WEP Warwick Evans Publishing

International Journal of Mechanical and Electrical Engineering

ISSN: 3005-9615 (Print), ISSN: 3005-7132 (Online) | Volume 1, Number 1, Year 2023
DOI: https://doi.org/10.62051/ijmee.v1n1.02
Journal homepage: https://wepub.org/index.php/IJMEE/index

Solution of Inverse Kinematics for Industry Robot Based on Multi-Strategy Improved Chimpanzee Optimization Algorithm

Shuzeng Hou, Chengyuan Luo *, Zhiming Wu, Weifeng Sun and Huikun Zhang

Department of Mechanical Engineering, Sichuan University of Science and Engineering, Yibin 644005, Sichuan, China

*Corresponding Author: Chengyuan Luo (Email: 2630994883@qq.com)

ABSTRACT

During the process of inverse kinematics solving for a robotic arm, heuristic optimization algorithms have been widely applied. However, due to the complex structure of the robotic arm, these algorithms suffer from drawbacks such as slow convergence speed and a tendency to get trapped in local optima. Therefore, a novel metaheuristic optimization algorithm called the Chimpan-zee Optimization Algorithm (ChOA) has been introduced, along with multiple strategy improvements to enhance its global search capability and convergence optimization performance. Through simulation experiments on the modified Chim-panzee Optimization Algorithm (MChOA), it has been demonstrated that this method outperforms the traditional ChOA algo-rithm in terms of convergence speed and accuracy, effectively addressing the issues of slow convergence and being trapped in local op-tima.

KEYWORDS

Industry Robot; Chimpanzee optimization algorithm; Solution of Inverse Kinematics.

1. INTRODUCTION

The industrial robotic arm stands as a pivotal electromechanical integration apparatus within the contemporary manufac-turing sector, having found extensive deployment across diverse industries. In pursuit of investigating matters pertaining to the motion control, trajectory planning, and operational spatial analysis of robotic arms, the initial imperative lies in conducting a kinematic analysis of these mechanical marvels, with the objective of elucidating the correspondence between Cartesian space and joint space. Nevertheless, in the present day, the escalating demand for heightened flexibility in robotic arms has compelled the designs to become increasingly intricate, consequently engendering a concomitant augmentation in the intrica-cy of inverse kinematic analysis. The conundrum of resolving the inverse kinematics for multiple nodes stands as a focal point under earnest deliberation by academic institutions and research consortia alike.

The conventional methods for solving inverse kinematics typically bifurcate into closed-form solutions and numerical ap-proaches. [1,2] Closed-form solutions yield results in analytical expressions, yet they are solely applicable to straightforward or low-degree-of-freedom robotic arms, proving inadequate for highly redundant robotic arms. Redundant robotic arms ne-cessitate recourse to numerical methodologies. Traditional numerical solving techniques primarily encompass the Jacobian matrix iteration method[3] and neural network methodologies[4] , among others. Due to their exceptional capabilities, various intelligent model algorithms have garnered extensive research and application in the realm of robotic arm inverse kinematics resolution. Kieffer[5] has propounded a technique employing neural networks to approximate the inverse kinematics of robot-ic arms. This

approach efficaciously addresses the inverse kinematics conundrum of robotic arms while circumventing the need for the pseudoinverse of the Jacobian matrix, thereby yielding specific solutions and chi-square solutions. Nonetheless, owing to its exigency for copious training data and time, it remains unsuitable for engineering applications. Deng Gangfeng et al.[6] have harnessed an enhanced genetic algorithm in the domain of kinematic inverse solutions, augmented by the incorporation of penalty functions, thereby elevating the practicality of inverse kinematics resolution. Han Xingguo and his col-leagues [7] have introduced a welding robotic arm inverse kinematics resolution approach grounded in Radial Basis Function (RBF) neural networks. This method transforms the inverse kinematics solution of the robotic arm into a predictive system with six inputs and six outputs. In comparison to inverse kinematics resolution methods based on Backpropagation (BP) neu-ral networks, it exhibits reduced coordinate and positional errors. However, it still bears the drawback of necessitating pre-training.

Among the plethora of intelligent algorithms, metaheuristic algorithms stand out as a category inspired by the natural world. They amalgamate the global exploration abilities of biological populations with the notion of utilizing information from pre-viously explored regions. These algorithms influence the intuition of evolving individuals through continuous iterations, guid-ing them towards the most promising segments within the search space, ultimately harnessing them to attain the highest-quality outcomes. [8] Nonetheless, metaheuristic techniques often exhibit some classical shortcomings, such as susceptibility to getting trapped in local optimization zones and encountering convergence issues[9].

The Chimpanzee Optimization Algorithm (ChOA) is a metaheuristic algorithm inspired by the hunting strategies of chim-panzee populations. As elucidated in reference [10] , this algorithm has the capacity to alleviate two prevalent issues in high-dimensional problems: slow convergence and susceptibility to local optima. In comparison to other metaheuristic algorithms, the Chimpanzee Optimization Algorithm boasts high stability, minimal parameter configuration, ease of programming im-plementation, independence from gradient information, and robust optimization performance. It has already been successful-ly applied to solving a diverse range of optimization problems. To date, there is no scholarly work that has employed ChOA for the resolution of robotic arm inverse kinematics.The preparation of manuscripts which are to be reproduced by photo-offset requires special care. Papers submitted in a technically unsuitable form will be returned for retyping, or canceled if the volume cannot otherwise be finished on time.

2. INVERSE KINEMATIC ANALYSIS OF ROBOTIC ARMS

2.1. Establishment of Kinematic Models

This article primarily focuses on the IRB-2600-12/1.65 industrial robotic arm, which falls under the category of a six-degree-of-freedom jointed robotic arm in compliance with the Pieper criteria. Leveraging the robotic arm joint parameters presented in Table 1, the link coordinate systems have been established using the Modified Denavit-Hartenberg (M-DH) convention, as depicted in Fig.1.

