An Adaptive Levy Flight Chicken Swarm Optimization with Differential Evolution for Function Optimization Problem

WEN-JUN LIU[1], AZLAN MOHD ZAIN[2], MOHAMAD SHUKOR BIN TALIB[3], SHENG-JUN MA[4]
Faculty of Computing, Universiti Teknologi Malaysia, Skudai, 80310, Malaysia[1,2,3]
Electrical and Electronic Engineering Department, Auckland University of Technology, Auckland, New Zealand[4]

Abstract—**This study proposes an improved swarm algorithm, Adaptive Levy Flight Chicken Swarm Optimization with Differential Evolution (ALCSODE), to overcome the low convergence accuracy and imbalance between exploration and exploitation in the original CSO algorithm. The method incorporates adaptive perturbation based on individual differences and a differential evolution mechanism into the rooster update process. An elitism preservation strategy is also applied to enhance population stability and information sharing. The algorithm is evaluated on 24 benchmark functions, including unimodal, high-dimensional multimodal, and CEC2022 functions. Performance metrics such as search trajectories and convergence curves are used to assess its effectiveness. Experimental results show that ALCSODE achieves a better exploration–exploitation trade-off and shows statistically superior performance over seven classical algorithms, confirming its potential as an effective tool for solving complex optimization problems.**

Keywords*—*Chicken swarm optimization; levy flight; differential evolution algorithm; adaptive adjustment strategy; function optimization

I. Introduction

As the dimensionality and complexity of real-world problems increase, optimization tasks exhibit diverse characteristics, such as the coexistence of discrete and continuous variables, low- and high-dimensional settings, and both convex and non-convex landscapes [1]. These challenges significantly limit the effectiveness of traditional analytical and gradient-based methods. In non-smooth, non-convex, or high-dimensional search spaces, gradients are often difficult to compute, and numerical methods are prone to local optima with slow convergence, which severely hampers solution quality and efficiency.

To overcome these limitations, metaheuristic algorithms have emerged as a powerful alternative. Inspired by natural processes such as evolution and swarm intelligence, these algorithms iteratively refine candidate solutions through stochastic operations, offering derivative-free optimization and strong adaptability [2]. Their core strength lies in the ability to balance global exploration and local exploitation [3]: exploration aims to identify promising regions in the search space, while exploitation focuses on refining existing solutions toward the global optimum. The balance between exploration and exploitation is crucial for achieving good optimization performance. Excessive exploration may reduce convergence speed, while excessive exploitation can lead to premature convergence. Therefore, designing algorithms that can effectively coordinate both aspects remains a central goal in metaheuristic development. Generally, metaheuristic algorithms are categorized into four major types: evolution-based, physics-based, mathematics-based, and swarm intelligence-based methods.

The first category comprises general population-based metaheuristic algorithms, which are inspired by biological evolution and natural selection. These algorithms simulate the competitive survival process within a population to iteratively improve solution quality. Representative methods include Genetic Algorithms (GA) [4], Differential Evolution (DE) [5], and Evolution Strategies (ES) [6]. They typically employ operators such as selection, crossover, and mutation to evolve the population, ensuring both diversity maintenance and convergence toward superior solutions.

The second category consists of physics-inspired optimization algorithms, which guide the search process by mimicking natural physical phenomena such as thermodynamics, gravitational force, and electromagnetic interactions. For instance, Simulated Annealing (SA) [7] is based on the annealing process of materials and helps the algorithm escape local optima. The Gravitational Search Algorithm (GSA) [8] and the Black Hole Algorithm (BH) [9] leverage mechanical or astrophysical models to drive the search process. These methods generally offer strong global search capabilities and are well suited to high-dimensional and complex optimization problems. Moreover, Harmony Search (HS) [10], inspired by musical improvisation, seeks optimal solutions. Dynamic Dimension Adjustment HS (DDA-HS) [11] adapts the search space and employs an absorbing Markov chain to improve exploration–exploitation balance, enhancing fuzzy rule extraction in neural networks for better classification and interpretability.

The third category includes mathematically inspired algorithms, which regulate the search process through rigorous mathematical strategies. The INFO [12] adopts an averaging-based update mechanism and a vector combination strategy to enhance convergence speed and local exploitation performance, while alleviating premature convergence and population stagnation. Similarly, the SHIO [13] incorporates a guidance mechanism based on the average of historical best solutions to balance exploration and exploitation, thus improving global performance and avoiding local optima.

The fourth category, swarm intelligence-based optimization algorithms, is inspired by collective foraging, migration, and cooperative decision-making behaviors observed in biological systems. These algorithms achieve global optimization through simple interactions among individuals and local adaptive search without centralized control, exhibiting strong self-organization, robustness, and parallelism [14].

In recent years, several novel algorithms have emerged in this domain. The Snow Geese Algorithm (SGA) [15] models global guidance and shortest path search by mimicking the V-formation and energy-sharing strategies of migratory geese. The Sled Dog Optimizer (SDO) [16] simulates sled dog team formation and dynamic role-switching to construct a multi-stage cooperative optimization mechanism. The HawkFish Optimization Algorithm (HFOA) [17] incorporates dual-fitness and vision-based dynamic strategies inspired by sex-switching behavior, demonstrating both stability and efficiency in solving multimodal problems. Classical swarm-based algorithms, including Particle Swarm Optimization (PSO) [18], Ant Colony Optimization (ACO) [19], Artificial Bee Colony (ABC) [20], Artificial Fish Swarm Algorithm (AFSA) [21], and Bat Algorithm (BA) [22], along with emerging variants such as Cuckoo Search (CS) [23], Hybrid Harmony Search with Cuckoo Search (HS-CS) [24], Krill Herd Optimization [25], Chaotic Fruit Fly Optimization Algorithm [26], and Social Spider Optimization [27], have found wide applications in fields such as computer science [28], engineering design [29], machine learning [30], image processing [31], and NP-complete problem solving [32].

The Chicken Swarm Optimization (CSO) algorithm [33] is a swarm intelligence technique inspired by the hierarchical structure and foraging behavior of chicken flocks. It assigns individuals as roosters, hens, or chicks, each applying distinct search strategies to achieve cooperative optimization and improve its capability to locate the global optimum [34].

However, the original CSO algorithm faces several limitations, such as a tendency to fall into local optima, slow convergence speed, and high sensitivity to parameter settings. To address these issues, various improved variants have been proposed. To strengthen global exploration, Huang et al. [35] introduce a global best-guided strategy combined with dynamic inertia weight, significantly improving the search capability of the swarm. To mitigate premature convergence, Kong et al. [36] propose a dual-guidance mechanism in which chicks learn from both hens and roosters, thereby enhancing their ability to escape local optima. Shi and Gao [37] integrate CSO with the Artificial Bee Colony (ABC) algorithm to leverage ABC's global search ability for better exploration-exploitation balance. A hybrid strategy is also developed by combining CSO with Particle Swarm Optimization (PSO) to enhance information exchange and population diversity [38]. Zhang et al. [39] further propose L-QCSO by incorporating Levy flight and quantum behavior, aiming to balance local exploitation and global exploration while improving convergence accuracy. Lin et al. [40] enhance the robustness and precision of CSO by modifying the position update rules for roosters and hens. Theoretical advancements have also been made, such as models based on Markov chains and convergence analysis, which provide a reliable foundation for parameter tuning and structural design [41]. In practical applications, CSO is often combined with other optimization or learning models. For example, Afzal et al. [42] present CSO-OEL by integrating CSO with the Optimized Extreme Learning Machine, resulting in improved convergence speed, stability, and overall efficiency.

Despite the significant achievements of various optimization algorithms in solving complex problems, the "No Free Lunch" theorem [43] states that no single algorithm can perform optimally across all problem domains. The performance of an algorithm is generally problem-dependent, making the NFL theorem a fundamental driving force behind the development of problem-specific optimization strategies.

The Chicken Swarm Optimization (CSO) algorithm has attracted attention for its simplicity and effective modeling of social behavior. However, existing studies reveal persistent limitations, including weak role coordination, slow convergence in complex landscapes, and sensitivity to parameter settings—often caused by the imbalance between exploration and exploitation. These issues hinder the algorithm's scalability, adaptability, and solution quality in high-dimensional or dynamic problems. To improve convergence accuracy and address these limitations, this study draws on recent advances in swarm intelligence and proposes that a high-performance optimizer should integrate multi-strategy cooperation and adaptive control mechanisms.

To enhance the scalability, robustness, and convergence behavior of the traditional Chicken Swarm Optimization (CSO), this research proposes a novel algorithm, Adaptive Levy Flight Chicken Swarm Optimization with Differential Evolution (ALCSODE). The main innovation of ALCSODE lies in its strategic integration of multiple adaptive and cooperative mechanisms, each designed to explicitly tackle key structural deficiencies of CSO, rather than simply stacking heuristic components. The contribution is a modular, coordinated optimization framework that strengthens global search capabilities, improves role collaboration, and enhances adaptability to complex landscapes.

Specifically, in the rooster phase, an adaptive differential perturbation mechanism enhances global exploration through individual difference vectors, supported by a fitness-based step-size strategy that dynamically adjusts the search range to balance exploration and exploitation. A non-inertia weight mechanism further improves convergence stability by adapting parameter intensity to the current search stage. In the hen phase, a Levy flight strategy utilizes long-tailed jumps to escape local optima and accelerate convergence. Additionally, an elitism preservation and information-sharing mechanism maintains diversity and guides inferior individuals toward high-quality solutions, improving convergence without compromising population diversity. The scientific contributions of this research are summarized as follows:

- The novelty of this study lies in the design of a new optimizer, ALCSODE, which incorporates an adaptive differential perturbation mechanism, Levy flight strategy, non-inertia weight adjustment, and elitism preservation into the Chicken Swarm Optimization (CSO) framework.
- The proposed ALCSODE algorithm addresses key limitations of the original CSO, including weak role coordination, premature convergence, and sensitivity to parameter settings caused by the imbalance between exploration and exploitation.
- A modular, adaptive optimization framework is developed, with each component designed to enhance

global exploration, convergence stability, and role collaboration in complex search spaces.

- The algorithm's core strategies are formally described and mathematically modeled to ensure transparency and reproducibility.
- ALCSODE is evaluated on 24 benchmark functions, covering unimodal, multimodal, and the CEC2022 test suite, to verify its scalability and effectiveness in diverse optimization scenarios.
- Comparative experiments with seven established meta-heuristic algorithms show that ALCSODE consistently achieves superior performance and robustness in most test cases.

The remainder of this study is organized as follows: Section II reviews the standard CSO and DE algorithms; Section III details the implementation steps of ALCSODE; Section IV presents the simulation experiments and analysis; and Section V summarizes the whole manuscript and outlines future research directions.

II. Related Work and Background Algorithms

Several swarm intelligence algorithms have been proposed in recent years, among which the Chicken Swarm Optimization (CSO) and the Differential Evolution (DE) algorithms are widely used due to their strong optimization capability. This section briefly introduces the basic principles of the standard CSO and DE algorithms, which form the foundation of our proposed ALCSODE method.

A. The Chicken Swarm Optimization

Swarm intelligence algorithms have demonstrated remarkable advantages in solving complex optimization problems. The Chicken Swarm Optimization (CSO) algorithm achieves efficient search performance by simulating the hierarchical behavior of chicken groups [34]. In this algorithm, the population is categorized into three distinct roles: roosters (leaders), hens, and chicks, which correspond to exploration, cooperation, and following behaviors, respectively.Typically, individuals with the best fitness values are assigned as roosters, while those with the worst fitness values are designated as chicks. The remaining individuals act as hens. The entire chicken swarm is further divided into several subgroups, each led by one rooster accompanied by a number of hens and chicks. To enhance the algorithm's adaptability in solving complex multimodal problems, the identities of individuals and the parent–child relationships are periodically redefined.

Let N be the total number of chickens, and R_N, H_N, C_N, and M_N denote the number of roosters, hens, chicks, and mother hens, respectively. The search space is defined as $\mathbb{R}^D$. The position of the i-th individual in the j-th dimension at iteration t is denoted by $x_{i,j}^t$, where $i \in [1, \ldots, N]$, $j \in [1, \ldots, D]$, and t is the current iteration. The position of roosters is updated using a random walk and competition mechanism, formulated as follows [see Eq. (1) and Eq. (2)]:

$$x_{i,j}^{t+1} = x_{i,j}^t(1 + \mathcal{N}(0, \sigma^2)), \tag{1}$$

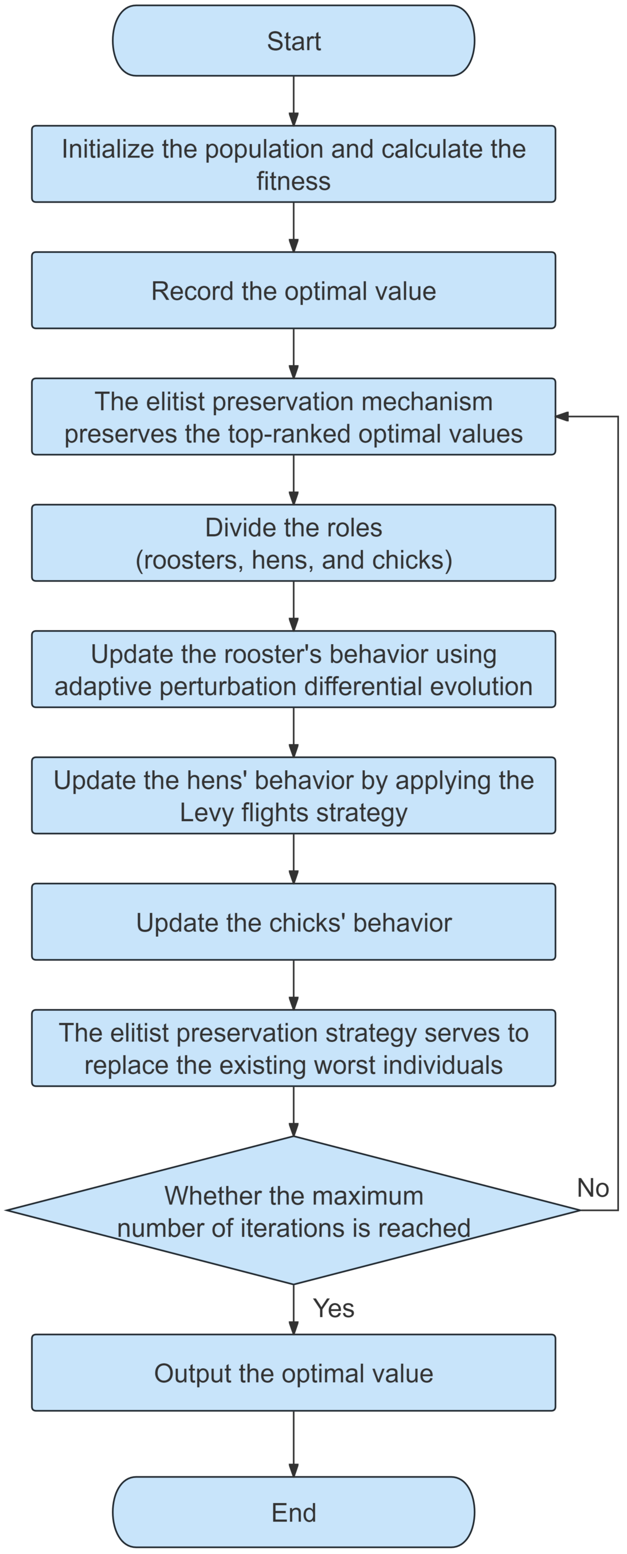

Start
Initialize the population and calculate the fitness
Record the optimal value
The elitist preservation mechanism preserves the top-ranked optimal values
Divide the roles (roosters, hens, and chicks)
Update the rooster's behavior using adaptive perturbation differential evolution
Update the hens' behavior by applying the Levy flights strategy
Update the chicks' behavior
The elitist preservation strategy serves to replace the existing worst individuals
Whether the maximum number of iterations is reached
No
Yes
Output the optimal value
End

Fig. 1. The flowchart of ALCSODE algorithm.

$$\sigma^2 = \begin{cases} \exp\left(\frac{f_k - f_i}{|f_i| + \epsilon}\right) & \text{if } f_k > f_i, \\ 1 & \text{otherwise,} \end{cases} \tag{2}$$

where, f_i represents the fitness of the current rooster,f_k is the fitness of a randomly selected rooster, andϵ is a small constant used to avoid division by zero.

