2.45E+05	7.52E+03	5.19E+03	5.02E+04
	Epoch	**4.53E+02**	4.78E+02	5.00E+02	5.00E+02	4.98E+02	5.00E+02	4.95E+02	4.98E+02	5.00E+02	4.98E+02	5.00E+02
F_6	Mean	**1.86E+03**	2.02E+03	2.44E+03	1.95E+03	2.07E+03	2.07E+03	2.14E+03	2.13E+03	1.92E+03	1.87E+03	1.90E+03
	Std.	1.46E+02	1.89E+02	3.67E+02	1.61E+02	1.66E+02	1.88E+02	2.01E+02	1.67E+02	1.32E+02	1.95E+02	**8.31E+01**
	Min	**1.58E+03**	1.65E+03	1.84E+03	1.69E+03	1.75E+03	1.72E+03	1.79E+03	1.80E+03	1.65E+03	1.68E+03	1.76E+03
	Epoch	4.84E+02	4.86E+02	4.99E+02	5.00E+02	4.90E+02	5.00E+02	4.96E+02	**4.79E+02**	4.91E+02	4.89E+02	4.91E+02
F_7	Mean	**1.02E+04**	2.16E+05	1.98E+05	5.79E+05	7.06E+04	7.06E+05	6.27E+06	6.54E+06	3.95E+04	1.25E+04	1.59E+05
	Std.	1.50E+04	2.09E+05	5.21E+05	1.71E+06	7.84E+04	4.67E+05	6.64E+06	2.64E+06	3.03E+04	**3.51E+02**	1.15E+05
	Min	**8.71E+03**	1.85E+04	1.54E+04	1.22E+05	8.92E+04	3.52E+05	3.15E+05	4.25E+05	2.54E+04	9.15E+03	3.00E+04
	Epoch	**4.67E+02**	4.77E+02	5.00E+02	4.99E+02	4.97E+02	5.00E+02	5.00E+02	4.99E+02	5.00E+02	4.99E+02	5.00E+02
F_8	Mean	2.62E+03	**2.33E+03**	3.56E+03	2.74E+03	2.70E+03	2.94E+03	3.24E+03	2.72E+03	2.71E+03	2.90E+03	2.33E+03
	Std.	6.54E+02	2.01E+01	1.38E+03	6.91E+02	7.72E+02	9.08E+02	1.23E+03	1.86E+02	7.40E+02	**1.43E+01**	3.31E+01
	Min	2.11E+03	2.29E+03	2.76E+03	**1.91E+03**	2.38E+03	2.36E+03	2.54E+03	2.39E+03	2.17E+03	2.23E+03	2.27E+03
	Epoch	**4.16E+02**	4.92E+02	4.99E+02	4.99E+02	4.99E+02	5.00E+02	4.95E+02	4.99E+02	4.27E+02	4.99E+02	4.17E+02
F_9	Mean	**2.82E+03**	2.87E+03	3.27E+03	2.91E+03	2.83E+03	2.98E+03	2.90E+03	2.94E+03	2.88E+03	2.88E+03	3.00E+03
	Std.	**2.04E+01**	3.28E+01	2.60E+02	1.08E+02	9.23E+01	1.18E+02	6.57E+01	3.77E+01	7.69E+01	2.15E+01	3.64E+01
	Min	**2.62E+03**	2.81E+03	2.75E+03	2.70E+03	2.64E+03	2.74E+03	2.77E+03	2.87E+03	2.73E+03	2.79E+03	2.74E+03
	Epoch	**3.98E+02**	4.87E+02	5.00E+02	4.99E+02	4.98E+02	5.00E+02	4.99E+02	4.72E+02	4.17E+02	4.72E+02	4.17E+02
F_{10}	Mean	**2.96E+03**	3.14E+03	3.12E+03	3.14E+03	3.12E+03	3.12E+03	3.23E+03	3.73E+03	3.06E+03	3.07E+03	3.03E+03
	Std.	6.15E+01	**5.81E+01**	1.09E+02	1.86E+02	9.06E+01	5.93E+01	6.02E+01	4.47E+02	1.14E+02	7.06E+01	6.40E+01
	Min	2.84E+03	3.02E+03	2.90E+03	**2.77E+03**	2.94E+03	3.00E+03	3.11E+03	2.84E+03	2.83E+03	2.92E+03	2.81E+03
	Epoch	**4.53E+02**	4.93E+02	5.00E+02	5.00E+02	4.99E+02	5.00E+02	4.96E+02	5.00E+02	4.65E+02	5.00E+02	4.65E+02

Note: The best results are highlighted in bold.