Table1. DH Parameter Table

Joint i	θ_i	d_i (mm)	a_{i-1} (mm)	α_{i-1} (°)	θ_i (°)
1	θ_1	445	0	0	-180-180
2	θ_2 +90	0	150	-90	-95-150
3	θ_3	0	-700	0	-180-75
4	θ_4	795	-115	90	-400-400
5	θ_5	0	0	-90	-120-120
6	θ_6	85	0	90	-400-400

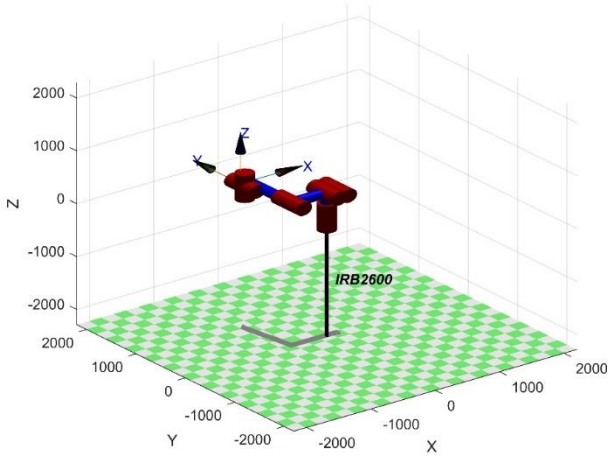

Figure 1 Robotic Arm Model

In accordance with the DH parameter table, the coordinate transformation matrix ,denoted as T_n ,describing the relationship between coordinate system n and coordinate system n-1 can be determined. The general expression for T_n is as follows:

$$. \ T_n = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & \alpha_{i-1} \\ S\theta_i C\alpha_{i-1} & C\theta_i C\alpha_{i-1} & -S\alpha_{i-1} & -S\alpha_{i-1} d_i \\ S\theta S\alpha_{i-1} & C\theta_i S\alpha_{i-1} & C\alpha_{i-1} & C\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Here, $S\theta_i$ represents the sine function of the angle θ_i , and $C\theta_i$ represents the cosine function of the angle θ_i . To derive the coordinate transformation matrix for each link of the robotic arm, the DH parameters are incorporated into the equation. Therefore, the transformation matrix for the end-effector position of the robotic arm can be expressed as:

$$T_{60} = T_1 T_2 T_3 T_4 T_5 T_6 = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{3\times3} & P \\ 0 & 1 \end{bmatrix} . \tag{2}$$

In this expression, the end-effector of the robotic arm is represented by vectors n , o , and a , which respectively denote its normal, sliding, and approach vectors, collectively forming the orientation matrix $R_{3\times3}$. P is the position vector of the target relative to the origin coordinate system.

2.2. Design of the Fitness Function

The inverse kinematics of a robotic arm is a pivotal step in kinematic analysis, typically involving the determination of all joint angles that satisfy the target pose and position. If we express the distance between the target position and the current position as the objective function, then the inverse kinematics problem becomes an optimization task of minimizing the objective function.

In the process of inverse kinematics solving using metaheuristic optimization algorithms, the Euclidean distance is commonly employed as the fitness function. This distance exclusively considers the positional error of the end-effector [1-3]as the fitness function. In equation(3), ΔP represents the absolute difference between the target Cartesian space parameters (x, y, and z) and the current parameters, which are computed using the forward kinematics with the current joint variables.

$$f_1(\theta_i) = \left\| p(\theta_i) - p_{target} \right\| = \left\| \Delta P \right\| \ . \tag{3}$$

$$\Delta P = \sqrt{\Delta P_x{}^2 + \Delta P_y{}^2 + \Delta P_z{}^2} \quad . \tag{4}$$

In the equation, $f(\theta_i)$ represents the fitness value associated with the position, $p(\theta_i)$ denotes the spatial position corresponding to the population individual at θ_i, and p_{target} signifies the ultimate target position.

Another common fitness function used in inverse kinematics based on swarm optimization [4-5] is the Mean Squared Error (MSE). It incorporates both position and orientation, as shown in equation(5) The MSE encompasses the absolute directional error between the expected direction of the end-effector, expressed in angular values, and the current direction.

$$f_2(\theta_i) = \text{MSE} = \sqrt{\Delta P_x^2 + \Delta P_y^2 + \Delta P_z^2 + \Delta R_x^2 + \Delta R_y^2 + \Delta R_z^2} \quad . \tag{5}$$

While MSE provides a more comprehensive consideration of both position and orientation, it lacks normalization in its calculations. Since angular and positional quantities have significantly different orders of magnitude, in the case of simple addition, the contribution of orientation deviation is minimal. Hence, another common approach to generating a fitness function is to assign independent weight coefficients α and β to the positional and directional errors $f_{_P}(\boldsymbol{\theta})$ $f_{_R}(\boldsymbol{\theta})$, respectively, and then combine the two linearly[6-9]. The choice of weight coefficients depends on the specific fitness function constructed as follows:

$$f_3(\theta_i) = \alpha \left\| \Delta \mathbf{p} \right\| + \beta \left\| \Delta \mathbf{R}_{s\times 3} \right\| \quad . \tag{6}$$

Variable bounds can be established in two distinct ways. One approach defines the domain of variables based on the range of joint movements. The other approach, particularly relevant in inverse kinematics solving within the context of trajectory design, often involves a tight connection between inverse kinematics solving and trajectory design. In this scenario, the inverse kinematics solving process frequently has a previously determined solution as a hidden condition, allowing for the adjustment of variable bounds to be suitably narrowed.

In this study, the third approach is employed to construct the fitness function, which is also used as the ultimate objective function. It is evident from the robotic arm kinematic transformation matrix that the orientation matrix is composed of trigonometric functions, implying that the magnitude of $\left\| \Delta \mathbf{R}_{s\times 3} \right\|$ is around 10^0. Conversely, based on the mechanical arm link parameters, the Euclidean norm of the positional error is approximately at a magnitude of 10^3. Therefore, it is necessary to harmonize the convergence speeds of the fitness function subcomponents during the solving process by assigning different values to the weight coefficients. This is done to prevent the influence of certain subcomponents from being neglected due to significant magnitude differences. The actual values are determined based on the specific parameters of the robotic arm.