For hens, their position update combines leadership from the roosters and social learning. The corresponding update equation is defined as follows [see Eq. (3)]:

$$x_{i,j}^{t+1} = x_{i,j}^t + c_1 r_1 (x_{r,j}^t - x_{i,j}^t) + c_2 r_2 (x_{m,j}^t - x_{i,j}^t), \quad (3)$$

where, $x_{r,j}^t$ denotes the position of the leading rooster, $x_{m,j}^t$ represents the position of another randomly selected hen, c_1 and c_2 are learning factors (typically defined as $c_1 = \exp(f_i/f_r)$, $c_2 = \exp(f_m - f_i)$), and $r_1, r_2 \in [0, 1]$ are random numbers.

As for the chicks, they strictly follow the movement of the mother hens. The update equation is as follows [see Eq. (4)]:

$$x_{i,j}^{t+1} = x_{i,j}^t + F(x_{h,j}^t - x_{i,j}^t). \quad (4)$$

where, $x_{h,j}^t$ denotes the position of the hen, and $F \in (0, 2]$ is the following step size factor.

B. The Differential Evolution Algorithm

Differential Evolution (DE [44]) is a population-based stochastic optimization algorithm proposed by Storn and Price in 1997. It is an important branch of evolutionary computation. The core idea of DE is to simulate the mutation, crossover, and selection operations in biological evolution to efficiently search for the global optimal solution in continuous space. The DE algorithm begins by initializing a fixed-size population, with each individual representing a candidate solution in the solution space (referred to as a target vector). The optimization process then iteratively refines the solution, gradually approaching the optimal one. During the mutation step, the algorithm generates mutation vectors using differential operations. Common strategies include generating mutation vectors by linearly combining three random individuals (DE/rand/1) and introducing the global best solution to guide mutation (DE/rand-to-best/1), which helps overcome the limitations of traditional mutation methods in genetic algorithms (GA [4]). In the crossover phase, the mutation vectors and target vectors are probabilistically mixed to create trial vectors, enhancing population diversity. Finally, a greedy selection strategy is used to retain individuals with better fitness for the next generation, continuously improving the solution quality. The DE algorithm is widely applied in single-objective and multi-objective optimization, combinatorial optimization, and engineering problems due to its simple structure, few parameters, and high efficiency.

III. The Proposed ALCSODE Method

In this section, we present our Adaptive Levy Flight Chicken Swarm Optimization with Differential Evolution algorithm, termed ALCSODE, whose detailed pseudocode is provided in Algorithm 1 and overall workflow is illustrated in Fig. 1. For clarity and focus, our method follows the standard CSO framework for initialization and role assignment.

The algorithm is divided into four core components, highlighting two innovative improvements: 1) The adaptive perturbation-based differential evolution strategy for rooster behavior updating. 2) The elitism preservation mechanism to improve global robustness.

A. Adaptive Perturbation-Based Differential Evolution for Rooster Behavior Updating

In Chicken Swarm Optimization (CSO) algorithms, rooster behavior plays a pivotal role in guiding population exploration and maintaining swarm diversity, as documented in foundational studies [45]. However, traditional updating mechanisms exhibit significant limitations: 1) single-strategy search susceptibility to local optima causing premature convergence, 2) insufficient spatial perturbation, and 3) fixed parameters lacking environmental adaptability [46]. To address these issues, our study proposes a method integrating individual adaptability perturbation with the differential evolution algorithm [5]. We use the differential information among individuals to guide the search to high-quality regions. The adaptive perturbation dynamically adjusts the search step size based on individual fitness. This synergy balances global exploration and local exploitation, avoiding search imbalance from single-strategy approaches. Additionally, parameters can adaptively adjust their intensity, enabling effective regulation of search efforts at different stages. The detailed method is introduced in the following section.

1) Calculation of adaptive differential mutation vector: To enhance global exploration in early stages and intensify local exploitation in later phases, this study introduces an adaptive update strategy for the scaling factor F and crossover probability CR in Differential Evolution. Let t denote the current iteration and T the maximum iterations, with initial values F_0 and CR_0. The adaptive update equations are defined as Eq. (5) and Eq. (6):

$$F_t = F_0 \left(1 - \frac{t}{T}\right)^{\theta} + F_{min}, \quad (5)$$

$$CR_t = CR_0 \left(1 - \frac{t}{T}\right)^{\theta} + CR_{min}, \quad (6)$$

where, CR_{min} and F_{min} are both set to 0.1, and $\theta > 0$ controls the decay rate. This adaptive mechanism ensures larger step sizes and higher mutation probabilities during initial phases, facilitating escape from local optima, while progressively decreasing parameters promote stable convergence as iterations advance.

Using differential evolution, our method leverage information from other population individuals to guide global search directions. Three distinct individuals are randomly selected from the population, denoted as $\mathbf{x}_{r_1}$, $\mathbf{x}_{r_2}$, $\mathbf{x}_{r_3}$. The differential evolution mutation vector is constructed as follows [see Eq. (7)]:

$$\mathbf{d}_i^{t+1} = \mathbf{x}_{r_1}^t + F_t \left(\mathbf{x}_{r_2}^t - \mathbf{x}_{r_3}^t\right). \quad (7)$$

where, t denotes the iteration, F_t represents the adaptive scaling factor. $\mathbf{d}_i^t$ provides a new search direction based on population differential information. This step effectively utilizes inter-population differential information to introduce diversity in global search, facilitating exploration of potential optimal regions.

2) Calculation of individual-adaptive perturbation factor: Individual-specific perturbations are applied to candidate solutions (representing rooster behaviors in the swarm) to enhance search capabilities of inferior individuals. Let f_i denote the fitness of the i-th individual and f_b the best fitness in the population. The perturbation standard deviation σ_i is defined as Eq. (8):

$$\sigma_i^2 = \begin{cases} 1, & f_i \leq f_b \\ \exp\left(\frac{f_b - f_i}{|f_i| + \varepsilon}\right), & f_i > f_b \end{cases} \tag{8}$$

where, ε represents a minimal positive constant to prevent division by zero. The perturbation step size δ_i is then sampled from a normal distribution with zero mean and variance σ_i^2, as in Eq. (9):

$$\delta_i \sim \mathcal{N}\big(0, \sigma_i^2\big). \tag{9}$$

Fitness-adaptive perturbation adjustment enables inferior individuals to receive larger perturbations, facilitating escape from local optima, while superior individuals maintain stability, ensuring convergence quality.

3) Updating the roosters behavior: Integrating global differential search information with current rooster states, the next generation is generated as follows. Let $\mathbf{x}_i^t$ denote the current individual position. With a trade-off parameter β, the next generation vector is defined as Eq. (10):

$$\mathbf{x}_i^{t+1} = \alpha\, \mathbf{d}_i^{t+1} + \delta_i\, \mathbf{x}_i^t, \tag{10}$$

where, $\alpha \in (0, \infty)$ controls the relative weight between global exploration and local exploitation. $\mathbf{d}_i^{t+1}$ represents the differential evolution mutation vector computed by Eq. (7), while δ_i denotes the adaptive perturbation factor calculated from Eq. (9).

To prevent excessive localization of new solutions, a component-wise crossover strategy is employed for dimension-by-dimension position updating. For each dimension j, the next generation rule is defined as Eq. (11):

$$x_{i,j}^{\prime t+1} = \begin{cases} x_{i,j}^{t+1}, & \text{if } rand(0,1) \leq CR_t \text{ or } j = j_{rand} \\ x_{i,j}^t, & otherwise \end{cases} \tag{11}$$

where, CR_t denotes the crossover probability calculated by the Eq. (6), and j_{rand} ensures at least one dimension is updated.

By integrating the differential evolution mutation vector with individual-specific perturbations and updating the rooster vector through component-wise crossover, this strategy maintains population diversity while effectively balancing global exploration and local exploitation, thereby enhancing convergence speed and stability.

B. Levy Flight-Based Hen Behavior Optimization

To enable hen individuals to adaptively adjust search step sizes based on fitness information during updating, dynamic scaling factors are first constructed from fitness differences between the current individual and two randomly selected individuals. The scaling factors are defined as Eq. (12) and Eq. (13):

$$\lambda_1 = \exp\left(\frac{f_i - f_l}{|f_i| + \varepsilon}\right), \tag{12}$$

$$\lambda_2 = \exp\left(f_k - f_i\right), \tag{13}$$

where, f_i denotes the fitness of the current hen individual; f_l and f_k represent the fitness values of randomly selected individuals; ε is a small positive constant preventing division by zero; λ_1 and λ_2 reflect the relative fitness advantage or disadvantage of the current individual, providing dynamic adjustment factors for subsequent position updating.

To enhance global exploration capability, the Levy flight function is employed. Given the Levy parameter β, the scale parameter σ_u is first calculated as Eq. (14):

$$\sigma_u = \left[\frac{\Gamma(1+\beta)\,\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right)\,\beta\, 2^{\frac{\beta-1}{2}}}\right]^{\frac{1}{\beta}}, \tag{14}$$

where, $\Gamma(\cdot)$ denotes the gamma function. The Levy flight step size is calculated as Eq. (15):

$$L = \sigma_u \left(u^2 + v^2\right)^{\frac{1}{2\beta}}, \tag{15}$$

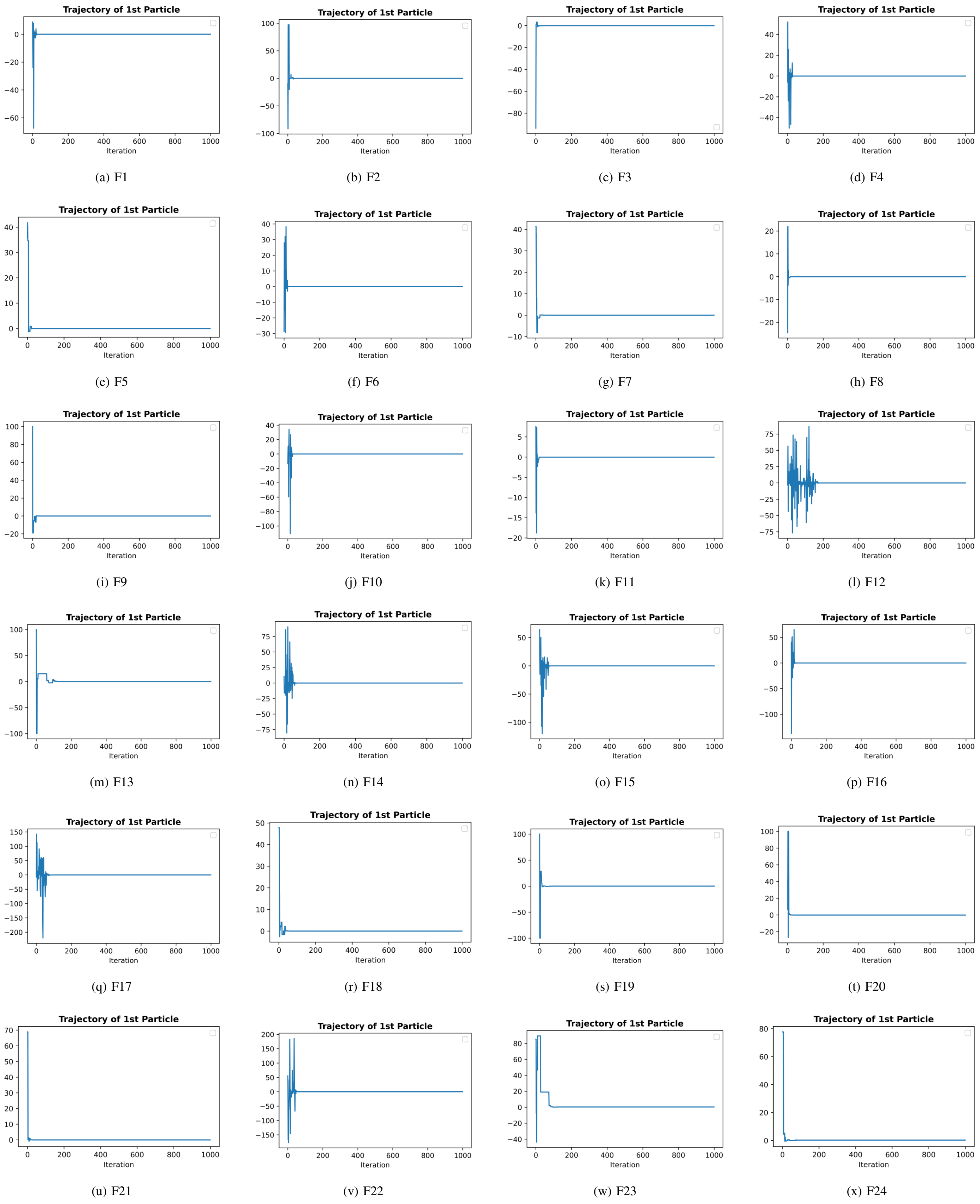
where, u and v are independent random variables following the standard normal distribution; β denotes the Levy flight parameter; L represents the random step size generated from the Levy distribution, exhibiting heavy-tailed characteristics that facilitate large-step.

Integrating the dynamic scaling factors and Levy flight step size, the position updating equation for hen individuals is defined as Eq. (16):

$$\mathbf{x}_{i,j}^{t+1} = \mathbf{x}_{i,j}^t + \lambda_1\, \rho_1\, L\, \left(\mathbf{x}_{l,j}^t - \mathbf{x}_{i,j}^t\right) + \lambda_2\, \rho_2\, L\, \left(\mathbf{x}_{k,j}^t - \mathbf{x}_{i,j}^t\right). \tag{16}$$

where, j denotes the dimension, t the iteration count, and ρ_1 and ρ_1 are random numbers uniformly sampled from the interval [0,1].

By integrating current hen and random individuals' position info with Levy-based step sizes, global exploration and local exploitation are effectively combined. A dynamic scaling factor gives personalized update magnitudes, and the Levy step's long-tail property boosts search jumps, aiding escape from local optima and accelerating global convergence.



(a) F1 (b) F2 (c) F3 (d) F4

(e) F5 (f) F6 (g) F7 (h) F8

(i) F9 (j) F10 (k) F11 (l) F12

(m) F13 (n) F14 (o) F15 (p) F16

(q) F17 (r) F18 (s) F19 (t) F20

(u) F21 (v) F22 (w) F23 (x) F24

Fig. 2. Trajectory of the first particle in ALCSODE during optimization across 24 benchmark functions, illustrating its behavior at different stages.