Appendix B Detailed Numerical Results for Wilcoxon Rank-sum Test

Appendix B.1 Wilcoxon Rank-sum Test Results on the CEC2017 Benchmark Functions

Table A3: Wilcoxon rank-sum test results of MCIAO and other compared algorithms on CEC2017.

	AO	GOOSE	BKA	SSA	HHO	WOA	GWO	PSO	LSHADE	iCSPM
F_1	1.86E−09	1.86E−09	3.90E−05	1.86E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09
F_2	2.61E−08	2.55E−07	1.86E−09	3.73E−09	1.86E−09	1.86E−09	2.61E−08	1.22E−05	7.30E−04	1.86E−09
F_3	1.86E−09	1.86E−09	1.53E−04	1.86E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09
F_4	3.73E−09	4.05E−02	1.30E−08	2.61E−08	6.15E−08	**1.58E−01**	1.86E−09	1.86E−09	1.86E−09	1.34E−03
F_5	1.86E−09	**8.71E−01**	6.91E−07	9.31E−09	2.69E−05	**7.92E−01**	1.86E−09	1.86E−09	1.86E−09	6.29E−05
F_6	1.86E−09	1.02E−07	1.86E−09	4.41E−05	**2.05E−01**	**3.39E−01**	1.86E−09	1.86E−09	1.86E−09	**9.03E−01**
F_7	1.86E−09	**4.28E−01**	2.35E−06	1.86E−09	3.73E−09	**4.52E−01**	1.86E−09	1.86E−09	1.86E−09	2.35E−06
F_8	**7.61E−01**	**8.08E−01**	1.40E−05	1.86E−09	3.45E−04	1.19E−06	1.86E−09	2.61E−08	1.86E−09	1.86E−09

(Continued)

Table A3 (continued)

	AO	GOOSE	BKA	SSA	HHO	WOA	GWO	PSO	LSHADE	iCSPM
F_9	1.68E−06	3.54E−08	5.59E−09	1.86E−09	5.59E−09	5.01E−03	2.61E−08	3.73E−09	1.86E−09	1.86E−09
F_{10}	2.61E−08	1.86E−09	1.30E−08	1.86E−09	1.86E−09	4.34E−03	3.73E−09	1.86E−09	1.86E−09	1.86E−09
F_{11}	1.86E−09	1.86E−09	1.30E−07	1.86E−09	1.86E−09	1.86E−09	1.86E−09	5.59E−09	1.86E−09	1.86E−09
F_{12}	1.86E−09	1.86E−09	1.22E−05	1.86E−09	1.86E−09	1.86E−09	2.61E−08	1.86E−09	1.86E−09	3.73E−09
F_{13}	**3.18E−01**	1.30E−08	1.02E−07	9.31E−09	**1.29E−01**	**3.49E−01**	2.56E−04	1.86E−09	1.86E−09	8.01E−08
F_{14}	1.86E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09	1.42E−06	1.86E−09	1.86E−09	1.86E−09	1.86E−09
F_{15}	1.19E−06	4.73E−02	2.55E−07	1.86E−08	**7.67E−02**	9.93E−03	1.86E−09	1.86E−09	5.59E−09	1.86E−09
F_{16}	2.56E−03	**1.77E−01**	4.66E−03	**1.05E−01**	**6.55E−01**	**6.12E−01**	3.54E−08	4.41E−05	8.01E−08	8.33E−07
F_{17}	**1.14E−01**	1.86E−09	1.86E−09	9.31E−09	9.22E−06	**5.98E−01**	3.15E−07	1.86E−09	3.73E−09	1.86E−09
F_{18}	3.73E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09	2.56E−03	8.01E−08	1.86E−09	1.86E−09	1.86E−09
F_{19}	9.22E−06	9.93E−03	9.22E−06	**7.61E−01**	**1.98E−01**	**2.99E−01**	9.98E−07	9.52E−04	1.22E−05	3.74E−03
F_{20}	1.30E−08	5.55E−04	6.91E−07	3.79E−06	4.97E−02	2.93E−02	1.86E−09	1.86E−09	1.86E−09	**7.77E−01**
F_{21}	4.66E−08	**6.99E−02**	**6.99E−02**	**6.36E−02**	4.05E−02	2.77E−02	**1.84E−01**	1.46E−03	3.15E−07	**7.61E−01**
F_{22}	2.55E−07	1.86E−09	6.29E−05	3.54E−08	**8.41E−02**	2.77E−02	1.86E−09	5.14E−06	1.86E−09	7.98E−04
F_{23}	9.31E−09	1.86E−09	1.97E−02	9.22E−06	9.98E−07	3.64E−02	1.86E−09	1.60E−05	1.86E−09	**4.65E−01**
F_{24}	1.86E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09	5.59E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09
F_{25}	3.24E−06	1.19E−06	**2.89E−01**	2.35E−06	**3.71E−01**	**6.55E−01**	3.73E−09	3.73E−09	3.73E−09	6.15E−08
F_{26}	5.97E−06	1.30E−08	7.99E−06	3.73E−09	**9.19E−01**	1.46E−03	1.86E−09	9.31E−09	1.86E−09	4.66E−08
F_{27}	1.86E−09	1.86E−09	5.59E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09
F_{28}	2.77E−03	**7.15E−01**	3.45E−04	5.59E−09	2.83E−04	**4.65E−01**	1.86E−09	1.86E−09	1.86E−09	5.59E−09
F_{29}	1.68E−06	1.86E−09	8.33E−07	1.86E−09	1.86E−09	6.19E−03	1.86E−09	1.86E−09	1.86E−09	1.86E−09