3. IMPROVEMENTS TO THE CHIMPANZEE OPTIMIZATION ALGORITHM

3.1. The Standard ChOA

ChOA, inspired by the hunting instincts of chimpanzees, is a swarm optimization algorithm that has evolved from this natural behavior [10]. Its population comprises four distinct categories of agents: Drivers, Chasers, Barriers, and Attackers. While each individual chimpanzee within these separate groups possesses independent search capabilities, these differences are essential for successful hunting. Here are some key mathematical descriptions of ChOA.

$$\mathrm{x}_{\text{chimp}}^{(t+1)} = \mathrm{x}_{\text{prey}}^{t} - \mathrm{a} \cdot \left| \mathrm{c} \cdot \mathrm{x}_{\text{prey}}^{t} - \mathrm{m} \cdot \mathrm{x}_{\text{chimp}}^{t} \right| . \tag{7}$$

In which, x_{prey}^{t} represents the location of the prey, x_{chimp}^{t} denotes the position of a specific chimpanzee, t represents the current iteration count, and a m, and c are coefficient vectors calculated using the following formulas.

$$a = 2 \cdot f \cdot rand_1 - f \quad . \tag{8}$$

$$c = 2 \cdot rand_2 \, . \tag{9}$$

$$m = Chaotic\_value \quad . \tag{10}$$

Where rand1 represents a coefficient between 0 and 1, jointly generated by a random number generator and a coefficient Cgi that varies with the increasing iteration count. Cgi has different representations in different groups, as shown in Figure 2 [10], illustrating distinct trends across the four groups. rand2 represents a uniformly distributed random value between 0 and 1, while f is linearly decreasing within the range [2.5, 0] throughout the entire iteration process. The chaos vector denoted as m signifies the impact of behavioral incentives expressed using various chaotic mappings on the agents

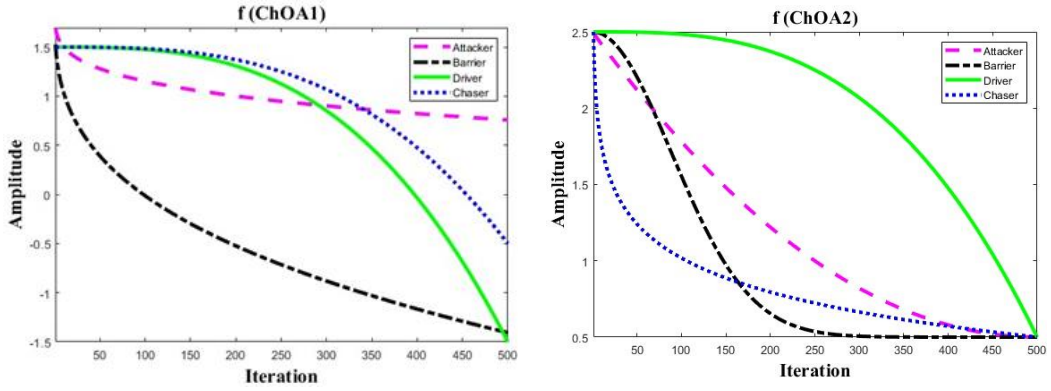

Figure 2 Two Preset Coefficient Variation Trends [10]

In the specific utilization of ChOA, the process begins by creating an initial population with a random distribution. Subsequently, the initial chimpanzee population is categorized into four groups based on their behaviors: Attackers, Drivers, Chasers, or Barriers. All chimpanzee individuals are engaged in estimating the optimal position of the prey, but each group of the population employs different methods to search for the vector value f at their current positions. At the conclusion of the iterations, the inferred position of the prey is represented by the current best individual position.

Chimpanzee individuals collaborate with other chimpanzees to aid in their search, utilizing the best solutions obtained to calculate the optimal prey position, facilitating the analysis and modeling of hunting behavior. The update rules for the positions of the optimal Attackers, Barriers, Drivers, and Chasers are as follows:

$$\begin{aligned}
\mathbf{x}_1 &= \mathbf{x}_{\text{Attacker}}^{t} - \mathbf{a}_1 \left| \mathbf{c}_1 \mathbf{x}_{\text{Attacker}}^{t} - \mathbf{m}_1 \mathbf{x}^{t} \right| \\
\mathbf{x}_2 &= \mathbf{x}_{\text{Barrier}}^{t} - \mathbf{a}_2 \left| \mathbf{c}_2 \mathbf{x}_{\text{Barrier}}^{t} - \mathbf{m}_2 \mathbf{x}^{t} \right| \\
\mathbf{x}_3 &= \mathbf{x}_{\text{Chaser}}^{t} - \mathbf{a}_3 \left| \mathbf{c}_3 \mathbf{x}_{\text{Chaser}}^{t} - \mathbf{m}_3 \mathbf{x}^{t} \right| \\
\mathbf{x}_4 &= \mathbf{x}_{\text{Driver}}^{t} - \mathbf{a}_4 \left| \mathbf{c}_4 \mathbf{x}_{\text{Driver}}^{t} - \mathbf{m}_4 \mathbf{x}^{t} \right| \\
\mathbf{x}^{(t+1)} &= \frac{\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3 + \mathbf{x}_4}{4}
\end{aligned} \tag{11}$$

There is evidence [11] to suggest that chaotic mappings can aid in enhancing the convergence speed and mitigating the problem of local optimization in high-dimensional problems.

Assuming that half of the chimpanzees exhibit systematic behavior during the later stages of the hunt, while the other half employ chaotic strategies to alter their positions, the following formula represents this update method.

$$\mathbf{x}_{\text{chimp}}^{(t+1)} = \begin{cases} \mathbf{x}_{\text{prey}}^{t} - \mathbf{a} \cdot \mathbf{d} & \text{if } \mu < 0.5 \\ \text{Chaoticvalue} & \text{if } \mu \geq 0.5 \end{cases} \tag{12}$$

Where μ is a random number between 0 and 1. Ultimately, this arrangement ensures that a proportion of chimpanzees with random behaviors initiates the ChOA. All individual chimpanzees are assigned to one of the four distinct groups, each of which updates the $\mathbf{f}$ vector using a different method. Then, in each iteration, the potential prey position is collectively assessed by the four groups. Finally, the chaotic mapping helps prevent local minima while expediting convergence.

3.2. Improvements to the Chimpanzee Optimization Algorithm

3.2.1. Initializing the Population with Chaotic Mapping

Using chaotic sequences to initialize the population can enhance population diversity, improve the algorithm's global search capabilities, expedite convergence, and increase robustness.

Chaotic sequences possess high randomness and unpredictability, generating sequences of random numbers distinct from those produced by traditional random number generators, thus augmenting population diversity. Random numbers generated from chaotic sequences exhibit high dispersion and uniformity, enabling genetic algorithms to explore optimal solutions more effectively within the global search space. Chaotic sequences can generate highly randomized initial populations, facilitating faster convergence to optimal solutions. Chaotic sequences are resilient to minor variations in input data, increasing algorithm robustness, and ensuring good results even when input data deviates slightly.