C. Chick Behavior Updating

In this method, the standard CSO algorithm [33] for updating chick behavior is adopted. For each chick (denoted as individual i), a randomly selected leader individual (indexed as r) serves as the reference during position updating. Randomness and update magnitude are regulated through the following Eq. (17):

$$\mathbf{x}_{i,j}^{t+1} = \mathbf{x}_{i,j}^{t} + \varphi\left(\mathbf{p}_{r,j} - \mathbf{x}_{i,j}^{t}\right). \tag{17}$$

where, $\mathbf{p}_{r,j}$ denotes the historical best position vector of individual r, and φ represents a random scaling factor regulating the update step size, ensuring appropriate stochastic perturbation during updating.

TABLE I. UNIMODAL AND MULTIMODAL TEST FUNCTIONS OF 12 STANDARD BENCHMARKS

No	Functions	Interval	Trait	$f(x)_{min}$
F1	Alpine	[-100,100]	Mulitimodal	0
F2	Elliptic	[-100,100]	Unimodal	0
F3	Schwefel2.22	[-100,100]	Unimodal	0
F4	Sphere	[-100,100]	Unimodal	0
F5	Sum Squares	[-100,100]	Unimodal	0
F6	Ackley	[-100,100]	Mulitimodal	0
F7	Bent Cigar	[-100,100]	Unimodal	0
F8	Discus	[-100,100]	Unimodal	0
F9	Schaffer	[-100,100]	Mulitimodal	0
F10	Quartic	[-100,100]	Unimodal	0
F11	Griewank	[-100,100]	Mulitimodal	0
F12	Rastrigin	[-100,100]	Mulitimodal	0

TABLE II. TEST FUNCTIONS OF CEC2022

No	Functions	Interval	Trait	$f(x)_{min}$
F13	Zakharov	[-100,100]	Unimodal	300
F14	Rosenbrock	[-100,100]	Mulitimodal	400
F15	Schaffer's f6	[-100,100]	Mulitimodal	600
F16	Non-Continuous Rastrigin	[-100,100]	Mulitimodal	800
F17	Levy	[-100,100]	Mulitimodal	900
F18	Hybrid Function 1(N=3)	[-100,100]	Hybrid	1800
F19	Hybrid Function 2(N=6)	[-100,100]	Hybrid	2000
F20	Hybrid Function 3(N=5)	[-100,100]	Hybrid	2200
F21	Composition Function 1(N=5)	[-100,100]	Composition	2300
F22	Composition Function 2(N=4)	[-100,100]	Composition	2400
F23	Composition Function 3(N=5)	[-100,100]	Composition	2600
F24	Composition Function 4(N=6)	[-100,100]	Composition	2700

D. The Elitism Preservation Mechanism

In most CSO and its variant methods, the stochastic nature of individual behavior updating mechanisms (e.g., differentiated strategies for roosters, hens, and chicks) may lead to the loss of high-quality solutions, forcing the algorithm to restart local search and significantly reducing convergence efficiency. To address this, an elitism preservation strategy is introduced to balance exploration-exploitation trade-offs: after each iteration t, individuals with the best fitness are selected to form an elite pool, replacing low-fitness individuals in iteration $t+1$. This mechanism enhances population stability through elite guidance while enabling non-elite individuals to accelerate convergence toward elite solution regions via information sharing, thereby optimizing global convergence performance.

TABLE III. PARAMETER SETTINGS

Algorithm	Parameters
PSO	$c1$ = $c2$ = 2.0, w = 0.8
SHIO	$N_{best} = 3$, $N_{worst} = 1$, a = 1.5
INFO	$cond1_t = 0.05$, $cond2_t = 0.05$
CSO	$R_N = 0.2 \times N$, $H_N = 0.6 \times N$, $C_N = N - R_N - H_N$, $M_N = 0.1 \times H_N$, $G = 10$, $F \in$ rand$(0.4, 1)$
L-QCSO	$R_N = 0.2 \times N$, $H_N = 0.6 \times N$, $C_N = N - R_N - H_N$, $M_N = 0.1 \times H_N$, $G = 10$, $F \in$ rand$(0.4, 1)$
ICSO	$R_N = 0.2 \times N$, $H_N = 0.6 \times N$, $C_N = N - R_N - H_N$, $M_N = 0.1 \times H_N$, $G = 10$, $F \in$ rand$(0.4, 1)$, C = 0.4
ECSO	$R_N = 0.2 \times N$, $H_N = 0.6 \times N$, $C_N = N - R_N - H_N$, $M_N = 0.1 \times H_N$, $G = 10$, $F \in$ rand$(0.4, 1)$, k = 1000
ALCSODE	$R_N = 0.2 \times N$, $H_N = 0.6 \times N$, $C_N = N - R_N - H_N$, $M_N = 0.1 \times H_N$, $G = 10$, $F \in$ rand$(0.4, 1)$, F_0 =0.8 ,CR_0=0.9, η=0.1

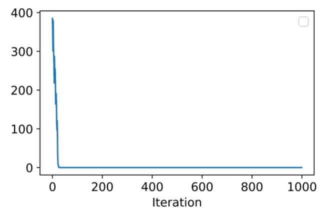
Algorithm 1 Adaptive Levy Flight Chicken Swarm Optimization with Differential Evolution (ALCSODE)

```
Require: $f(\mathbf{x})$, $N$, $D$, $T$, $F_0, F_{\min}$, $CR_0, CR_{\min}$, $\theta$, $\eta$, $[LB, UB]$, $\beta$, $\varepsilon$
Ensure: Best solution $\mathbf{x}^*$ and fitness $f(\mathbf{x}^*)$
 1: Init: sample $\{\mathbf{x}_i\}_{i=1}^{N} \sim U[LB, UB]$, eval. $f_i$, set $\mathbf{x}^*$
 2: for $t = 1 \ldots T$ do
 3:     Elite: sort $\{\mathbf{x}_i\}$ by $f_i$, extract top $\eta N$ into $E$
 4:     Partition remaining into Roosters $R$, Hens $H$, Chicks $C$
 5:     Compute $F_t = F_0(1 - \frac{t}{T})^{\theta} + F_{\min}$, $CR_t = CR_0(1 - \frac{t}{T})^{\theta} + CR_{\min}$
 6:     for all $i \in R$ do    ▷ Rooster (Adaptive DE + Perturb)
 7:         pick distinct $r_1, r_2, r_3$, set $\mathbf{d} = \mathbf{x}_{r_1} + F_t(\mathbf{x}_{r_2} - \mathbf{x}_{r_3})$
 8:         $\sigma_i^2 = 1$ if $f_i \leq f_{\min}$ else $\exp((f_{\min} - f_i)/(|f_i| + \varepsilon))$
 9:         sample $\delta_i \sim \mathcal{N}(0, \sigma_i^2)$, form $\mathbf{y} = \alpha\mathbf{d} + \delta_i\mathbf{x}_i$
10:         crossover $[\mathbf{y}, \mathbf{x}_i] \xrightarrow{CR_t} \mathbf{x}_i'$, replace if $f(\mathbf{x}_i') < f_i$
11:     end for
12:     for all $i \in H$ do    ▷ Hen (Levy Flight)
13:         pick $l, k \neq i$, compute $\lambda_1, \lambda_2$, compute $L$
14:         update $\mathbf{x}_i'$ , replace if improved
15:     end for
16:     for all $i \in C$ do    ▷ Chick (Follow Leader)
17:         pick leader $r$, sample $\varphi \sim U(0, 1)$, set $\mathbf{x}_i' = \mathbf{x}_i + \varphi(\mathbf{p}_r - \mathbf{x}_i)$, replace if $f(\mathbf{x}_i') < f_i$
18:     end for
19:     Elite Replacement: inject $E$ into worst slots
20:     Update global best $\mathbf{x}^*$
21: end for
        return $\mathbf{x}^*$, $f(\mathbf{x}^*)$
```

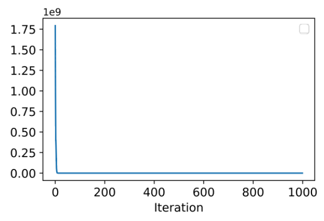
E. Flowchart of the ALCSODE Algorithm

Following the introduction of the four sections above, the detailed pseudocode of the ALCSODE algorithm is presented in Algorithm 1, followed by the overall flowchart illustrated in Fig. 1, the specific steps are then detailed as follows:

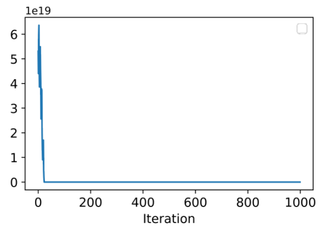
Step 1: Problem modeling and algorithm parameters initialization. Define the objective function $f(x)$, population size N, problem dimension j, initial scaling factor F_0, initial crossover



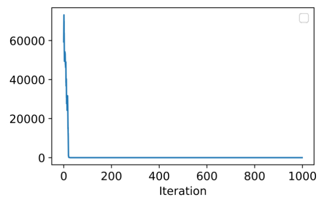
(a) F1



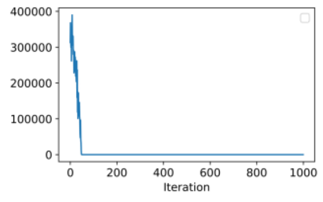
(b) F2



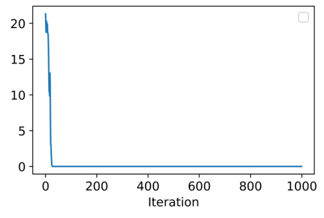
(c) F3



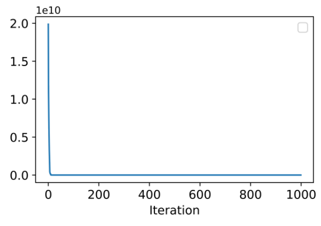
(d) F4



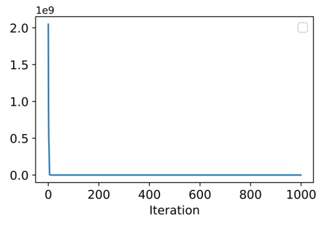
(e) F5



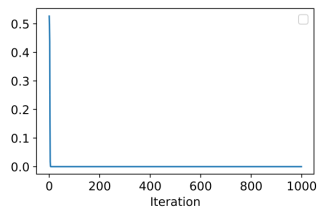
(f) F6



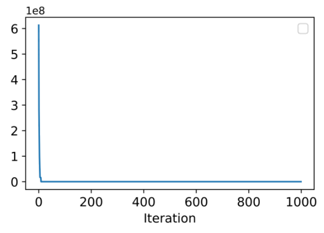
(g) F7



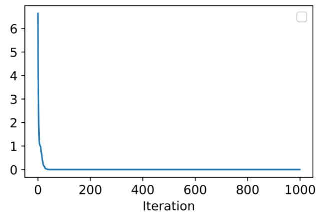
(h) F8



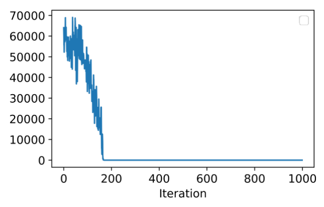
(i) F9



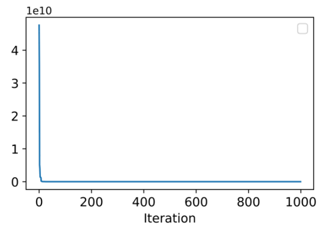
(j) F10



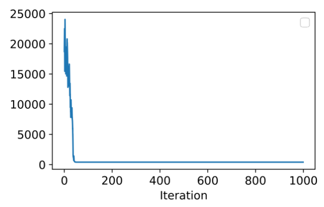
(k) F11



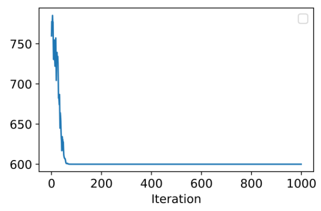
(l) F12



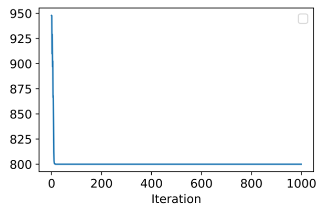
(m) F13



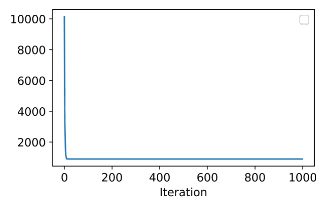
(n) F14



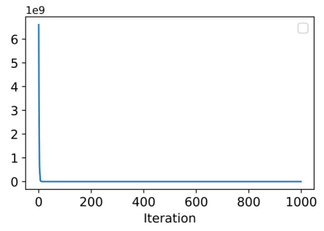
(o) F15



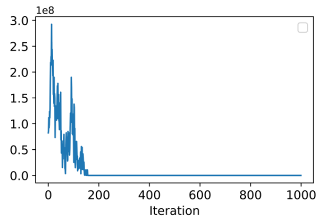
(p) F16



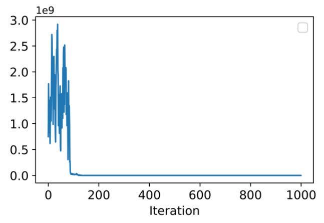
(q) F17



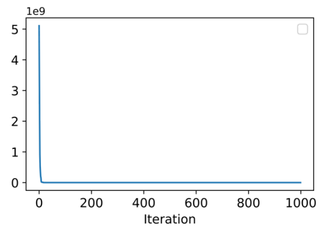
(r) F18



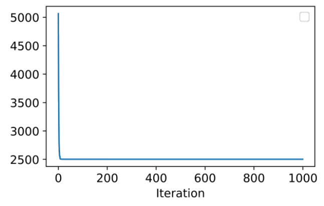
(s) F19



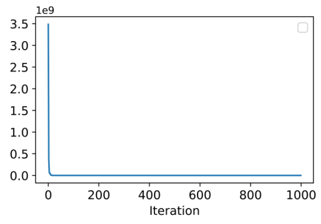
(t) F20



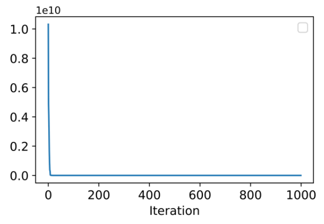
(u) F21



(v) F22



(w) F23



(x) F24

Fig. 3. Average fitness curve of all particles in ALCSODE during optimization, showing the population's overall convergence trend.