Note: The best results are highlighted in bold.

Appendix B.2 Wilcoxon Rank-Sum Test Results on the CEC2020 Benchmark Functions

Table A4: Wilcoxon rank-sum test results of MCIAO and other compared algorithms on CEC2020.

	AO	GOOSE	BKA	SSA	HHO	WOA	GWO	PSO	LSHADE	iCSPM
F_1	1.86E−09	1.86E−09	1.42E−06	1.86E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09	1.86E−09
F_2	2.32E−04	**3.28E−01**	3.79E−06	4.97E−05	7.06E−05	**3.28E−01**	2.61E−08	1.86E−08	1.85E−02	1.86E−09
F_3	1.86E−09	1.30E−08	1.82E−05	**9.19E−02**	3.13E−04	1.85E−02	1.86E−09	1.86E−09	1.86E−09	1.86E−09
F_4	3.73E−09	3.73E−09	1.23E−04	1.86E−09	1.86E−09	4.42E−06	1.86E−09	1.86E−09	1.86E−09	1.86E−09
F_5	1.28E−02	3.24E−06	1.86E−09	3.15E−07	3.13E−04	**5.16E−01**	1.23E−04	2.55E−07	1.86E−09	7.91E−05
F_6	**1.35E−01**	6.67E−04	5.05E−04	**2.62E−01**	**2.45E−01**	**9.03E−01**	2.05E−07	6.92E−06	4.71E−07	4.71E−07
F_7	3.73E−09	1.30E−08	1.86E−09	1.86E−09	3.73E−09	**3.39E−01**	2.05E−07	1.86E−09	1.86E−09	1.86E−09
F_8	1.86E−09	4.73E−02	**6.12E−01**	**3.28E−01**	**7.15E−01**	**3.93E−01**	**5.49E−02**	**3.49E−01**	1.86E−09	1.86E−09
F_9	3.86E−07	1.60E−05	**2.37E−01**	1.86E−09	**8.79E−02**	2.02E−03	1.86E−09	1.11E−04	1.86E−09	1.86E−09
F_{10}	1.30E−08	1.86E−09	9.31E−09	3.73E−09	3.73E−09	1.02E−07	9.31E−09	1.86E−09	3.73E−09	1.86E−09

Note: The best results are highlighted in bold.

Appendix C Detailed Numerical Results for Ablation Study

Appendix C.1 Test Results on the CEC2020 Benchmark Functions

Table A5: Comparison of the results of the ablation simulation experiments for CEC2020.