Reference [12] conducted a comparison of various chaotic mappings during the improvement process of ChOA, and the results indicated that the Iteration mapping exhibited the most uniform distribution between 0 and 1. Consequently, this paper also employs the Iteration chaotic mapping to initialize the positions of the chimpanzee population, with its expression as follows:

$$x_{n+1} = \sin(b\pi / x_n) \tag{13}$$

Using the rand function to generate six random numbers, each serving as the initial term of a chaotic sequence, where the length of the chaotic sequence matches the population size. Finally, the initialization of the population is accomplished using the formula (14).

$$Positions(:, i) = chaos \cdot (ub_i - lb_i) + lb_i \tag{14}$$

Where lb_i and ub_i represent the lower and upper bounds of the i-th variable, $chaos$ denotes the corresponding value from the generated chaotic sequence, and $Positions$ represents a parameter variable of an individual in the population.

3.2.2. Optimizing Coefficients Using Opposition-based learning Approach

Tizhoosh[13] introduced Opposition-based Learning (OBL), a method for selecting complementary alternative solutions from a list of potential solutions. OBL aids the search process of the original algorithm by considering the reverse mapping of potential solutions, i.e., comparing other potential solutions that are opposite to the original sample. This approach enhances the coverage, accuracy, and convergence of the search region. [14]

Applying OBL to the initial population will result in a population composed of reverse individuals. Selecting the superior individuals and retaining them can enhance population stability. As shown below, an individual L can be generated from L in a reverse manner:

$$L = Lower_{bound} + Upper_{bound} - L \tag{15}$$

Where $Lower_{bound}$ and $Upper_{bound}$ respectively represent the lower and upper bounds of the solution space $\mathbb{R}$. If $f(L)$ is superior to $f(L)$, then the current chosen solution is L; otherwise, L is selected.

This method generates reverse individuals that are symmetrically generated around the center of the solution space. There are also other methods to generate reverse individuals using different symmetry approaches.

By comparing the standard Grey Wolf Optimizer (GWO) and ChOA, it can be observed that the position update formulas for both algorithms can be represented as:

$$X(t+1) = X_A(t) - \mathrm{a} \cdot d \tag{16}$$

Where X represents the individual's position, X_A represents the position of a selected optimal individual, and d represents a certain distance from the individual to the optimal individual. ChOA improves the expression of the parameter vector a in the agent position update formula of GWO, which can be expressed as follows after simplification:

$$\mathrm{a} = 2rand \cdot \mathrm{f} - \mathrm{f} = (2rand - 1) \cdot \mathrm{f} \tag{17}$$

Where f is a parameter that linearly decreases from 2 to 0 as the number of iterations progresses. In this case, $2rand - 1$ can be regarded as a uniformly distributed random number ranging from -1 to 1. In standard ChOA, the expression of $rand$ has been improved, optimizing its speed at different stages of the iteration process, which has shown good results. However, through simulations, it has been observed that simple changes in the way $rand$ is calculated can lead to the parameter a no longer exhibiting symmetry about 0 concerning f .After a certain number of iterations, the sign of $(2rand - 1)$ may cease to change.

While modifying $rand$, a new parameter b is introduced to ensure that the coefficient of $(2rand - b)$ exhibits symmetry about 0. This can accelerate convergence while enhancing global search capabilities. When the change in parameter a with respect to f exhibits symmetry about 0, as d is, in some sense, the distance perceived by biological individuals, it can be understood that $X(t)$ generates a reverse point $X'(t)$ centered around $X_A(t)$. The original individual $X(t)$ can reach the optimal point $X_A(t)$ by searching along the d direction. With the introduction of the reverse point $X'(t)$, it can reach the optimal point $X_A(t)$ by searching along the $-d$ direction.

The expression for the modified parameter vector a is as follows:

$$\mathrm{a} = 2 \cdot \mathrm{Cg}_i \cdot rand \cdot \mathrm{f} - \mathrm{Cg}_i \mathrm{f} \tag{18}$$

Cg_i is represented by the variation coefficients in Figure 2.

3.2.3. Standardizing Random Number Coefficients

In the standard ChOA, such as when using the position update formula (11), using independent random numbers for different role populations when computing the system vector a can actually weaken the uniformity of the final generated individual positions in the spatial distribution.

For instance, in the optimization scenario shown in Figure 3, the actual location of the prey is not within the region formed by the best four individual positions, namely X1, X2, X3, and X4. If the original formula is used to update individual positions, it is highly unlikely to search for the optimal position.

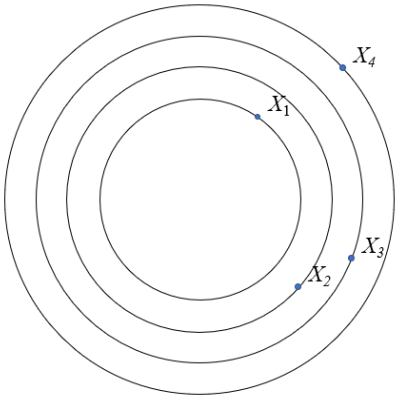

Figure 3. A Possible Optimization Scenario

The analysis suggests that using independently generated different random numbers can lead to a concentration phenomenon similar to multiple mutually independent identically distributed outcomes in the direction of the expected value in a Bernoulli experiment. In the optimization process of ChOA, this manifests as an enrichment of updated positions occurring in the central region of the best four individual positions. This indirectly weakens the global search capability. Therefore, using uniformly generated random numbers when computing the vector "a" in the formula can redistribute the iteratively generated individual positions more evenly around the region collectively formed by the best four individual positions. This can enhance the algorithm's global search capability.

3.2.4. Adaptive Weight Adjustment

In a minimization optimization problem, the smaller the fitness function values in the population individuals, the closer they are likely to be to the global optimum. Specifically, when using optimization algorithms to solve specific problems in real situations, if it's possible to know the exact value of the optimal fitness function in advance, then the credibility of the optimal point can be represented directly using the difference between the fitness value of an individual's position and the optimal fitness value. When using the ChOA position update formula, it can be modified as follows:

$$\mathbf{x} = \frac{\mathbf{x}_1 \cdot \mathrm{w}_1 + \mathbf{x}_2 \cdot \mathrm{w}_2 + \mathbf{x}_3 \cdot \mathrm{w}_3 + \mathbf{x}_4 \cdot \mathrm{w}_4}{(\mathrm{w}_1 + \mathrm{w}_2 + \mathrm{w}_3 + \mathrm{w}_4)} \tag{19}$$