TABLE IV. COMPARISON OF EXPERIMENTAL RESULTS FROM F1 TO F8

Function	Algorithm	Mean(dim=10)	Std dev(dim=10)	Best(dim=10)	Mean(dim=50)	Std dev(dim=50)	Best(dim=50)
F1	PSO	1.2171e1	2.8476e1	4.2222	7.4153e2	1.2193e2	7.1206e2
	CSO	8.0693	6.5431	6.2974	1.3867e2	4.0942e1	1.2089e2
	L-QCSO	6.5275	4.3767	6.0746	1.6510e2	2.7145e1	1.5939e2
	ICSO	1.3988	4.4637	5.6363e−1	6.2475e1	5.0660e1	4.1254e1
	ECSO	1.6364e1	5.1912	1.5696e1	2.4910e2	3.6614e1	2.4635e2
	SHIO	3.0936	2.6862	2.2920	1.1520e2	8.4097e1	8.3814e1
	INFO	2.7377	9.2771	3.5371e−4	9.0033	4.0948e1	6.7776e−27
	ALCSODE	**2.4438e-2**	**3.6105e-1**	**5.5180e-174**	**8.6178e-2**	**1.0753**	**4.0827e-165**
F2	PSO	2.0592e7	7.0560e7	1.3823e7	2.3792e9	6.6404e8	2.2575e9
	CSO	6.1484e5	7.0106e6	1.3942e5	6.9277e7	8.0645e7	6.3069e7
	L-QCSO	3.0958e5	4.6929e6	2.5306e3	9.9319e6	6.9314e7	3.4195e6
	ICSO	7.3493e4	1.1988e6	4.5347e3	1.6017e7	6.6362e7	1.0614e6
	ECSO	2.4478e6	2.1051e7	8.5139e5	1.7463e8	1.8639e8	1.6221e8
	SHIO	1.4512e5	3.1824e6	3.0066e−129	8.7830e6	9.7354e7	3.8254e−36
	INFO	2.5434e5	2.4428e6	1.0305e−49	1.0270e7	9.9802e7	7.0123e−48
	ALCSODE	**4.3809e4**	**9.9335e5**	**0.0000**	**1.7922e6**	**4.0895e7**	**0.0000**
F3	PSO	5.0466e10	4.7825e11	4.3917e2	7.8276e77	1.0064e79	2.4860e8
	CSO	2.0637e6	3.2371e7	3.6657e1	1.2770e62	4.0361e63	3.6083e2
	L-QCSO	2.9970e8	6.7707e9	5.1316	4.4075e56	1.3931e58	3.2877e2
	ICSO	2.5172e6	7.9539e7	2.1552e−1	1.6412e62	5.1874e63	5.4136e16
	ECSO	1.0843e6	1.7605e7	7.3739e1	8.0613e60	1.8008e62	1.1096e36
	SHIO	8.1182e8	1.9947e10	2.2337e−82	1.0701e53	1.9500e54	**1.1586e-27**
	INFO	2.7265e7	8.4745e8	8.1567e−27	2.1412e42	6.7677e43	7.6097e−26
	ALCSODE	**1.2155e5**	**3.6764e6**	**3.0275e-246**	**9.8591**	**7.0672e-1**	9.8327
F4	PSO	5.1901e2	3.3705e3	3.6445e1	1.4171e5	1.0273e4	1.3982e5
	CSO	1.4060e2	2.8364e2	1.0768e2	4.0931e3	2.0681e3	3.9001e3
	L-QCSO	2.1161e1	1.3107e2	9.2691	4.4675e3	1.1940e3	4.3806e3
	ICSO	1.4684e1	2.3197e2	4.5753e−3	7.4711e2	3.1091e3	3.0651e−1
	ECSO	2.2854e2	3.7754e2	1.8268e2	9.4319e3	3.4674e3	9.2133e3
	SHIO	7.4621	9.6427e1	5.0667e−133	3.7651e2	3.0527e3	2.3577e−39
	INFO	1.8716e1	2.0239e2	1.3674e−54	1.5604e2	1.5520e3	7.8158e−53
	ALCSODE	**1.5102e-2**	**1.9428e-1**	**5.0988e-321**	**4.8442e-1**	**1.3814e1**	**0.0000**
F5	PSO	2.3160e4	1.3810e4	2.1276e4	2.7388e6	3.2382e5	2.6813e6
	CSO	1.0870e3	1.6029e3	9.5040e2	1.1909e5	5.4754e4	1.1518e5
	L-QCSO	2.8691e2	7.0381e2	2.3292e2	8.2984e4	**3.0516e4**	8.0998e4
	ICSO	1.2231e2	1.6128e3	4.1939e−2	2.1288e4	7.5819e4	1.3021e1
	ECSO	3.1497e3	1.3458e3	3.0072e3	2.7433e5	1.2475e5	2.6596e5
	SHIO	1.2551e1	1.9611e2	9.8009e−130	6.0821e3	4.7958e4	1.5286e−38
	INFO	**9.1802**	**8.0626e1**	7.0286e−54	4.3272e3	3.9028e4	4.4386e−51
	ALCSODE	5.0307e1	1.0947e3	**0.0000**	**1.5263e3**	3.4385e4	**0.0000**
F6	PSO	2.0001e1	**1.7293e-2**	2.0000e1	2.0081e1	3.6345e−2	2.0069e1
	CSO	9.9341	2.8200	8.6036	2.0000e1	8.9650e−1	1.7869e1
	L-QCSO	3.1154	1.8389	2.7988	1.6499e1	7.7940e−1	1.6129e1
	ICSO	7.9773	1.6355	7.7176	2.0072e1	4.3520e−1	1.9810e1
	ECSO	1.4689e1	6.2106e−1	1.4612e1	1.9841e1	**1.8158e-1**	1.9807e1
	SHIO	3.8085	1.7584	3.5751	7.6821	4.9520	5.5864
	INFO	1.0991	3.9546	**4.4409e-16**	6.7263	9.3060	3.9968e−15
	ALCSODE	**3.2380e-2**	3.4513e−1	**4.4409e-16**	**5.2054e-2**	5.5285e−1	**4.4409e-16**
F7	PSO	2.4477e8	1.8254e9	5.0589e4	1.2063e11	1.2824e10	1.1825e11
	CSO	2.4465e8	2.1680e8	2.2719e8	4.6980e9	2.2347e9	4.5431e9
	L-QCSO	4.8091e7	1.8254e8	3.8036e7	3.0407e9	**1.2999e9**	2.9397e9
	ICSO	1.3758e7	1.9220e8	4.9460e3	5.9858e8	3.0465e9	3.1540e5
	ECSO	5.0713e8	2.7607e8	4.8645e8	1.1830e10	3.0300e9	1.1643e10
	SHIO	1.0690e7	1.5594e8	1.9939e−127	2.6346e8	2.1162e9	5.6676e−35
	INFO	1.4142e7	1.7633e8	3.3738e−48	2.1443e8	3.2860e9	1.3765e−46
	ALCSODE	**6.1252e6**	**1.1558e8**	**0.0000**	**6.7484e7**	1.3447e9	**0.0000**
F8	PSO	1.3858e5	1.5930e6	4.9400e4	2.1096e5	1.4863e5	1.9474e5
	CSO	1.3100e4	3.5874e5	5.8645e2	1.1812e4	1.3979e4	1.0421e4
	L-QCSO	7.7057e2	9.0170e3	9.7570e1	3.8709e3	1.6819e4	2.0931e3
	ICSO	4.2348e3	2.7635e3	3.2313e3	2.3510e4	3.5057e4	1.8347e4
	ECSO	1.9217e3	1.7315e3	1.6479e3	1.0926e4	8.4838e3	1.0063e4
	SHIO	**7.5133e1**	**1.0423e3**	2.0267e−126	7.5092e2	**5.7429e3**	3.9283e−39
	INFO	5.9024e3	1.5928e5	7.3494e−50	5.8405e3	2.4436e4	9.0716e−50
	ALCSODE	1.1903e2	2.2259e3	**0.0000**	**4.4433e2**	9.1330e3	**0.0000**

probability CR_0, elite ratio η, maximum iteration times T , and search domain range $[LB, UB]$.

TABLE V. COMPARISON OF EXPERIMENTAL RESULTS FROM F9 TO F16

Function	Algorithm	Mean(dim=10)	Std dev(dim=10)	Best(dim=10)	Mean(dim=50)	Std dev(dim=50)	Best(dim=50)
F9	PSO	2.1915e-3	2.3394e-2	**0.0000**	4.3981e-3	4.0932e-2	**0.0000**
	CSO	1.2950e-3	1.0902e-2	**0.0000**	1.5582e-3	1.5368e-2	1.3605e-11
	L-QCSO	2.8970e-4	4.1479e-3	**0.0000**	7.0125e-4	1.1484e-2	**0.0000**
	ICSO	1.1525e-3	1.3775e-2	9.9178e-10	5.6907e-4	1.0126e-2	2.4241e-11
	ECSO	2.0857e-3	1.8151e-2	2.5352e-6	1.5202e-3	8.8111e-3	7.5935e-8
	SHIO	6.7496e-4	2.6408e-3	**0.0000**	9.3982e-4	7.2033e-3	**0.0000**
	INFO	2.0511e-4	3.5885e-3	**0.0000**	1.2323e-3	1.1032e-2	**0.0000**
	ALCSODE	**3.8614e-5**	**1.2176e-3**	**0.0000**	**2.8502e-7**	**8.4581e-6**	**0.0000**
F10	PSO	9.8715e6	8.7741e7	1.0265e4	1.5410e10	2.7870e9	1.4932e10
	CSO	8.5630e4	1.5200e6	8.9011e2	1.4151e7	1.1050e8	8.4274e6
	L-QCSO	2.3330e4	**3.7674e5**	1.9332e-1	9.5990e6	6.3321e7	5.5855e6
	ICSO	6.1236e4	1.8500e6	1.5713e-3	2.0692e7	2.2624e8	2.4192e-1
	ECSO	2.6265e5	1.6985e6	1.5588e5	4.9450e7	2.0985e8	3.4797e7
	SHIO	5.3305e4	9.7508e5	**2.1546e-5**	1.1907e7	1.3905e8	2.6926e-3
	INFO	2.5538e4	4.3033e5	3.8682e-4	2.4971e6	3.4058e7	**1.1924e-3**
	ALCSODE	**2.1052e4**	5.5612e5	1.0986e-4	**1.3696e2**	**3.1244e2**	1.2381e2
F11	PSO	2.2655	6.9911e-1	2.1518	2.7139e1	3.5048	2.6552e1
	CSO	4.0950e-1	1.2417e-1	3.9084e-1	2.0201	6.0190e-1	1.9773
	L-QCSO	6.4440e-1	6.0465e-2	6.3673e-1	2.2367	2.4165e-1	2.2226
	ICSO	1.0660e-1	1.5407e-1	8.4411e-2	4.2695e-1	1.0284	2.2509e-2
	ECSO	1.1209	8.1574e-2	1.1151	2.3045	5.6463e-1	2.2648
	SHIO	1.6011e-1	1.2006e-1	1.1470e-1	2.0404e-1	1.0539	**0.0000**
	INFO	1.5093e-1	2.8325e-1	2.4612e-2	2.6216e-1	5.6060e-1	**0.0000**
	ALCSODE	**1.3474e-3**	**2.7760e-2**	**0.0000**	**1.5361e-3**	**2.9620e-2**	**0.0000**
F12	PSO	4.0863e2	3.3017e3	2.3099e1	1.3827e5	1.2042e4	1.3592e5
	CSO	4.8146e2	2.9826e2	4.5710e2	3.7476e3	2.0547e3	3.5948e3
	L-QCSO	1.4756e2	1.5853e2	1.2427e2	7.2951e3	1.0877e3	7.2002e3
	ICSO	3.0308e2	1.7108e2	2.8938e2	4.8178e3	2.5093e3	4.4025e3
	ECSO	5.0184e2	5.2995e2	4.6734e2	7.1712e3	3.7697e3	6.9471e3
	SHIO	5.3312e1	1.9080e2	3.5870e1	1.1539e3	2.1315e3	8.4338e2
	INFO	2.7375e1	2.0187e2	**0.0000**	2.2399e2	1.1400e3	**0.0000**
	ALCSODE	**6.2019**	**1.2584e2**	**0.0000**	**1.3460e2**	**2.4440e1**	1.3308e2
F13	PSO	1.4231e4	4.4013e3	1.3391e4	1.5764e5	2.2018e4	1.5286e5
	CSO	8.5161e2	1.0744e3	7.7142e2	2.6140e7	7.9264e8	7.5380e3
	L-QCSO	2.9007e3	2.8382e3	2.6788e3	1.2193e4	1.6243e4	1.0348e4
	ICSO	5.9110e2	1.3715e3	3.0002e2	3.9556e6	1.1438e8	1.4113e4
	ECSO	1.1083e3	3.7852e2	1.0808e3	1.3757e4	8.8871e4	9.2386e3
	SHIO	3.5886e2	3.8100e2	**3.0000e2**	1.9168e4	1.0995e5	6.8916e2
	INFO	6.3154e2	2.5764e3	**3.0000e2**	2.4158e3	6.6430e3	**3.0000e2**
	ALCSODE	**3.5761e2**	**1.2225e3**	3.0201e2	**4.1672e2**	**3.1668e3**	3.0686e2
F14	PSO	4.3204e2	2.2300e2	4.0918e2	2.2040e4	4.6964e3	2.1181e4
	CSO	4.2989e2	2.3438e1	4.2754e2	1.3786e3	4.1279e2	1.3488e3
	L-QCSO	4.0247e2	2.6004e1	4.0011e2	6.9569e2	2.9382e2	6.3642e2
	ICSO	4.0208e2	2.2963e1	4.0030e2	5.5840e2	4.9277e2	4.1015e2
	ECSO	4.4100e2	4.1539e1	4.1380e2	1.6100e3	6.0620e2	1.5690e3
	SHIO	4.0976e2	1.1963e1	4.0665e2	4.8056e2	6.2838e2	**4.0000e2**
	INFO	4.0083e2	**5.9497**	**4.0000e2**	4.3623e2	3.8766e2	**4.0000e2**
	ALCSODE	**4.0065e2**	1.0023e1	**4.0000e2**	**4.1052e2**	**1.6400e2**	**4.0000e2**
F15	PSO	6.3579e2	8.2919	6.3446e2	6.8799e2	1.0618e1	6.8505e2
	CSO	6.0600e2	3.2536	6.0528e2	6.1918e2	3.2556	6.1854e2
	L-QCSO	6.0093e2	2.3546	6.0060e2	6.2087e2	**2.2176**	6.2060e2
	ICSO	6.0352e2	2.3401	6.0324e2	6.1784e2	4.0763	6.1697e2
	ECSO	6.1522e2	2.2993	6.1402e2	6.2467e2	3.4395	6.2439e2
	SHIO	6.0255e2	2.7280	6.0208e2	6.1064e2	6.0517	6.0847e2
	INFO	6.0070e2	2.5381	**6.0000e2**	6.0121e2	5.1117	**6.0000e2**
	ALCSODE	**6.0015e2**	**2.0986**	**6.0000e2**	**6.0016e2**	2.3491	**6.0000e2**
F16	PSO	8.2662e2	5.4037	8.2600e2	1.1815e3	2.3858e1	1.1780e3
	CSO	8.0007e2	7.6322e-1	**8.0000e2**	8.1058e2	6.1036	8.1000e2
	L-QCSO	8.0005e2	5.4844e-1	**8.0000e2**	8.1039e2	**3.2107**	8.1000e2
	ICSO	8.0004e2	6.4170e-1	**8.0000e2**	8.1348e2	6.6564	8.1300e2
	ECSO	8.0008e2	1.1140	**8.0000e2**	8.2844e2	6.9513	8.2800e2
	SHIO	8.0001e2	**1.4784e-1**	**8.0000e2**	8.0102e2	6.2935	**8.0000e2**
	INFO	8.2100e2	8.6180e-1	8.1200e2	8.1040e2	4.2054	8.1000e2
	ALCSODE	**8.0001e2**	3.0299e-1	**8.0000e2**	**8.0021e2**	4.3630	**8.0000e2**