	Metrics	MCIAO	AO-1	AO-2	AO-3	AO-4	AO
F_1	Mean	**1.84E+08**	3.45E+08	4.25E+08	5.13E+09	3.35E+08	2.79E+08
	Std.	**1.32E+08**	2.72E+08	3.48E+08	1.93E+09	2.56E+08	3.65E+08
F_2	Mean	**2.20E+03**	2.58E+03	2.76E+03	2.62E+03	2.68E+03	2.71E+03
	Std.	**2.13E+02**	4.68E+02	3.22E+02	2.50E+02	3.76E+02	3.55E+02
F_3	Mean	**7.65E+02**	8.17E+02	8.15E+02	8.17E+02	8.21E+02	8.19E+02
	Std.	**2.02E+01**	2.83E+01	3.29E+01	2.32E+01	2.24E+01	2.51E+01
F_4	Mean	**1.92E+03**	1.99E+03	1.99E+03	2.07E+03	2.00E+03	1.99E+03
	Std.	**4.06E+01**	9.65E+01	1.78E+02	2.73E+02	1.11E+02	2.47E+02
F_5	Mean	**5.68E+05**	1.04E+06	7.97E+05	3.79E+06	1.39E+06	1.31E+06
	Std.	7.30E+05	7.39E+05	**4.17E+05**	4.14E+06	1.11E+06	1.31E+06
F_6	Mean	**1.86E+03**	2.08E+03	2.01E+03	2.06E+03	1.95E+03	2.00E+03
	Std.	**1.46E+02**	2.27E+02	1.90E+02	1.58E+02	1.92E+02	1.67E+02
F_7	Mean	5.79E+05	2.65E+05	2.35E+05	3.98E+06	2.84E+05	**1.78E+05**
	Std.	1.71E+06	3.29E+05	2.67E+05	2.99E+06	3.48E+05	**1.62E+05**
F_8	Mean	**2.33E+03**	2.62E+03	2.33E+03	2.34E+03	2.33E+03	2.33E+03
	Std.	**1.03E+01**	6.54E+02	1.85E+01	1.61E+01	1.68E+01	1.73E+01
F_9	Mean	**2.82E+03**	2.87E+03	2.87E+03	2.88E+03	2.86E+03	2.87E+03
	Std.	**2.04E+01**	2.89E+01	2.93E+01	3.43E+01	2.64E+01	2.18E+01
F_{10}	Mean	**3.11E+03**	3.12E+03	3.13E+03	3.18E+03	3.12E+03	3.19E+03
	Std.	**5.27E+01**	6.15E+01	6.20E+01	6.36E+01	6.04E+01	5.76E+01

Note: The best results are highlighted in bold.

References

1. Singh M, Chauhan N, Popli R. Test case reduction and SWOA optimization for distributed agile software development using regression testing. Multimed Tools Appl. 2025;84(10):7065–90. doi:10.1007/s11042-024-19148-1.
2. Villoth JP, Zivkovic M, Zivkovic T, Abdel-salam M, Hammad M, Jovanovic L, et al. Two-tier deep and machine learning approach optimized by adaptive multi-population firefly algorithm for software defects prediction. Neurocomputing. 2025;630(7):129695. doi:10.1016/j.neucom.2025.129695.
3. Petrovic A, Jovanovic L, Bacanin N, Antonijevic M, Savanovic N, Zivkovic M, et al. Exploring metaheuristic optimized machine learning for software defect detection on natural language and classical datasets. Mathematics. 2024;12(18):2918. doi:10.3390/math12182918.
4. Hunko I. Adaptive approaches to software testing with embedded artificial intelligence in dynamic environments. Int J Curr Sci Res Rev. 2025;8(5):2036–51. doi:10.47191/ijcsrr/v8-i5-10.
5. Elbaum S, Malishevsky AG, Rothermel G. Prioritizing test cases for regression testing. In: Proceedings of the 2000 ACM SIGSOFT International Symposium on Software Testing and Analysis; 2000 Aug 21–24; Portland, OR, USA. p. 102–12.