Where w_i represents the weight relative to the confidence in $\mathbf{x}_i$, and it is calculated using the following formula:：

$$\mathrm{w}_i = \frac{|\mathbf{x}_1 - \mathrm{x}_{best}| + |\mathbf{x}_2 - \mathrm{x}_{best}| + |\mathbf{x}_3 - \mathrm{x}_{best}| + |\mathbf{x}_4 - \mathrm{x}_{best}|}{|\mathbf{x}_i - \mathrm{x}_{best}| + |\mathbf{x}_2 - \mathrm{x}_{best}| + |\mathbf{x}_3 - \mathrm{x}_{best}| + |\mathbf{x}_4 - \mathrm{x}_{best}|} \tag{20}$$

Where x_{best} represents the known optimal fitness function value. Changing the weight ratio can effectively improve the convergence speed. When constructing the fitness function, it can be adjusted using a certain method so that $\mathrm{x}_{\mathrm{best}} \geq 0$. In some cases, $\mathrm{x}_{best} = 0$, and thus the weight formula can be adjusted to:

$$\mathrm{w}_i = \frac{P_1 + P_2 + P_3 + P_4}{P_i + P_2 + P_3 + P_4} \tag{21}$$

Where P_i represents the fitness function value of the i-th best individual. Adjusting the weight coefficients can accelerate the convergence speed.

3.3. Algorithm Steps

Synthesizing the aforementioned improvement methods, the implementation steps of the Modified Chimp Optimization Algorithm (MChOA) provided in this paper are as follows:

Step 1: Initialize algorithm-related parameters, including population size N, spatial dimension $\dim$, searchable space $[lb, ub]$, and maximum iteration count $Max\_iteration$;

Step 2: Utilize the Iteration mapping sequence as per formulas (13) and (14) to initialize the population;

Step 3: Compute the fitness values of individuals in the population, select the positions of the four individuals with the lowest fitness values, and record them as $X_{Attacker}$, $X_{Barrier}$, X_{Chaser}, and X_{Driver}, respectively;

Step 4: Update the population positions using formulas (18) and (21);

Step 5: Check if the iteration count satisfies the termination condition for the number of iterations. If it does, output the current global best individual position $X_{Attacker}$; otherwise, return to Step 3 and continue the execution.

4. EXPERIMENTAL SIMULATION

4.1. Parameter Settings

This paper was based on hardware environment with an Intel I7 12700H 2.7GHz processor, 32GB of memory, using Matlab R2022b as the programming software, and the operating system was Windows 11. The population size N was set to 50, the maximum number of iterations was 1000, and the algorithm was run 30 times. The low-dimensional experimental dimension was set to 30, and the high-dimensional experimental dimensions were 500 and 1000.

4.2. Test Functions

Table 2. Specific Parameters for Test Functions

function	dim	Domain	Theoretical Values
$F_1 = \sum_{i=1}^{n} x_i^2$	30/500/1000	[-100,100]	0
$F_2 = \sum_{i=1}^{n} (\lvert x_i \rvert + \prod_{j=1}^{i} \lvert x_j \rvert)$	30/500/1000	[-10,10]	0
$F_3 = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	30/500/1000	[-100,100]	0
$F_4 = Max_i \left\{ \lvert x_j \rvert, 1 \le j \le i \right\}$	30/500/1000	[-100,100]	0
$F_5 = \sum_{i=1}^{n} 100(x_i - x_{i-1}^2)^2 + (x_i - 1)^2$	30/500/1000	[-30,30]	0
$F_6 = \sum_{i=1}^{n} \lvert x + 0.5 \rvert^2$	30/500/1000	[-100,100]	0
$F_7 = \sum_{i=1}^{n} i^2 x_i^4 + random[0,1)$	30/500/1000	[-1.28,1.28]	0
$F_8 = \sum_{i=1}^{n} (-x_i \cdot \sin(\sqrt{\lvert x_i \rvert}))$	30/500/1000	[-500,500]	-419*n
$F_9 = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	30/500/1000	[-5.12,5.12]	0

$F_{10} = -20\exp(-0.2\sqrt{\sum_{i=1}^{n} x_i^2 / n}) - \exp(\sum_{i=1}^{n} \cos(2\pi x_i) / n) + 20e$	30/500/1000	[-32,32]	0
$F_{11} = \sum_{i=1}^{n} x_i^2 / 4000 - \prod_{i=1}^{n} \cos(x_i / \sqrt{i}) + 1$	30/500/1000	[-600,600]	0
$F_{12} = \frac{\pi}{n} \cdot (10\sin^2(\pi(1+\frac{x_1+1}{4}))) + \sum_{i=1}^{n} Ufun(x,10,100,4) + (\frac{x_n+1}{4})^2 + \sum_{i=1}^{n-1} \{(\frac{x_i+1}{4})^2 [1+10\sin^2(\pi(1+\frac{x_{i+1}+1}{4}))]\}$	30/500/1000	[-50,50]	0
$F_{13} = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n} Ufun(x,5,100,4) + \sum_{i=1}^{n} [(x_i-1)^2(1+\sin^2(3\pi x_{i+1})) + (x_n-1)^2(1+\sin^2(2\pi x_n))]\}$	30/500/1000	[-50,50]	0
$F_{14} = (\frac{1}{500} + \sum_{i=1}^{n} \frac{1}{i + \sum_{j=1}^{i} (x_i - m)^6})^{-1}$	2	[-65.536, 65.536]	1
$F_{15} = \sum_{i=1}^{n} [k - x_{i,1}(m^2 + mx_{i,2}) / (m^2 + mx_{i,3} + x_{i,4})]^2$	4	[-5,5]	0
$F_{16} = 4x_1^2 - 2.1x_1^4 + x_1^6 / 3 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[-5,5]	-1.036
$F_{17} = x_2 - (\frac{5.1x_1^2}{4\pi^2} + \frac{5}{\pi x_1} - 6)^2 + 10(1-\frac{1}{8\pi})\cos x_1 + 10$	2	[-5,10] [10,15]	
$F_{18} = 1 + (x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2 + 6x_1x_2+3x_2^2) \cdot (18-32x_1+12x_1^2+48x_2 - 36x_1x_2+27x_2^2) \cdot (30+2x_1-3x_2)^2$	2	[-2,2]	3
$F_{19} = \sum_{i=1}^{4} -c_i \exp(-\sum_{j=1}^{3} a(x_j - p)^2)$	3	[0,1]	N/A
$F_{20} = \sum_{i=1}^{4} -c_i \exp(-\sum_{j=1}^{6} a(x_j - p)^2)$	6	[0,1]	N/A
$F_{21} = \sum_{i=1}^{n} -[(x-a)(x-a)^T + c]^{-1}$	4	[0,10]	N/A
$F_{22} = \sum_{i=1}^{n} -[(x-a)(x-a)^T + c]^{-1}$	4	[0,10]	N/A
$F_{23} = \sum_{i=1}^{n} -[(x-a)(x-a)^T + c]^{-1}$	4	[0,10]	N/A

Where $Ufun$ represents a function:

$$Ufun(x,a,k,m) = \begin{cases} k(x-a)^m & x > a \\ k(-x-a)^m & x < -a \\ 0 & others \end{cases} \quad (22)$$

F_{19} and F_{20} have different values for constants c and p, while F_{21}, F_{22}, and F_{23} have different values for constants a and c.