TABLE VI. COMPARISON OF EXPERIMENTAL RESULTS FROM F17 TO F24

Function	Algorithm	Mean(dim=10)	Std dev(dim=10)	Best(dim=10)	Mean(dim=50)	Std dev(dim=50)	Best(dim=50)
F17	PSO	2.6326e3	5.6572e2	2.5372e3	2.7243e4	8.5072e3	2.4246e4
	CSO	9.2941e2	8.7053e1	9.2097e2	1.8570e3	7.6056e2	1.7195e3
	L-QCSO	9.0734e2	5.1628e1	9.0183e2	1.8656e3	**3.1121e2**	1.8282e3
	ICSO	9.2844e2	6.9121e1	9.2023e2	1.5213e3	9.5674e2	1.4210e3
	ECSO	9.5393e2	1.3974e2	9.4287e2	3.0568e3	9.3457e2	2.9835e3
	SHIO	9.0695e2	5.4043e1	9.0026e2	4.5717e3	1.2557e3	3.8546e3
	INFO	9.0316e2	4.5299e1	**9.0000e2**	9.8810e2	7.7951e2	9.0032e2
	ALCSODE	**9.0121e2**	**2.6958e1**	**9.0000e2**	**9.2830e2**	5.4272e2	**9.0000e2**
F18	PSO	1.5642e7	1.7129e8	1.0142e4	2.1166e10	5.6262e9	2.0352e10
	CSO	3.8821e5	4.3947e6	1.8530e3	1.4970e9	7.8855e8	1.4391e9
	L-QCSO	7.7114e5	1.4315e7	1.8059e3	2.6978e9	5.0931e8	2.6678e9
	ICSO	**1.5312e5**	**2.7530e6**	2.1249e3	4.6877e7	7.9032e8	4.0565e4
	ECSO	2.0135e6	2.2907e7	2.1130e3	1.1210e9	1.5212e9	1.0253e9
	SHIO	4.8034e5	1.4679e7	**1.8004e3**	7.0391e7	7.6071e8	1.8015e3
	INFO	9.7803e5	1.2583e7	1.8005e3	3.8950e7	3.7818e8	1.8015e3
	ALCSODE	3.1144e5	9.1831e6	**1.8004e3**	**8.8508e6**	**1.7131e8**	**1.8014e3**
F19	PSO	2.0669e3	2.1771e1	2.0637e3	1.1338e4	7.4889e3	1.0181e4
	CSO	2.0217e3	1.7799e1	2.0194e3	2.9263e3	2.0822e3	2.7509e3
	L-QCSO	2.0083e3	5.8092e1	2.0038e3	3.3907e3	4.2409e3	3.0374e3
	ICSO	2.0476e3	2.5859e1	2.0457e3	2.5364e3	**1.1565e3**	2.1599e3
	ECSO	2.0585e3	6.9333e1	2.0415e3	3.1794e3	4.5674e3	2.9015e3
	SHIO	2.0213e3	**1.6653e1**	2.0190e3	3.4178e3	4.1336e3	2.1328e3
	INFO	2.0040e3	2.8061e1	**2.0000e3**	2.5296e3	3.1631e3	**2.0009e3**
	ALCSODE	**2.0026e3**	6.0033e1	2.0006e3	**2.1070e3**	3.0972e3	**2.0009e3**
F20	PSO	3.1400e3	2.2436e3	2.9085e3	3.9415e4	1.7860e4	3.6786e4
	CSO	2.2278e3	7.7401e1	2.2209e3	5.9860e3	7.0967e4	3.5996e3
	L-QCSO	2.2176e3	1.5813e2	2.2093e3	4.7364e3	9.5444e3	3.9668e3
	ICSO	2.2103e3	1.1385e2	2.2038e3	4.5770e3	2.9931e3	2.6551e3
	ECSO	2.3216e3	1.6637e2	2.3078e3	4.7201e3	2.9933e3	4.4365e3
	SHIO	2.2094e3	**4.3933e1**	2.2015e3	2.9708e3	**1.5570e3**	2.3036e3
	INFO	2.2117e3	8.6726e1	2.2001e3	4.4880e3	1.0877e4	2.2021e3
	ALCSODE	**2204.03632**	7.9725e1	**2.2000e3**	**2.7890e3**	2.4339e3	**2.2004e3**
F21	PSO	1.2330e8	7.9524e8	6.5777e6	3.1866e10	2.3750e9	3.1437e10
	CSO	8.9897e7	7.8170e7	8.3546e7	1.1761e9	5.2367e8	1.1365e9
	L-QCSO	1.2882e7	5.8898e7	1.5843e6	7.8225e8	2.6790e8	7.6479e8
	ICSO	2.7389e6	5.7325e7	4.2223e3	1.7390e8	7.6455e8	7.0218e4
	ECSO	3.2675e7	9.7277e7	2.5080e7	1.4227e9	6.9794e8	1.3683e9
	SHIO	3.7806e6	5.4935e7	**2.5000e3**	9.2896e7	7.0122e8	**2.5000e3**
	INFO	3.3374e6	3.4619e7	**2.5000e3**	4.0015e7	4.0280e8	**2.5000e3**
	ALCSODE	**5.4514e5**	**1.0724e7**	**2.5000e3**	**1.4465e7**	**2.7567e8**	**2.5000e3**
F22	PSO	2.6580e3	2.3430e2	2.6236e3	6.7897e3	2.7290e3	6.0614e3
	CSO	2.5191e3	2.4691e1	2.5159e3	2.9446e3	3.1401e2	2.8995e3
	L-QCSO	2.5091e3	1.6583e1	2.5061e3	2.9955e3	**1.2763e2**	2.9810e3
	ICSO	2.5139e3	2.2874e1	2.5113e3	2.7975e3	3.0825e2	2.7530e3
	ECSO	2.5249e3	4.2371e1	2.5226e3	3.2843e3	4.9190e2	3.2497e3
	SHIO	2.5127e3	**1.5887e1**	2.5104e3	3.5894e3	4.6065e2	3.4663e3
	INFO	2.5082e3	2.3643e1	**2.5030e3**	2.5465e3	2.0419e2	**2.5156e3**
	ALCSODE	**2.5039e3**	1.8663e1	**2.5030e3**	**2.5247e3**	1.6702e2	**2.5156e3**
F23	PSO	7.0138e3	4.7479e4	5.0129e3	9.9541e4	9.9534e5	3.8218e4
	CSO	3.1115e3	5.3457e2	3.0597e3	1.7654e4	2.5887e5	5.8710e3
	L-QCSO	3.6831e3	6.8820e2	3.1785e3	4.5210e3	5.2698e3	3.8771e3
	ICSO	2.9424e3	**3.0850e2**	**2.8207e3**	9.3559e4	2.5690e6	9.3078e3
	ECSO	3.1238e3	3.2901e2	3.0977e3	5.8154e3	**1.8505e3**	5.6269e3
	SHIO	2.9200e3	4.3141e2	2.8208e3	6.1869e3	5.3195e3	2.8485e3
	INFO	2.9486e3	8.6845e2	**2.8207e3**	1.0965e4	5.4872e4	2.8236e3
	ALCSODE	**2.9199e3**	3.0715e3	**2.8207e3**	**3.1587e3**	7.3032e3	**2.8218e3**
F24	PSO	7.8219e7	5.9782e8	4.7312e4	9.0122e9	3.6756e9	8.3934e9
	CSO	3.0742e7	5.4115e7	2.6376e7	7.6600e8	4.7189e8	7.2143e8
	L-QCSO	2.0039e7	3.6293e7	5.3834e6	1.1190e9	4.8782e8	1.0777e9
	ICSO	3.0929e6	4.9371e7	1.2905e4	3.5972e8	9.6735e8	5.8719e4
	ECSO	4.8070e7	5.1804e7	4.0272e7	1.0143e9	1.0010e9	9.3863e8
	SHIO	1.7249e6	2.2613e7	**2.9522e3**	1.4510e8	8.2402e8	**2.9614e3**
	INFO	2.2004e6	**1.8753e7**	**2.9522e3**	**4.0366e7**	**2.9775e8**	**2.9614e3**
	ALCSODE	**9.6401e5**	2.5747e7	**2.9522e3**	5.8906e7	5.0165e8	**2.9614e3**

TABLE VII. Wilcoxon Test Results for Best Values with Fixed Iterations

	Dim	PSO		CSO		L-QCSO		ICSO		ECSO		SHIO		INFO		ALCSODE
		Wilcox	Rank	Wilcox	Rank	Wilcox	Rank	Wilcox	Rank	Wilcox	Rank	Wilcox	Rank	Wilcox	Rank	Rank
F1	10	+	5	+	7	+	6	+	3	+	8	+	4	+	2	1
	50	+	8	+	5	+	6	+	3	+	7	+	4	+	2	1
F2	10	+	8	+	6	+	4	+	5	+	7	+	2	+	3	1
	50	+	8	+	6	+	5	+	4	+	7	+	3	+	2	1
F3	10	+	8	+	6	+	5	+	4	+	7	+	2	+	3	1
	50	+	7	+	5	-	4	+	6	+	8	-	1	-	2	3
F4	10	+	6	+	7	+	5	+	4	+	8	+	2	+	3	1
	50	+	8	+	5	+	6	+	4	+	7	+	3	+	2	1
F5	10	+	8	+	6	+	5	+	4	+	7	+	2	+	3	1
	50	+	8	+	6	+	5	+	4	+	7	+	3	+	2	1
F6	10	+	8	+	6	+	3	+	5	+	7	+	4	=	1	1
	50	+	8	+	5	+	4	+	7	+	6	+	3	+	2	1
F7	10	+	5	+	7	+	6	+	4	+	8	+	2	+	3	1
	50	+	8	+	6	+	5	+	4	+	7	+	3	+	2	1
F8	10	+	8	+	5	+	4	+	7	+	6	+	2	+	3	1
	50	+	8	+	6	+	4	+	7	+	5	+	3	+	2	1
F9	10	=	1	=	1	=	1	+	7	+	8	=	1	=	1	1
	50	=	1	+	6	=	1	+	7	+	8	=	1	=	1	1
F10	10	+	7	+	6	+	5	+	4	+	8	-	1	+	3	2
	50	+	8	+	6	+	5	-	3	+	7	-	2	-	1	4
F11	10	+	8	+	5	+	6	+	3	+	7	+	4	+	2	1
	50	+	8	+	5	+	6	+	4	+	7	=	1	=	1	1
F12	10	+	3	+	7	+	5	+	6	+	8	+	4	=	1	1
	50	+	8	+	4	+	7	+	5	+	6	+	3	-	1	2
F13	10	+	8	+	5	+	7	+	4	+	6	-	1	-	1	3
	50	+	8	+	4	+	6	+	7	+	5	+	3	-	1	2
F14	10	+	6	+	8	+	3	+	4	+	7	+	5	=	1	1
	50	+	8	+	6	+	5	+	4	+	7	=	1	=	1	1
F15	10	+	8	+	6	+	3	+	5	+	7	+	4	=	1	1
	50	+	8	+	5	+	6	+	4	+	7	+	3	=	1	1
F16	10	+	8	=	1	=	1	=	1	=	1	=	1	+	7	1
	50	+	8	+	2	+	2	+	6	+	7	=	1	+	2	1
F17	10	+	8	+	6	+	4	+	5	+	7	+	3	=	1	1
	50	+	8	+	4	+	5	+	3	+	6	+	7	+	2	1
F18	10	+	8	+	5	+	4	+	7	+	6	=	1	+	3	1
	50	+	8	+	6	+	7	+	4	+	5	+	2	+	2	1
F19	10	+	8	+	4	+	3	+	7	+	6	+	4	-	1	2
	50	+	8	+	5	+	7	+	4	+	6	+	3	=	1	1
F20	10	+	8	+	6	+	5	+	4	+	7	+	3	+	2	1
	50	+	8	+	6	+	5	+	4	+	7	+	3	+	2	1
F21	10	+	6	+	8	+	5	+	4	+	7	=	1	=	1	1
	50	+	8	+	7	+	5	+	4	+	6	=	1	=	1	1
F22	10	+	8	+	6	+	3	+	5	+	7	+	4	=	1	1
	50	+	8	+	4	+	5	+	3	+	6	+	7	=	1	1
F23	10	+	8	+	5	+	7	=	1	+	6	+	4	=	1	1
	50	+	8	+	6	+	4	+	7	+	5	+	3	+	2	1
F24	10	+	5	+	7	+	6	+	4	+	8	=	1	=	1	1
	50	+	8	+	5	+	7	+	4	+	6	=	1	=	1	1
Avg			7.25		5.43		4.75		4.56		6.64		2.58		1.77	1.22
Final			8		6		5		4		7		3		2	1

Step 2: Population initialization based on the normal distribution, calculate the fitness using the objective function $f(x)$, and record the optimal value.

Step 3: Elite selection. Select top $\eta * N$ individuals. The residual population is categorized into roosters, hens, and chicks based on fitness hierarchy.

Step 4: Adaptive perturbation-based differential evolution for rooster behavior updating. First, the adaptive differential mutation vector is computed using the adaptive scaling factor F_t through Eq. (7). Then, the individual-adaptive perturbation factor δ_i is calculated, where inferior individuals receive larger perturbations through a fitness-dependent variance σ_i^2, sampled from a zero-mean normal distribution as defined in Eq. (9). Finally, the differential evolution mutation vector is combined with the current rooster vector controlled by the adaptive perturbation factor δ_i to generate the next generation, as defined in Eq. (10). Additionally, a component-wise crossover strategy prevents excessive localization by updating positions dimension-by-dimension, as defined in Eq. (11).

Step 5: Levy flight-based hen behavior optimization. By leveraging the Levy flight algorithm concept, hen behavior is updated in Equation 16 through the integration of dynamic scaling factors λ_1, λ_2 and Levy flight step size L.

Step 6: Chick behavior updating. For each chick i, a randomly selected leader r serves as the reference for position updating in Eq. (17).

Step 7: The fitness values of the population in the new state are evaluated by $f(x)$, with updates occurring only if the new fitness is better than the previous value. Additionally, the elite strategy replaces the current worst-performing individuals.

Step 8: Termination condition. The algorithm terminates if the maximum iteration count is reached, outputting the optimal solution; otherwise, the process returns to Step 3.

F. Time Complexity Analysis

The time complexity of the proposed Adaptive Levy Flight Chicken Swarm Optimization with Differential Evolution algorithm is analyzed with respect to the population size N, problem dimensionality Dim, fitness evaluation cost $f(Dim)$, and the maximum number of iterations T. In each iteration, the algorithm updates the positions of all individuals—including roosters, hens, and chicks—through vector operations such as mutation, adaptive perturbation, crossover, and Levy flight-based steps, each requiring $O(Dim)$ computations per individual and resulting in an overall $O(N \times Dim)$ complexity. Fitness evaluations for all individuals contribute an additional cost of $O(N \times f(Dim))$ per iteration. Although elitism preservation involves selecting elite individuals and replacing low-fitness ones, which may require sorting with a complexity of $O(N \log N)$, this cost is generally minor compared to the dominant vector updates and fitness evaluations. Consequently, the total time complexity per iteration can be approximated by combining these main components as $O(N \times Dim) + O(N \times f(Dim))$, and over T iterations, the overall time complexity is $O\big(T \times N \times (Dim + f(Dim))\big)$. This formulation highlights that the computational cost of ALCSODE scales linearly with population size, problem dimensionality, and the complexity of fitness evaluation, demonstrating its suitability for large-scale and high-dimensional optimization problems.