6. Huynh H, Pham N, Nguyen TN, Nguyen V. Segment-based test case prioritization: a multi-objective approach. In: Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis; 2024 Sep 16–20; Vienna, Austria. p. 1149–60.
7. Khatibsyarbini M, Isa MA, Abang Jawawi DN. A hybrid weight-based and string distances using particle swarm optimization for prioritizing test cases. J Theor Appl Inf Technol. 2017;95(12):2723–32. doi:10.1166/asl.2018.12918.
8. Walcott KR, Soffa ML, Kapfhammer GM, Roos RS. TimeAware test suite prioritization. In: Proceedings of the 2006 International Symposium on Software Testing and Analysis; 2006 Jul 17–20; Portland, MA, USA. p. 1–12.
9. Singh Y, Kaur A, Suri B. Test case prioritization using ant colony optimization. SIGSOFT Softw Eng Notes. 2010;35(4):1–7. doi:10.1145/1811226.1811238.
10. Hla KHS, Choi Y, Park JS. Applying particle swarm optimization to prioritizing test cases for embedded real time software retesting. In: Proceedings of the 2008 IEEE 8th International Conference on Computer and Information Technology Workshops; 2008 Jul 8–11; Sydney, NSW, Australia. p. 527–32. doi:10.1109/CIT.2008.Workshops.104.
11. Garg K, Shekhar S. Optimizing test case prioritization through ranked NSGA-2 for enhanced fault sensitivity analysis. Innov Syst Softw Eng. 2024;20(3):307–28. doi:10.1007/s11334-024-00561-6.
12. Shu T, He Z, Yin X, Ding Z, Zhou M. Model-based diversity-driven learn-to-rank test case prioritization. Expert Syst Appl. 2024;255(6):124768. doi:10.1016/j.eswa.2024.124768.
13. Zeng Q, Zhou Y, Zhou G, Luo Q. Multi-strategies enhanced aquila optimizer for global optimization: comprehensive review and comparative analysis. J Comput Des Eng. 2025;12(5):134–60. doi:10.1093/jcde/qwaf047.
14. AlRassas AM, Al-qaness MAA, Ewees AA, Ren S, Abd Elaziz M, Damaševičius R, et al. Optimized ANFIS model using Aquila optimizer for oil production forecasting. Processes. 2021;9(7):1194. doi:10.3390/pr9071194.
15. Abd Elaziz M, Dahou A, Alsaleh NA, Elsheikh AH, Saba AI, Ahmadein M. Boosting COVID-19 image classification using MobileNetV3 and aquila optimizer algorithm. Entropy. 2021;23(11):1383. doi:10.3390/e23111383.
16. Gupta DK, Dei G, Soni AK, Jha AV, Appasani B, Bizon N, et al. Fractional order PID controller for load frequency control in a deregulated hybrid power system using aquila optimization. Results Eng. 2024;23(1):102442. doi:10.1016/j.rineng.2024.102442.
17. Li Z, Harman M, Hierons RM. Search algorithms for regression test case prioritization. IEEE Trans Softw Eng. 2007;33(4):225–37. doi:10.1109/TSE.2007.38.
18. Shahzad Ahmed F, Majeed A, Ahmed Khan T. Value-based test case prioritization for regression testing using genetic algorithms. Comput Mater Contin. 2023;74(1):2211–38. doi:10.32604/cmc.2023.032664.
19. Tyagi M, Malhotra S. Test case prioritization using multi objective particle swarm optimizer. In: 2014 International Conference on Signal Propagation and Computer Technology (ICSPCT 2014); 2014 Jul 12–13; Ajmer, India. p. 390–5. doi:10.1109/ICSPCT.2014.6884931.
20. Bajaj A, Abraham A, Ratnoo S, Gabralla LA. Test case prioritization, selection, and reduction using improved quantum-behaved particle swarm optimization. Sensors. 2022;22(12):4374. doi:10.3390/s22124374.
21. Ahmad SF, Singh DK, Suman P. Prioritization for regression testing using ant colony optimization based on test factors. In: Intelligent communication, control and devices. Singapore, Singapore: Springer; 2018. p. 1353–60. doi:10.1007/978-981-10-5903-2_142.
22. Zhang W, Qi Y, Zhang X, Wei B, Zhang M, Dou Z. On test case prioritization using ant colony optimization algorithm. In: Proceedings of the 2019 IEEE 21st International Conference on High Performance Computing and Communications, IEEE 17th International Conference on Smart City, IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS); 2019 Aug 10–12; Zhangjiajie, China. p. 2767–773. doi:10.1109/hpcc/smartcity/dss.2019.00388.
23. Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Trans Evol Comput. 1997;1(1):67–82. doi:10.1109/4235.585893.
24. Khatibsyarbini M, Isa MA, Jawawi DNA, Hamed HNA, Mohamed Suffian MD. Test case prioritization using firefly algorithm for software testing. IEEE Access. 2019;7:132360–73. doi:10.1109/ACCESS.2019.2940620.
25. Bajaj A, Sangwan OP. Discrete cuckoo search algorithms for test case prioritization. Appl Soft Comput. 2021;110(3):107584. doi:10.1016/j.asoc.2021.107584.