Among them, F_1 to F_7 are continuous single-peak test functions used to test the algorithm's optimization accuracy. F_8 to F_{13} are continuous multi-peak test functions, and F_{14} to F_{23} are fixed-dimensional multi-peak test functions used to test the algorithm's global search ability and convergence speed.

Adding F_{24} as an inverse kinematics objective function is used to test the algorithm's ability to handle specific problems. The specific setup involves establishing the fitness function in the manner of Equation (6) and using a set of preset numerical values substituted into Equation (2) as the final position's pose matrix.

4.3. The comparative analysis of algorithmic experimental outcomes

To comprehensively validate the efficacy and superiority of the modified MChOA (Modified Chimpanzee Optimization Algorithm), we conducted experimental comparisons on 23 benchmark test functions and inverse kinematics problems. The standard ChOA (Chimpanzee Optimization Algorithm) was compared against the following variants: CChOA (Chaos Chimp Optimization Algorithm) with chaos sequence initialization, OIChOA (OBL Inspiring Chimp Optimization Algorithm) incorporating coefficient modifications based on reverse learning concepts, OChOA (Oneness Chimp Optimization Algorithm) exhibiting consistent random directions, WChOA (Weight Chimp Optimization Algorithm) with weighted coefficient adjustments, and the composite Black Chimp Optimization Algorithm (MChOA), integrating all strategies.

The algorithmic conditions were held constant, with a maximum iteration count of 1000 and a population size of 50. The experiments were conducted iteratively for 30 trials, and the performance of each algorithm was assessed based on four performance metrics: the minimum, maximum, average, and standard deviation of the 30 optimal values collected as an array.

4.3.1. Comparative Analysis of Optimization Enhancements with Distinct Improvement Strategies

Table 3 and Figure 4 depict the specific optimization outcomes resulting from various strategy improvements and the general distribution trends of fitness functions for the test functions. In Figure 4, for functions with more than two input variables, only the transformation trends induced by the first two variables were considered in the plot, while the remaining variables were held constant at zero. The specific functions used were provided by [10] within the MATLAB environment.

Table 3. Comparative Analysis of Optimization Results for Test Functions with Different Improvement Strategies.

Function	Algorithm	Best value	Worst value	Ave	Std
F_1	ChOA	0	1.14E-12	3.78E-14	2.07E-13
	CChOA	0	8.62E-12	2.87E-13	1.57E-12
	OIChOA	0	6.04E-36	2.01E-37	1.1E-36
	OChOA	0	4.46E-13	1.49E-14	8.14E-14
	WChOA	0	2.65E-21	8.83E-23	4.84E-22
	MChOA	0	1.45E-16	4.84E-18	2.65E-17
F_2	ChOA	0	1.86E-08	6.19E-10	3.39E-09
	CChOA	0	1.41E-09	4.71E-11	2.58E-10
	OIChOA	0	8.17E-21	2.72E-22	1.49E-21
	OChOA	0	9.34E-10	3.11E-11	1.7E-10
	WChOA	0	1.50E-11	5.01E-13	2.74E-12
	MChOA	0	6.90E-10	2.30E-11	1.26E-10
F_3	ChOA	0	4.867627	0.162254	0.888703
	CChOA	0	6.443852	0.214795	1.176481

Function	Algorithm	Best value	Worst value	Ave	Std
	OIChOA	0	8.63E-08	2.88E-09	1.58E-08
	OChOA	0	3.87335	0.129112	0.707174
	WChOA	0	0.024684	0.000823	0.004507
	MChOA	0	1.201586	0.040053	0.219379
F_4	ChOA	0	0.000509	1.7E-05	9.3E-05
	CChOA	0	0.054921	0.001831	0.010027
	OIChOA	0	4.95E-10	1.65E-11	9.04E-11
	OChOA	0	0.000512	1.71E-05	9.35E-05
	WChOA	0	1.12E-06	3.74E-08	2.05E-07
	MChOA	0	0.00079	2.63E-05	0.000144
F_5	ChOA	0	28.97016	0.965672	5.289204
	CChOA	0	28.94123	0.964708	5.283922
	OIChOA	0	27.14082	0.904694	4.955213
	OChOA	0	28.87767	0.962589	5.272317
	WChOA	0	28.93802	0.964601	5.283336
	MChOA	0	25.53542	0.851181	4.662108
F_6	ChOA	0	2.527468	0.084249	0.46145
	CChOA	0	2.748515	0.091617	0.501808
	OIChOA	0	1.00442	0.033481	0.183381
	OChOA	0	2.511073	0.083702	0.458457
	WChOA	0	2.108716	0.070291	0.384997
	MChOA	0	2.25E-06	7.50E-08	4.11E-07
F_7	ChOA	0	0.000142	4.72E-06	2.59E-05
	CChOA	0	0.000876	2.92E-05	0.00016
	OIChOA	0	0.001888	6.29E-05	0.000345
	OChOA	0	0.001913	6.38E-05	0.000349
	WChOA	0	6.19E-05	2.06E-06	1.13E-05
	MChOA	0	0.001651	5.50E-05	0.000302
F_8	ChOA	-5642.784	0	-188.093	1030.227
	CChOA	-5609.06	0	-186.969	1024.07
	OIChOA	-3404.126	0	-113.471	621.5056
	OChOA	-5460.912	0	-182.03	997.0215
	WChOA	-5772.4	0	-192.413	1053.891
	MChOA	-8274.41	0	-275.814	1510.694
F_9	ChOA	0	13.6424	0.454747	2.49075
	CChOA	0	8.781849	0.292728	1.603339
	OIChOA	0	26.99093	0.899698	4.927846
	OChOA	0	1.53E-11	5.1E-13	2.79E-12
	WChOA	0	8.87E-12	2.96E-13	1.62E-12
	MChOA	0	1.33E-07	4.44E-09	2.43E-08
F_{10}	ChOA	0	19.96098	0.665366	3.64436
	CChOA	0	19.96383	0.665461	3.64488
	OIChOA	0	1.47E-14	4.88E-16	2.68E-15
	OChOA	0	19.98092	0.666031	3.648
	WChOA	0	19.96329	0.665443	3.644782
	MChOA	0	4.84E-07	1.61E-08	8.84E-08
	ChOA	0	0.021972	0.000732	0.004012