IV. Simulation Experiment

To validate the effectiveness of the proposed ALSCODE algorithm, extensive experiments are conducted on 24 benchmark functions, including unimodal functions, high-dimensional multimodal functions, and the CEC2022 test suite [47]. This section first presents the experimental configurations and parameter settings of the comparative methods, followed by an analysis and verification of the algorithm's effectiveness based on the experimental results.

A. Experimental Setup

The experiments are conducted on a Windows 11 system equipped with a 2.9 GHz CPU and 16 GB of RAM, using Python 3.10 as the programming environment. The ALSCODE algorithm is evaluated on 24 benchmark functions, including 8 unimodal functions, 9 multimodal functions, 3 hybrid functions, and 4 composition functions. Detailed information, including function names, search ranges, and theoretical global optima, is provided in Table I and Table II, the latter corresponding to the standard CEC2022 benchmark.

The experimental comparison includes the standard Particle Swarm Optimization (PSO) algorithm inspired by bird flocking behavior [48], the standard Chicken Swarm Optimization (CSO) algorithm [33], along with three of its variants (L-QCSO [39], ICSO [36], and ECSO [49]), as well as the metaheuristic algorithms INFO [12] and SHIO [13], which serve as baseline methods. To ensure a fair comparison, all algorithms adopt the same configuration: a population size of 50, a maximum of 1000 iterations, and search spaces of 10 and 50 dimensions. Each algorithm is executed 30 times on each benchmark function, and the average performance is recorded. Additional parameter settings are provided in Table III.

B. Qualitative Results and Discussion of ALCSODE

The qualitative experiments are conducted on the two-dimensional versions of benchmark functions to visually observe the search behavior of the ALSCODE algorithm. For each test function, two types of diagrams are provided: one illustrating the trajectory of the first particle in the first dimension, and the other depicting the average fitness of the particle swarm. The former reflects the search dynamics of a representative particle, while the latter indicates the convergence trend of the entire population.

Fig. 2 shows the trajectory of the first particle in the first dimension, which serves to evaluate the algorithm's behavior at different stages, particularly whether it exhibits abrupt movements in the early iterations and stabilizes in later stages. This figure consists of 24 subplots, each representing the trajectory corresponding to a specific benchmark function. For example, Fig. 2a illustrates the trajectory for the F1 function (Alpine). According to the study by Berg et al. [50], an effective population-based algorithm performs wide-range exploration in the early phase and gradually shifts to local exploitation to achieve high-quality solutions. As shown in the figure, the first particle in ALSCODE exhibits a significant and abrupt movement exceeding 50% of the search space in the early phase, indicating that the algorithm fully utilizes population diversity and individual differences to escape local optima and conduct global exploration. As the iterations proceed, the particle's movement amplitude gradually decreases and the trajectory becomes smoother. This progressive stabilization is attributed to ALSCODE's adaptive perturbation mechanism, which dynamically adjusts the step size of the movement vector based on particle fitness, thereby narrowing the search range and guiding particles toward more promising regions for enhanced local exploitation.

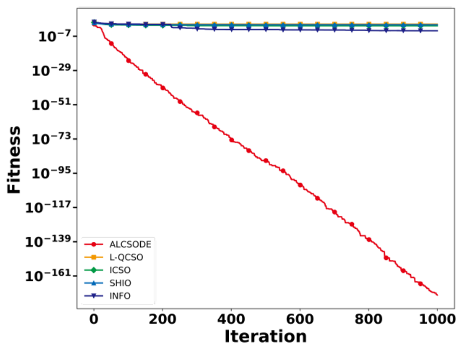
The convergence curves of average fitness in Fig. 3 further validate this behavior. All test functions exhibit a consistently decreasing trend, indicating that the overall solution quality of the population improves continuously throughout the optimization process. This result confirms that ALSCODE effectively optimizes the swarm's search behavior and significantly enhances the accuracy of the approximated optimal solutions.

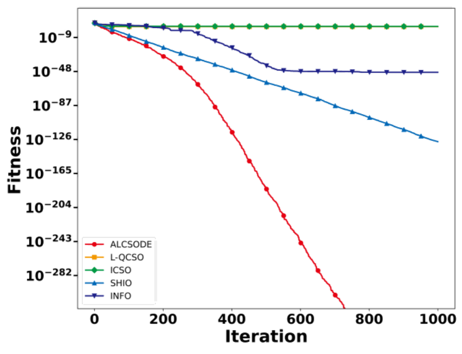
In summary, Fig. 2 and Fig. 3 jointly demonstrate that ALSCODE maintains a well-balanced dynamic between exploration and exploitation. The large-scale fluctuations in the early stage enhance diversity and global coverage, while the gradually decreasing fluctuations in the later stage ensure convergence and fine-tuning. Ultimately, the particle trajectories on nearly all benchmark functions converge to a stable point, verifying that ALSCODE achieves a smooth transition from global exploration to local exploitation and exhibits strong convergence performance and robustness.

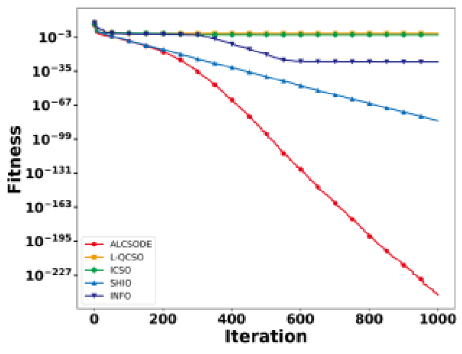
C. ALCSODE Statistical Results and Comparison with Other Algorithms

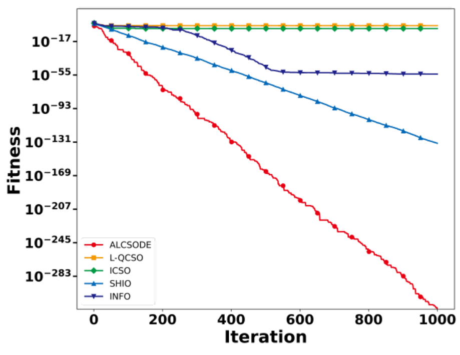
Functions F1 to F24 (including the CEC2022 test suite) are evaluated under 10-dimensional and 50-dimensional settings. Experiments are based on data obtained from 30 independent runs for each function, and performance metrics include the best value, mean value, and standard deviation. The results are summarized in Table IV, Table V, and Table VI, where the best results for each function are highlighted in bold.

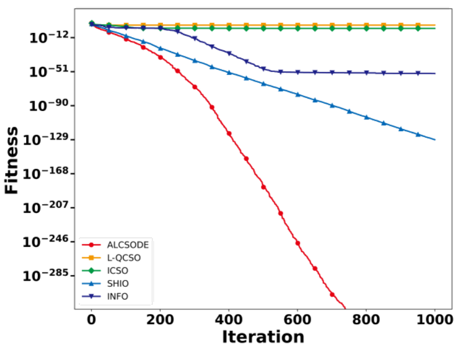
To comprehensively validate the performance of the ALSCODE algorithm, it is compared with the classical Chicken Swarm Optimization (CSO) and its three variants L-QCSO, ICSO, and ECSO, as well as representative metaheuristic algorithms from outside the domain, including PSO, INFO, and SHIO. Fig. 4 to Fig. 7 present the convergence curves of a randomly selected run among the 30 executions. To reduce visual clutter, only the top-performing or representative algorithms are shown: L-QCSO and ICSO among chicken swarm algorithms, and INFO and SHIO among metaheuristic algorithms. This choice facilitates a clearer comparison of convergence behavior. Comparative results indicate that, under different dimensional conditions, ALSCODE exhibits significant advantages in solution accuracy, result stability, and convergence efficiency.

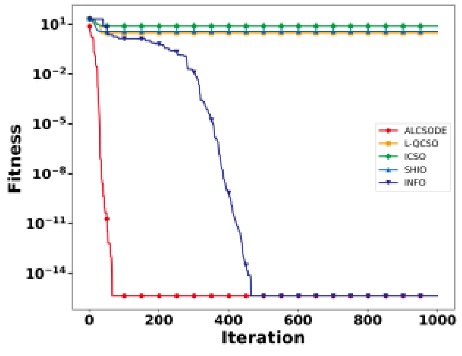

(a) F1

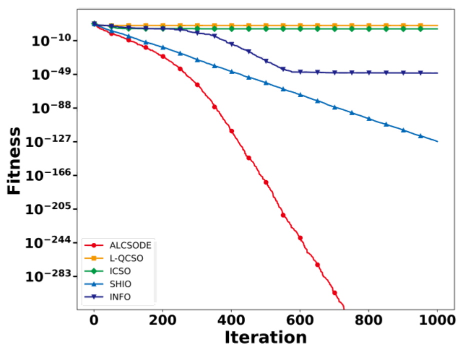

(b) F2

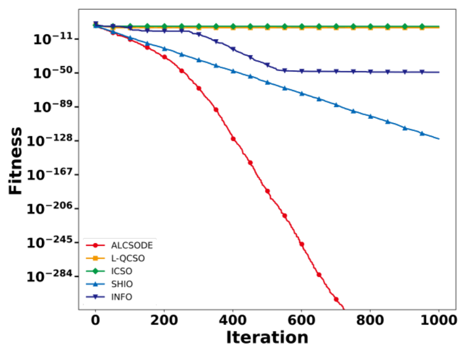

(c) F3

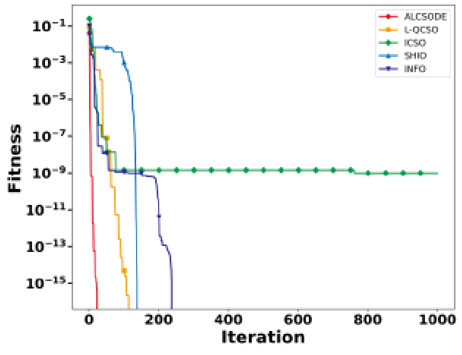

(d) F4

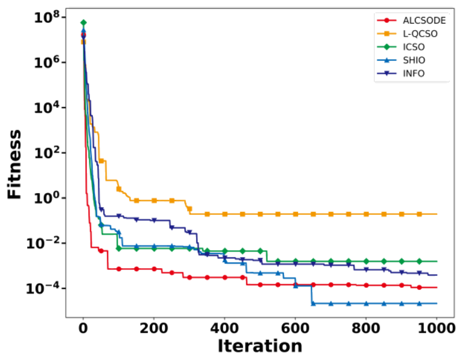

(e) F5

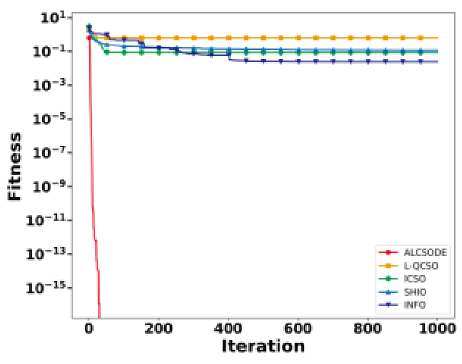

(f) F6


(g) F7


(h) F8


(i) F9


(j) F10


(k) F11

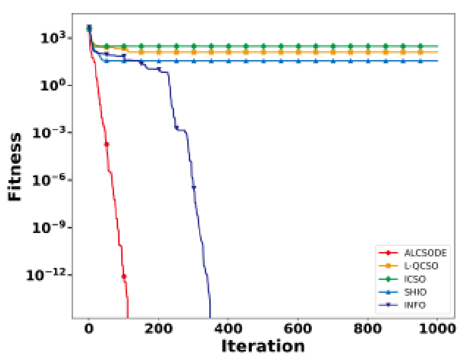
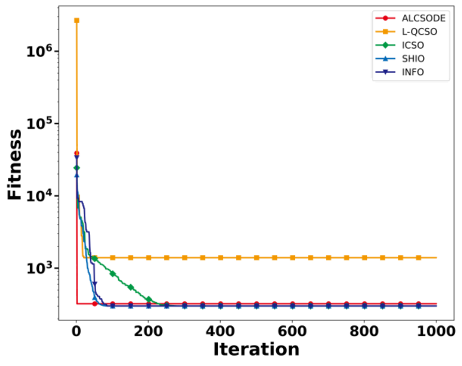
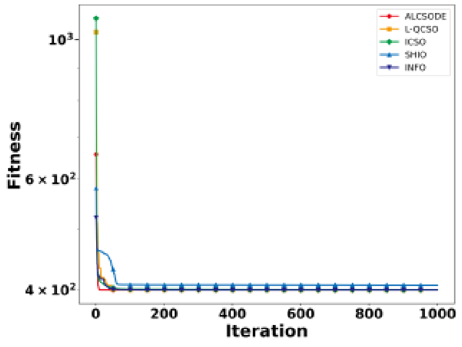
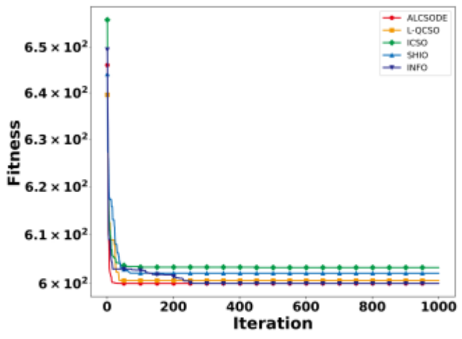
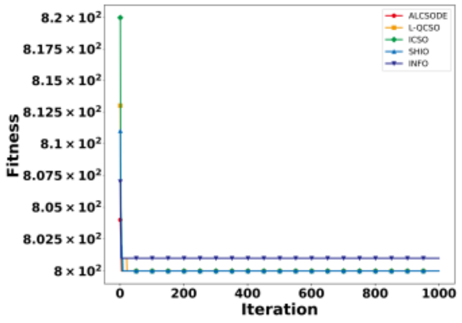

(l) F12

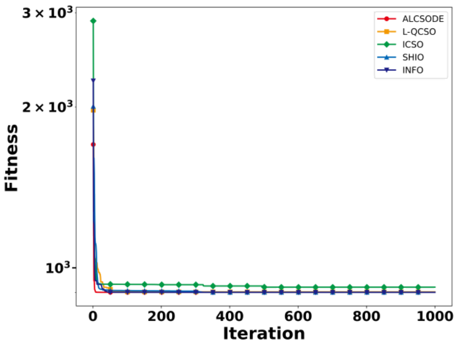
Fig. 4. Performance comparison of convergence curve on F1-F12 test functions in 10D.