26. Ahmed Hamza M, Abdelmaboud A, Larabi-Marie-Sainte S, Mesfer Alshahrani H, Al Duhayyim M, Awad Ibrahim H, et al. Modified Harris hawks optimization based test case prioritization for software testing. Comput Mater Contin. 2022;72(1):1951–65. doi:10.32604/cmc.2022.024692.
27. Nayak G, Barisal SK, Ray M. CGWO: an improved grey wolf optimization technique for test case prioritization. Program Comput Softw. 2023;49(8):942–53. doi:10.1134/S0361768823080169.
28. Yang B, Li H, Xing Y, Zeng F, Qian C, Shen Y, et al. Directed search based on improved whale optimization algorithm for test case prioritization. Int J Comput Commun Control. 2023;18(2):5049. doi:10.15837/ijccc.2023.2.5049.
29. Heba H, Younis MI. A hybrid meta-heuristic approach for test case prioritization and optimization. Fusion Pract Appl. 2024;14(2):261–71. doi:10.54216/fpa.140221.
30. Pan R, Bagherzadeh M, Ghaleb TA, Briand L. Test case selection and prioritization using machine learning: a systematic literature review. Empir Softw Eng. 2021;27(2):29. doi:10.1007/s10664-021-10066-6.
31. Yaraghi AS, Bagherzadeh M, Kahani N, Briand LC. Scalable and accurate test case prioritization in continuous integration contexts. IEEE Trans Softw Eng. 2023;49(4):1615–39. doi:10.1109/TSE.2022.3184842.
32. Li Y, Wang Z, Wang J, Chen J, Mou R, Li G. Semantic-aware two-phase test case prioritization for continuous integration. Softw Test Verif Reliab. 2024;34(1):e1864. doi:10.1002/stvr.1864.
33. Fatih Tasgetiren M, Liang YC, Sevkli M, Gencyilmaz G. Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem. Int J Prod Res. 2006;44(22):4737–54. doi:10.1080/00207540600620849.
34. Wang P, Wang Y, Xiang J, Xiao X. Fast image encryption algorithm for logistics-sine-cosine mapping. Sensors. 2022;22(24):9929. doi:10.3390/s22249929.
35. Wang M, Wang JS, Li XD, Zhang M, Hao WK. Harris hawk optimization algorithm based on cauchy distribution inverse cumulative function and tangent flight operator. Appl Intell. 2022;52(10):10999–1026. doi:10.1007/s10489-021-03080-0.
36. Wang C, Shi J, Cai J, Zhang Y, Zheng X, Zhang N. DriverRWH: discovering cancer driver genes by random walk on a gene mutation hypergraph. BMC Bioinform. 2022;23(1):277. doi:10.1186/s12859-022-04788-7.
37. Zhang Y. Backtracking search algorithm with specular reflection learning for global optimization. Knowl Based Syst. 2021;212:106546. doi:10.1016/j.knosys.2020.106546.
38. Hamad RK, Rashid TA. GOOSE algorithm: a powerful optimization tool for real-world engineering challenges and beyond. Evol Syst. 2024;15(4):1249–74. doi:10.1007/s12530-023-09553-6.
39. Zhu X, Zhang J, Jia C, Liu Y, Fu M. A hybrid black-winged kite algorithm with PSO and differential mutation for superior global optimization and engineering applications. Biomimetics. 2025;10(4):236. doi:10.3390/biomimetics10040236.
40. Xue J, Shen B. A novel swarm intelligence optimization approach: sparrow search algorithm. Syst Sci Control Eng. 2020;8(1):22–34. doi:10.1080/21642583.2019.1708830.
41. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H. Harris Hawks optimization: algorithm and applications. Future Gener Comput Syst. 2019;97:849–72. doi:10.1016/j.future.2019.02.028.
42. Mirjalili S, Lewis A. The whale optimization algorithm. Adv Eng Softw. 2016;95(12):51–67. doi:10.1016/j.advengsoft.2016.01.008.
43. Gupta S, Deep K. A novel random walk grey wolf optimizer. Swarm Evol Comput. 2019;44(4):101–12. doi:10.1016/j.swevo.2018.01.001.
44. Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95—International Conference on Neural Networks; 1995 Nov 27–Dec 1; Perth, WA, Australia. p. 1942–8. doi:10.1109/ICNN.1995.488968.
45. Tanabe R, Fukunaga AS. Improving the search performance of SHADE using linear population size reduction. In: Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC); 2014 Jul 6–11; Beijing, China. p. 1658–65. doi:10.1109/CEC.2014.6900380.
46. Abed-alguni BH, Paul D. Island-based Cuckoo Search with elite opposition-based learning and multiple mutation methods for solving optimization problems. Soft Comput. 2022;26(7):3293–312. doi:10.1007/s00500-021-06665-6.
47. Abed-Alguni BH. EvoMapX: an explainable framework for metaheuristic optimization algorithms. Expert Syst Appl. 2026;298(10):129514. doi:10.1016/j.eswa.2025.129514.