Function	Algorithm	Best value	Worst value	Ave	Std
F_{11}	CChOA	0	0.017724	0.000591	0.003236
	OIChOA	0	0	0	0
	OChOA	0	1.23E-12	4.08E-14	2.24E-13
	WChOA	0	0.017882	0.000596	0.003265
	MChOA	0	3.87E-14	1.29E-15	7.07E-15
F_{12}	ChOA	0	0.694738	0.023158	0.126841
	CChOA	0	0.195426	0.006514	0.03568
	OIChOA	0	0.104896	0.003497	0.019151
	OChOA	0	0.184525	0.006151	0.03369
	WChOA	0	0.119645	0.003988	0.021844
	MChOA	0	1.13E-05	3.78E-07	2.07E-06
F_{13}	ChOA	0	2.82643	0.094214	0.516033
	CChOA	0	2.900033	0.096668	0.529471
	OIChOA	0	0.760831	0.025361	0.138908
	OChOA	0	2.438465	0.081282	0.445201
	WChOA	0	2.988642	0.099621	0.545649
	MChOA	0	0.404427	0.013481	0.073838
F_{14}	ChOA	0	0.998078	0.033269	0.182223
	CChOA	0	0.998163	0.033272	0.182239
	OIChOA	0	0.998004	0.033267	0.18221
	OChOA	0	0.998004	0.033267	0.18221
	WChOA	0	0.998004	0.033267	0.18221
	MChOA	0	0.998004	0.033267	0.18221
F_{15}	ChOA	0	0.001284	4.28E-05	0.000234
	CChOA	0	0.001256	4.19E-05	0.000229
	OIChOA	0	0.000706	2.35E-05	0.000129
	OChOA	0	0.00129	4.3E-05	0.000235
	WChOA	0	0.001275	4.25E-05	0.000233
	MChOA	0	0.001223	4.08E-05	0.000223
F_{16}	ChOA	-1.031627	0	-0.03439	0.188349
	CChOA	-1.0316	0	-0.03439	0.188344
	OIChOA	-1.031628	0	-0.03439	0.188349
	OChOA	-1.031604	0	-0.03439	0.188344
	WChOA	-1.03162	0	-0.03439	0.188346
	MChOA	-1.03163	0	-0.03439	0.188349
F_{17}	ChOA	0	0.39905	0.013302	0.072856
	CChOA	0	0.397902	0.013263	0.072647
	OIChOA	0	0.397887	0.013263	0.072644
	OChOA	0	0.398096	0.01327	0.072682
	WChOA	0	0.398668	0.013289	0.072786
	MChOA	0	0.397887	0.013263	0.072644
F_{18}	ChOA	0	3.000077	0.100003	0.547737
	CChOA	0	3.000171	0.100006	0.547754
	OIChOA	0	3.000001	0.1	0.547723
	OChOA	0	3.000021	0.100001	0.547726
	WChOA	0	3.00003	0.100001	0.547728
	MChOA	0	3	0.1	0.547723

Function	Algorithm	Best value	Worst value	Ave	Std
F_{19}	ChOA	-3.855408	0	-0.12851	0.703898
	CChOA	-3.85406	0	-0.12847	0.703652
	OIChOA	-3.855412	0	-0.12851	0.703899
	OChOA	-3.854758	0	-0.12849	0.703779
	WChOA	-3.85557	0	-0.12852	0.703927
	MChOA	-3.86278	0	-0.12876	0.705244
F_{20}	ChOA	-3.051353	0	-0.10171	0.557098
	CChOA	-3.1683	0	-0.10561	0.578449
	OIChOA	-3.133432	0	-0.10445	0.572084
	OChOA	-3.289111	0	-0.10964	0.600507
	WChOA	-3.01241	0	-0.10041	0.549989
	MChOA	-3.322	0	-0.11073	0.606511
F_{21}	ChOA	-4.964642	0	-0.16549	0.906416
	CChOA	-5.04807	0	-0.16827	0.921647
	OIChOA	-10.15078	0	-0.33836	1.85327
	OChOA	-4.859771	0	-0.16199	0.887269
	WChOA	-4.97573	0	-0.16586	0.908439
	MChOA	-10.1532	0	-0.33844	1.853712
F_{22}	ChOA	-4.982012	0	-0.16607	0.909587
	CChOA	-0.9114	0	-0.03038	0.166399
	OIChOA	-10.39909	0	-0.34664	1.898605
	OChOA	-5.069655	0	-0.16899	0.925588
	WChOA	-5.05295	0	-0.16843	0.922539
	MChOA	-10.4029	0	-0.34676	1.899308
F_{23}	ChOA	-5.095689	0	-0.16986	0.930341
	CChOA	-0.94621	0	-0.03154	0.172754
	OIChOA	-10.53588	0	-0.3512	1.92358
	OChOA	-5.01289	0	-0.1671	0.915224
	WChOA	-5.06288	0	-0.16876	0.92435
	MChOA	-10.5364	0	-0.35121	1.923676
F_{24}	ChOA	9.882795	125.3056	52.71412	25.74718
	CChOA	13.16743	150.035	67.46652	40.84871
	OIChOA	0.246745	5.846956	2.507314	1.437395
	OChOA	18.47393	146.7699	59.87447	34.31539
	WChOA	13.12069	160.5925	71.04506	36.44165
	MChOA	0.536507	5.914908	3.539791	1.534913

The optimal values, worst values, mean values, and standard deviations obtained through the test functions can reflect the algorithm's optimization performance and convergence accuracy. Firstly, when solving F1 to F4, OIChOA consistently outperforms standard ChOA in all four evaluation criteria, achieving the theoretical optimum values of the functions and demonstrating superior optimization accuracy and stability.

F5 is a valley-shaped test function with a global minimum located at the bottom of the valley. In the vicinity, there exists an area with relatively little change in fitness function values, making it prone to slow optimization or stagnation. MChOA, with multiple strategy improvements, excels in this scenario.

F6, F7 and F9 exhibit shapes resembling parabolic surfaces and feature numerous local optima. WChOA, through improved weight coefficients, achieves better results by accurately identifying the optimization direction. MChOA also significantly surpasses these functions, particularly in the case of F6.