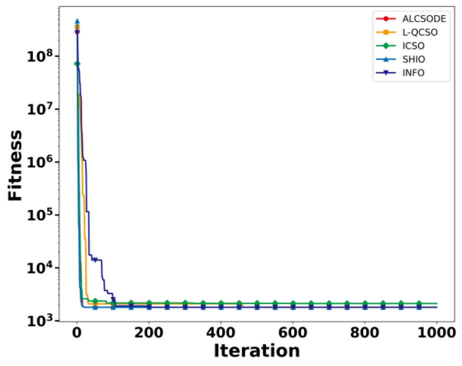

(a) F13

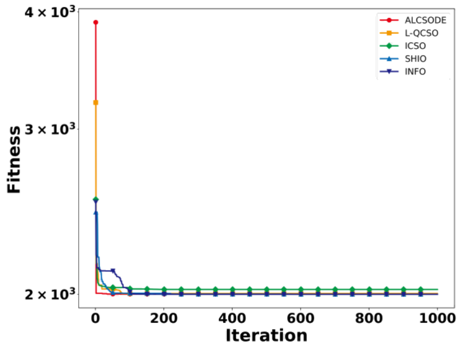

(b) F14

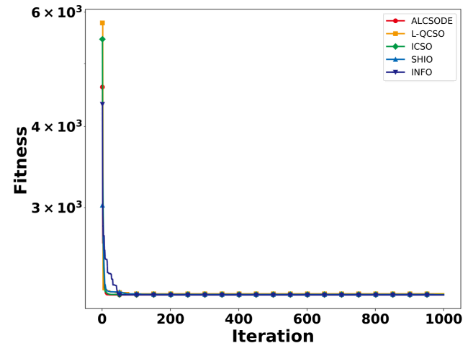

(c) F15

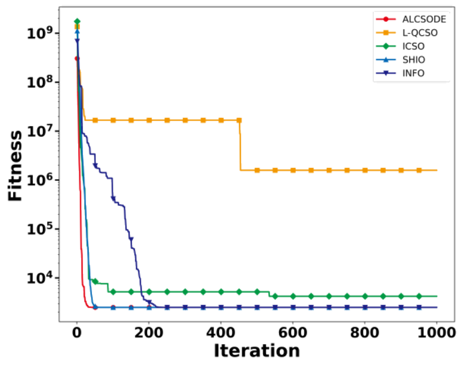

(d) F16

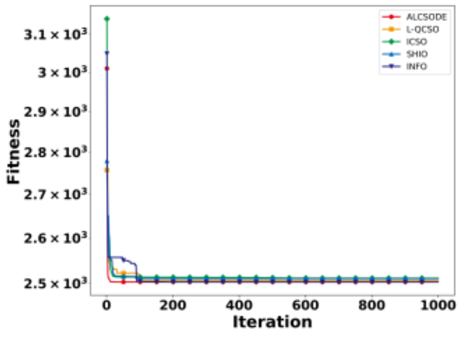

(e) F17

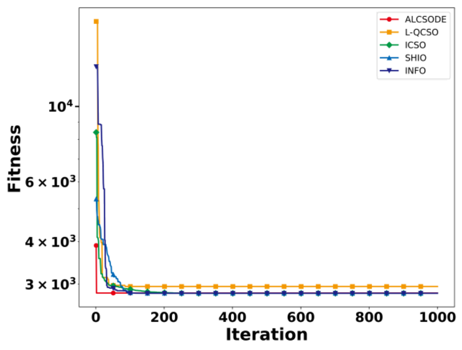

(f) F18


(g) F19


(h) F20


(i) F21


(j) F22


(k) F23

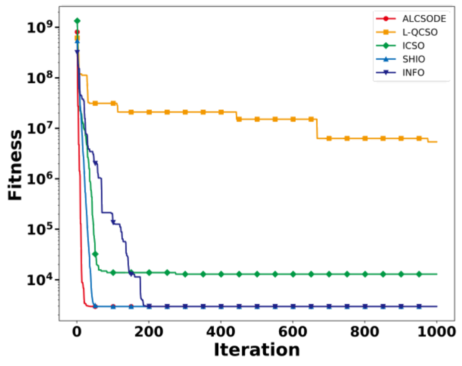
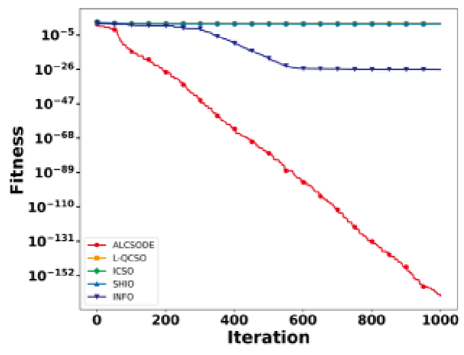
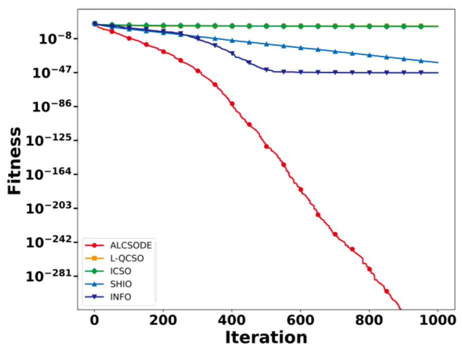

(l) F24

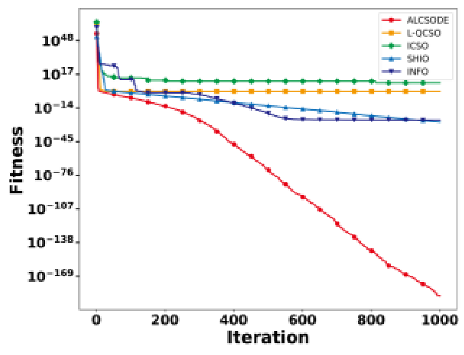
Fig. 5. Performance comparison of convergence curve on F13-F24 test functions in 10D.



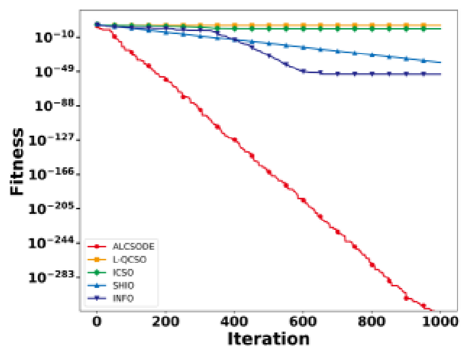
(a) F1



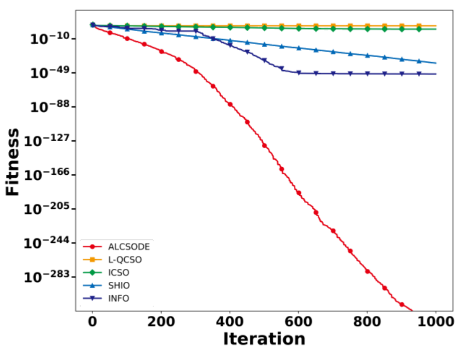
(b) F2



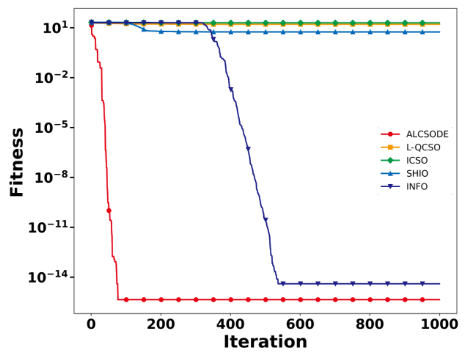
(c) F3



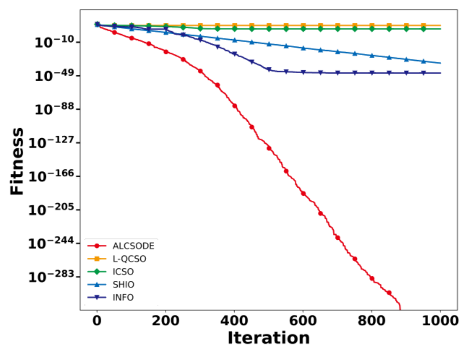
(d) F4



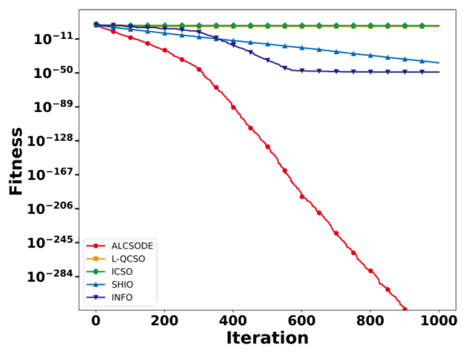
(e) F5



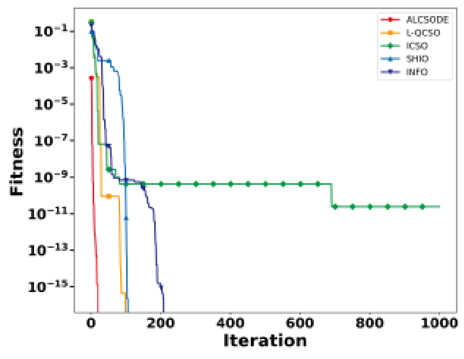
(f) F6



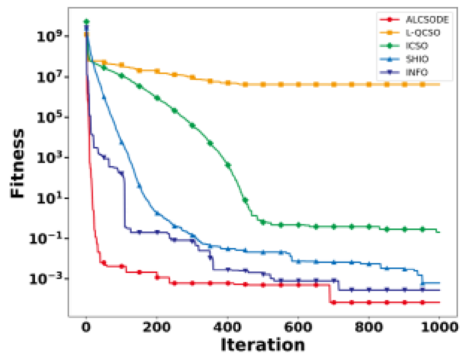
(g) F7



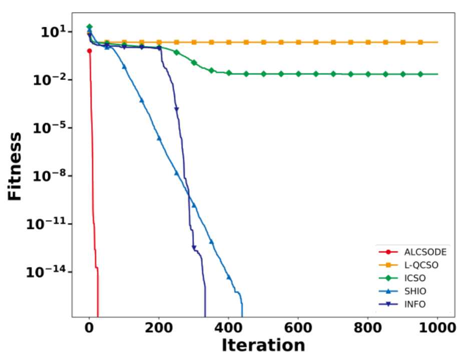
(h) F8



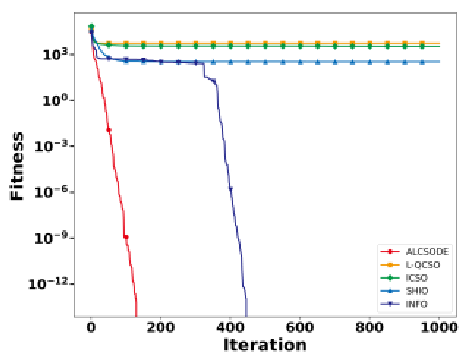
(i) F9



(j) F10

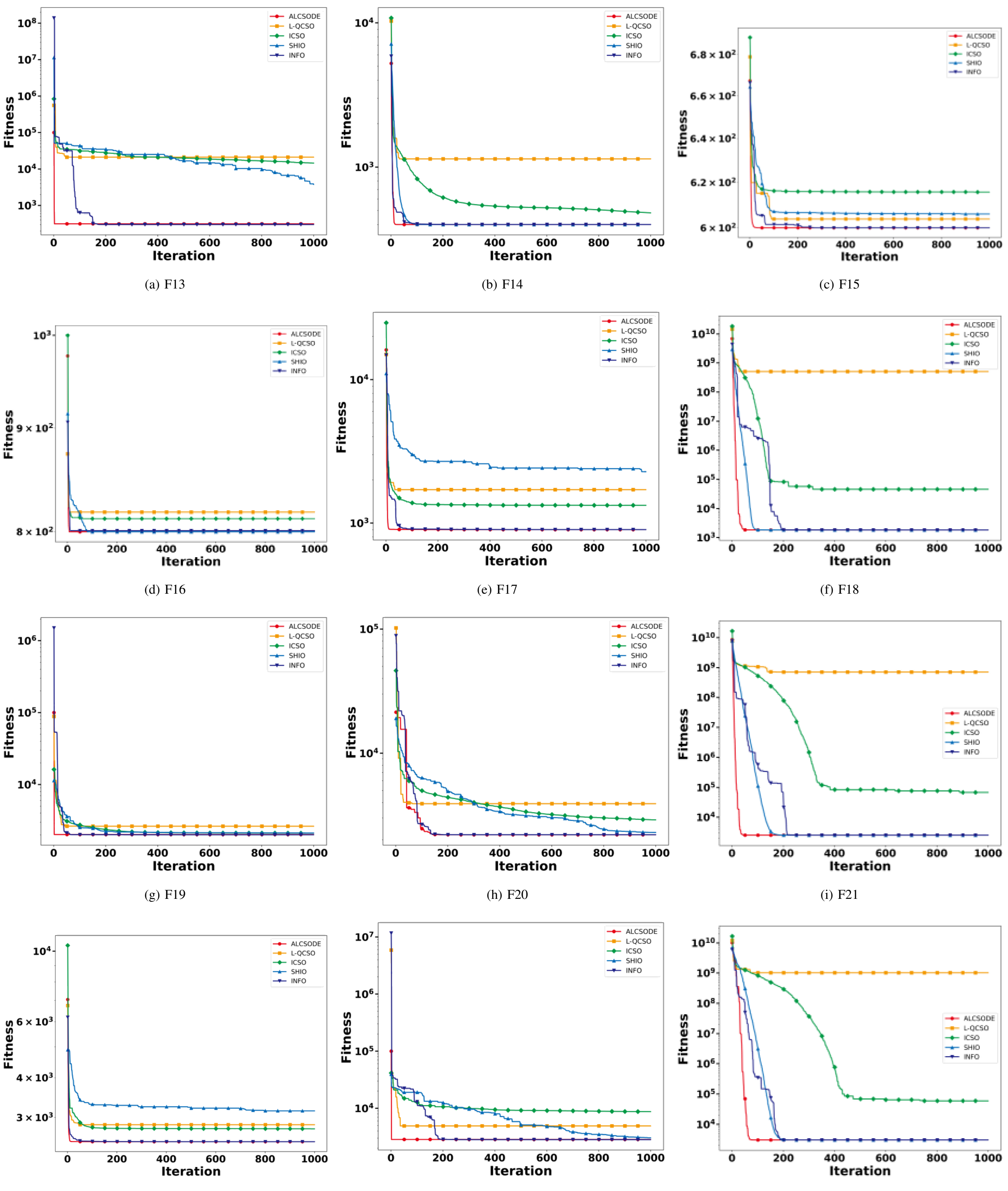


(k) F11



(l) F12

Fig. 6. Performance comparison of convergence curve on F1-F12 test functions in 50D.



(a) F13 (b) F14 (c) F15

(d) F16 (e) F17 (f) F18

(g) F19 (h) F20 (i) F21

(j) F22 (k) F23 (l) F24

Fig. 7. Performance comparison of convergence curve on F13-F24 test functions in 50D.

1) Evaluation of unimodal banchmark: Unimodal functions F2, F3, F4, F5, F7, F8, F10, and F13 contain only a single global optimum and no local optima, making them ideal benchmarks for evaluating the development capabilities of optimization algorithms. According to the comparison of Mean, Standard Deviation, and Best results in both 10-dimensional and 50-dimensional cases, ALCSODE successfully reaches the global optimum on F2, F5, and F8, and demonstrates the best performance on all functions except F10 and F13. Compared to other competing algorithms, ALCSODE exhibits superior convergence accuracy and stability on these unimodal functions. Overall, the simulation results clearly indicate that ALCSODE possesses a significant performance advantage in unimodal function optimization tasks.

2) Evaluation of multimodal banchmark: Functions F1, F6, F9, F11, F12, F14, F15, F16, and F17 are typical high-dimensional multimodal functions that contain not only a primary global optimum but also multiple local optima, making them well-suited for evaluating the exploration capability of optimization algorithms. Tables IV to VI present the optimization results of the proposed algorithm ALCSODE compared with seven other competing algorithms on these functions. Benefiting from its strong exploration ability, ALCSODE is able to locate the global optimum for functions F9, F11, F12, F14, F15, F16, and F17 after identifying the promising regions. In the optimization of F1 and F6, ALCSODE also demonstrates superior performance. The simulation results indicate that ALCSODE exhibits satisfactory exploration capability in accurately scanning the search space and avoiding local optima.

3) Evaluation of CEC 2022 benchmark: CEC2022 serves as an effective benchmark for evaluating the performance of optimization algorithms and their ability to solve complex optimization problems. It requires algorithms to handle multi-modality, multi-scaled landscapes, and fused complex function shapes. The test suite consists of 12 benchmark functions, including unimodal, multimodal, hybrid, and composition functions. Specifically, F13 is a unimodal function; F14–F17 are multimodal functions; F18–F20 are hybrid functions (which may exhibit either unimodal or multimodal characteristics); and F21–F24 are composition functions with multimodal properties. Tables V and VI present the optimization results of the proposed ALCSODE algorithm compared with seven competing algorithms on the CEC2022 benchmark set. The results show that ALCSODE achieves the global optimum for F14, F15, F16, F17, and F20. For F18, F19, F21, F22, F23, and F24, ALCSODE outperforms all competitors except for F19 in the 10-dimensional case, where the global optimum is not fully reached. Overall, ALCSODE demonstrates superior optimization performance in most test cases compared to the other algorithms.

4) Statistical analysis: To provide a clear and intuitive comparison of algorithmic performance, the non-parametric Wilcoxon signed-rank test is employed to assess the statistical significance of the results. Using the best convergence value as the evaluation criterion, statistical analysis is conducted based on the average outcomes from 30 independent runs, aiming to systematically examine the effectiveness of ALCSODE. In the test outcomes, a "+" symbol indicates that ALCSODE performs better than the compared algorithm, "-" indicates relatively lower performance, and "=" denotes no significant difference. The "Rank" column represents the relative ranking in terms of solution accuracy. As shown in Table VII, ALCSODE performs competitively across most of the twelve benchmark functions. Although ALCSODE does not achieve the best performance among all algorithms on F3, F10, F12, and F13 in the 10-dimensional case, and on F10, F13, and F19 in the 50-dimensional case, slightly inferior to INFO or SHIO, it exhibits significant advantages on the remaining functions, attaining 85.5% (41 out of 48) of the best convergence results. Furthermore, the Wilcoxon rank-sum test is applied to validate the consistency of performance rankings across different dimensions. The overall results indicate that ALCSODE demonstrates strong competitiveness among all the compared algorithms and exhibits advantages in convergence accuracy from a statistical perspective.

D. Scalability and Stability Analysis of ALCSODE

This subsection analyzes the scalability and stability of ALCSODE. For scalability analysis, ALCSODE is tested on functions F1 to F24 under 10-dimensional and 50-dimensional settings, with the simulation results presented in Tables IV to VI. The analysis shows that although the performance of all algorithms declines as the dimensionality increases from 10 to 50, ALCSODE experiences a relatively smaller performance drop, demonstrating strong robustness. It continues to achieve the same optimal convergence accuracy on several functions as in the 10-dimensional case. In contrast, traditional PSO and CSO algorithms exhibit significant performance degradation in high-dimensional spaces. Other CSO variants, as well as metaheuristic algorithms such as INFO and SHIO, show moderate performance but are more sensitive to initial settings, with greater convergence fluctuations and a higher risk of premature convergence.

In terms of algorithmic stability, ALCSODE generally exhibits lower standard deviations compared to other algorithms. For instance, in both 10-dimensional and 50-dimensional experiments, ALCSODE achieves the smallest standard deviation values in most cases for functions F1, F2, F3, F4, F9, F11, F12, F13, and F21. This indicates a lower sensitivity to initial conditions and stronger resistance to random disturbances, which is particularly important for maintaining stable convergence in noisy engineering optimization environments.

V. Conclusion and Future Work

This study proposes Adaptive Levy Flight Chicken Swarm Optimization with Differential Evolution (ALCSODE) to overcome traditional CSO limitations like local optima entrapment, weak perturbation, and fixed parameters. ALCSODE enhances exploration and exploitation balance by introducing an adaptive differential perturbation in the rooster phase, combined with fitness-based step-size control and a non-inertia weight strategy. The hen phase applies dynamic scaling and Levy flights to improve local optima escape and convergence speed. An elitism preservation mechanism maintains diversity and guides solutions via information sharing. The mathematical modeling of ALCSODE and descriptions of its steps and strategies are also presented.

The performance of ALCSODE is validated on 24 benchmark functions, including unimodal, multimodal, and the

CEC2022 test suite. Results on unimodal functions demonstrate ALCSODE's strong exploitation ability, effectively converging toward the global optimum. For multimodal functions, ALCSODE exhibits robust exploration capability, successfully escaping local optima and locating the primary optimal regions. To comprehensively evaluate its effectiveness, ALCSODE is compared with seven well-known algorithms, including PSO, CSO, L-QCSO, ICSO, ECSO, SHIO, and INFO. Experimental results indicate that ALCSODE achieves superior balance between exploration and exploitation, outperforming these competitors and delivering more competitive optimization results.

However, the integration of multiple mechanisms, such as differential computation, dynamic step adjustment, and elitism preservation, may increase computational complexity, potentially affecting efficiency in large-scale or real-time optimization. Future research may focus on enhancing inter-module synergy, developing automated parameter adjustment mechanisms to reduce hyperparameter sensitivity, and exploring dimensionality reduction or decomposition strategies to improve efficiency in high-dimensional problems.

References

[1] S. Mirjalili, "The ant lion optimizer," *Advances in engineering software*, vol. 83, pp. 80–98, 2015.

[2] R. G. Rakotonirainy and J. H. van Vuuren, "Improved metaheuristics for the two-dimensional strip packing problem," *Applied Soft Computing*, vol. 92, p. 106268, 2020.

[3] S.-H. Liu, M. Mernik, D. Hrnčič, and M. Črepinšek, "A parameter control method of evolutionary algorithms using exploration and exploitation measures with a practical application for fitting sovova's mass transfer model," *Applied Soft Computing*, vol. 13, no. 9, pp. 3792–3805, 2013.

[4] A. Lambora, K. Gupta, and K. Chopra, "Genetic algorithm-a literature review," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. IEEE, 2019, pp. 380–384.

[5] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[6] H.-G. Beyer, "Evolution strategies," *Scholarpedia*, vol. 2, no. 8, p. 1965, 2007.

[7] S. Kirkpatrick, C. Gelatt, M. Vecchi *et al.*, "Optimization by simulated annealing. science. v220," *Lever & B. Richards (1994) The applications of generic planning architecture to ight allocation CHIC deliverable*, vol. 4, no. 3, 1983.

[8] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "Gsa: a gravitational search algorithm," *Information sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[9] H. R. Bouchekara, "Optimal design of electromagnetic devices using a black-hole-based optimization technique," *IEEE Transactions on Magnetics*, vol. 49, no. 12, pp. 5709–5714, 2013.

[10] F. Qin, A. M. Zain, and K.-Q. Zhou, "Harmony search algorithm and related variants: A systematic review," *Swarm and Evolutionary Computation*, vol. 74, p. 101126, 2022.

[11] F. Qin, A. M. Zain, K.-Q. Zhou, and D.-B. Zhuo, "Hybrid weighted fuzzy production rule extraction utilizing modified harmony search and bpnn," *Scientific Reports*, vol. 15, no. 1, p. 11012, 2025.

[12] I. Ahmadianfar, A. A. Heidari, S. Noshadian, H. Chen, and A. H. Gandomi, "Info: An efficient optimization algorithm based on weighted mean of vectors," *Expert Systems with Applications*, vol. 195, p. 116516, 2022.

[13] H. N. Fakhouri, F. Hamad, and A. Alawamrah, "Success history intelligent optimizer," *The Journal of Supercomputing*, vol. 78, no. 5, pp. 6461–6502, 2022.

[14] A. Chakraborty and A. K. Kar, "Swarm intelligence: A review of algorithms," *Nature-inspired Computing and Optimization: Theory and Applications*, pp. 475–494, 2017.

[15] A.-Q. Tian, F.-F. Liu, and H.-X. Lv, "Snow geese algorithm: A novel migration-inspired meta-heuristic algorithm for constrained engineering optimization problems," *Applied Mathematical Modelling*, vol. 126, pp. 327–347, 2024.

[16] G. Hu, M. Cheng, E. H. Houssein, A. G. Hussien, and L. Abualigah, "Sdo: A novel sled dog-inspired optimizer for solving engineering problems," *Advanced Engineering Informatics*, vol. 62, p. 102783, 2024.

[17] A. Alkharsan and O. Ata, "Hawkfish optimization algorithm: A gender-bending approach for solving complex optimization problems," *Electronics*, vol. 14, no. 3, p. 611, 2025.

[18] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[19] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[20] R. Akbari, A. Mohammadi, and K. Ziarati, "A novel bee swarm optimization algorithm for numerical function optimization," *Communications in Nonlinear Science and Numerical Simulation*, vol. 15, no. 10, pp. 3142–3155, 2010.

[21] F. Pourpanah, R. Wang, C. P. Lim, X.-Z. Wang, and D. Yazdani, "A review of artificial fish swarm algorithms: Recent advances and applications," *Artificial Intelligence Review*, vol. 56, no. 3, pp. 1867–1903, 2023.

[22] X.-S. Yang, "A new metaheuristic bat-inspired algorithm (2010)," *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pp. 65–74, 2023.

[23] X.-S. Yang and S. Deb, "Cuckoo search via lévy flights," in *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*. IEEE, 2009, pp. 210–214.

[24] S. Ye, K. Zhou, A. M. Zain, F. Wang, and Y. Yusoff, "A modified harmony search algorithm and its applications in weighted fuzzy production rule extraction," *Frontiers of Information Technology & Electronic Engineering*, vol. 24, no. 11, pp. 1574–1590, 2023.

[25] A. H. Gandomi and A. H. Alavi, "Krill herd: a new bio-inspired optimization algorithm," *Communications in nonlinear science and numerical simulation*, vol. 17, no. 12, pp. 4831–4845, 2012.

[26] M. Mitić, N. Vuković, M. Petrović, and Z. Miljković, "Chaotic fruit fly optimization algorithm," *Knowledge-based systems*, vol. 89, pp. 446–458, 2015.

[27] J. James and V. O. Li, "A social spider algorithm for global optimization," *Applied Soft Computing*, vol. 30, pp. 614–627, 2015.

[28] J. Yang, L. Qu, Y. Shen, Y. Shi, S. Cheng, J. Zhao, and X. Shen, "Swarm intelligence in data science: applications, opportunities and challenges," in *International Conference on Swarm Intelligence*. Springer, 2020, pp. 3–14.

[29] J. R. Manne, "Swarm intelligence for multi-objective optimization in engineering design," in *Advanced Methodologies and Technologies in Artificial Intelligence, Computer Simulation, and Human-Computer Interaction*. IGI Global, 2019, pp. 180–194.

[30] T. Nakane, N. Bold, H. Sun, X. Lu, T. Akashi, and C. Zhang, "Application of evolutionary and swarm optimization in computer vision: a literature survey," *IPSJ Transactions on Computer Vision and Applications*, vol. 12, no. 1, p. 3, 2020.

[31] M. Xu, L. Cao, D. Lu, Z. Hu, and Y. Yue, "Application of swarm intelligence optimization algorithms in image processing: A comprehensive review of analysis, synthesis, and optimization," *Biomimetics*, vol. 8, no. 2, p. 235, 2023.

[32] S. Almufti, "Using swarm intelligence for solving nphard problems," *Academic Journal of Nawroz University*, vol. 6, no. 3, pp. 46–50, 2017.

[33] I. Fister Jr, X.-S. Yang, I. Fister, J. Brest, and D. Fister, "A brief review of nature-inspired algorithms for optimization," *ArXiv Preprint ArXiv:1307.4186*, 2013.

[34] S. Deb, X.-Z. Gao, K. Tammi, K. Kalita, and P. Mahanta, "Recent studies on chicken swarm optimization algorithm: a review (2014–2018)," *Artificial Intelligence Review*, vol. 53, pp. 1737–1765, 2020.

[35] X. Huang, C. Ye, and J. Zheng, "Chicken swarm optimization algorithm of hybrid evolutionary searching strategy," *Computer Engineering and Applications*, vol. 54, no. 7, pp. 176–181, 2018.

[36] D. Wu, F. Kong, W. Gao, Y. Shen, and Z. Ji, "Improved chicken swarm optimization," in *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE, 2015, pp. 681–686.

[37] X. Shi and Y. Gao, "Hybrid algorithm based on chicken swarm optimization and artificial bee colony," *Journal of Hefei University of Technology (Natural Science)*, vol. 41, no. 5, pp. 589–594, 2018.

[38] A. Meng, Z. Li, H. Yin, S. Chen, and Z. Guo, "Accelerating particle swarm optimization using crisscross search," *Information Sciences*, vol. 329, pp. 52–72, 2016.

[39] J. Zhang, K. Xia, and Z. Yin, "Quantum chicken swarm optimization with levy flight and its application in parameter optimization of random forest," in *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*. IEEE, 2019, pp. 864–867.

[40] M. Lin, Y. Zhong, J. Lin, and X. Lin, "Enhanced chicken swarm optimisation for function optimisation problem," *International Journal of Wireless and Mobile Computing*, vol. 15, no. 3, pp. 258–269, 2018.

[41] D. Wu, S. Xu, and F. Kong, "Convergence analysis and improvement of the chicken swarm optimization algorithm," *IEEE Access*, vol. 4, pp. 9400–9412, 2016.

[42] M. Z. Afzal, F. Wen, N. Saeed, and M. Aurangzeb, "Enhanced state of charge estimation in electric vehicle batteries using chicken swarm optimization with open ended learning," *Scientific Reports*, vol. 15, no. 1, p. 10833, 2025.

[43] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[44] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2010.

[45] X. Meng, Y. Liu, X. Gao, and H. Zhang, "A new bio-inspired algorithm: chicken swarm optimization," in *Advances in Swarm Intelligence: 5th International Conference, ICSI 2014, Hefei, China, October 17-20, 2014, Proceedings, Part I 5*. Springer, 2014, pp. 86–94.

[46] S. Deb, X.-Z. Gao, K. Tammi, K. Kalita, and P. Mahanta, "Recent studies on chicken swarm optimization algorithm: a review (2014–2018)," *Artificial Intelligence Review*, vol. 53, pp. 1737–1765, 2020.

[47] D. Yazdani, J. Branke, M. N. Omidvar, X. Li, C. Li, M. Mavrovouniotis, T. T. Nguyen, S. Yang, and X. Yao, "Ieee cec 2022 competition on dynamic optimization problems generated by generalized moving peaks benchmark," *arXiv preprint arXiv:2106.06174*, 2021.

[48] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle swarm optimization: A comprehensive survey," *IEEE Access*, vol. 10, pp. 10 031–10 061, 2022.

[49] F. Han, Q. Zhao, Z. Du *et al.*, "Enhanced chicken swarm algorithm for global optimization," *Application Research of Computers*, vol. 36, no. 8, pp. 2317–2319, 2019.

[50] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the sixth international symposium on micro machine and human science*. IEEE, 1995, pp. 39–43.