F8 represents a test function with a random distribution of numerous local optima. MChOA exhibits the highest optimization accuracy in terms of optimal, worst, and mean values, even though its standard deviation is not ideal. This discrepancy can be attributed to MChOA's superior global optimization capability, as it is capable of discovering exceptionally superior optimal solutions, thereby widening the standard deviation. However, its global optimization ability follows a probabilistic and less stable pattern, dependent on population size and iteration count.

The above results demonstrate that different improvement strategies lead to varying degrees of enhancement in algorithmic optimization performance. In low-dimensional multi-modal functions (F7 to F13), OIChOA consistently performs exceptionally well, especially in the case of F11, where it consistently achieves the theoretical optimum value. In the more complex optimization scenarios of F12 and F13, MChOA shines.

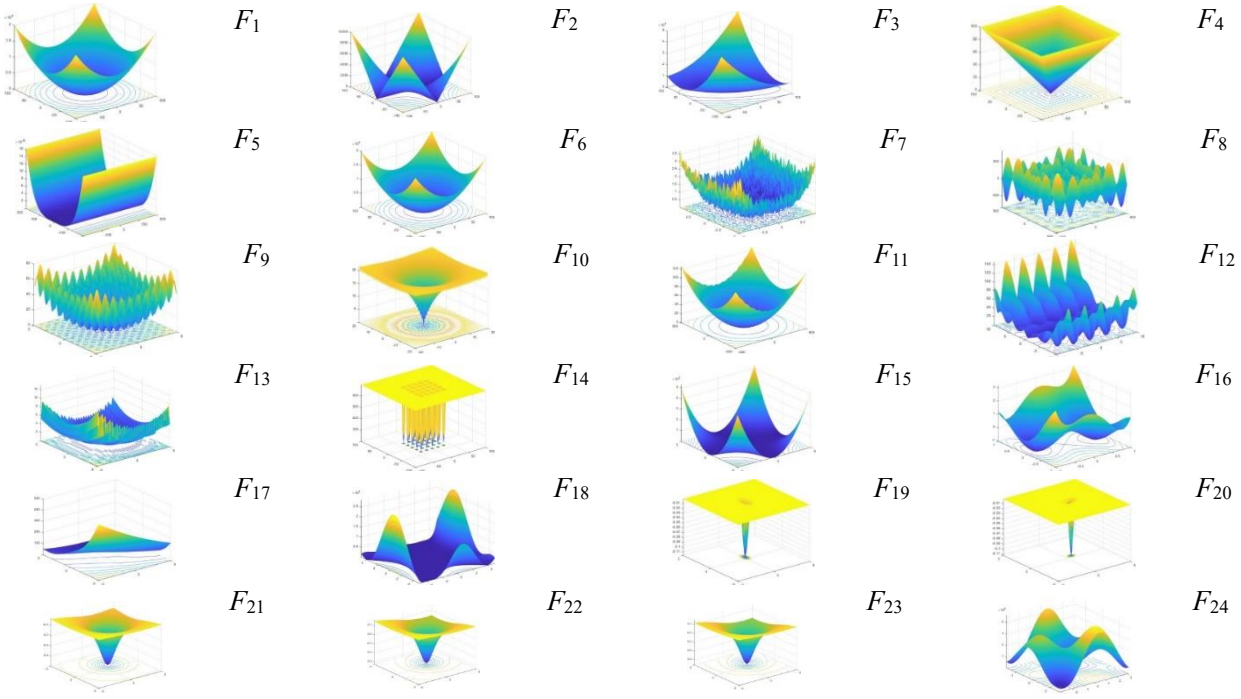

Figure 4. Trends in Test Function and Inverse Kinematics Fitness Functions

In fixed-dimensional multi-modal functions (F14 to F23), MChOA consistently maintains a leading position. In summary, the CChOA, improved only with a chaotic sequence initialization strategy, exhibits average performance, mainly due to the limited number of individuals in the initial population selected during testing. In cases where the initial population generated by the chaotic strategy is sparse and the characteristics of the test region are widely distributed, it does not distinguish itself significantly compared to populations generated randomly.

OIChOA, on the other hand, consistently achieves near-optimal values in the majority of test functions. Its coefficient improvement, reintroducing reverse learning concepts, simultaneously enhances the algorithm's optimization accuracy and convergence speed. OIChOA, with its standardized random coefficients, indeed increases the algorithm's global optimization capability and randomness but also introduces instability.

WChOA effectively accelerates optimization speed by adjusting weight coefficients but has a limited impact on global optimization capability. In conclusion, the improvement strategies proposed in this study have a noticeable positive impact on standard ChOA, and the integrated MChOA, combining

four improvement strategies, exhibits advantages in terms of optimization accuracy, convergence speed, global search capability, and stability.

4.3.2. Comparative Analysis of Test Functions at Different Dimensions

For OIChOA and MChOA, which exhibited more prominent performance in previous evaluations, a comparison of their improvement effects was conducted using higher-dimensional test functions F1 to F13. The results are presented in Table 4 below.

Table 4. Comparative Analysis of the Effects of Different Improvement Strategies on Test Functions at Various Dimensions

Function	Algorithm	dim=500		dim=1000	
		Ave	Std	Ave	Std
F_1	ChOA	0.024050	0.1317295	0.6549245	3.5871694
	OIChOA	2.58E-16	1.41E-15	8.20E-12	4.49E-11
	MChOA	2.97E-09	1.63E-08	1.77E-06	9.72E-06
F_2	ChOA	0.001526	0.0083558	0.0252429	0.1382608
	OIChOA	6.18E-11	3.38E-10	5.46E-07	2.99E-06
	MChOA	5.15E-07	2.82E-06	*	*
F_3	ChOA	35354.83	193646.378	2726256.3	14932321
	OIChOA	1160.177	6354.55113	14419.237	78977.414
	MChOA	5187.576	28413.5263	12411.53	67980.73
F_4	ChOA	3.2738576	17.9316568	3.2976802	18.062138
	OIChOA	2.1867935	11.9775612	2.7641061	15.139633
	MChOA	2.000253	10.9558365	2.339106	12.81181
F_5	ChOA	20.291295	111.140002	123.78393	677.99249
	OIChOA	16.606196	90.9558841	33.26392	182.19399
	MChOA	16.58832	90.8579641	33.25571	182.149
F_6	ChOA	3.9662225	21.7238952	10.252082	56.152966
	OIChOA	3.4300686	18.7872595	7.4452911	40.779539
	MChOA	3.385168	18.5413289	7.222899	39.56145
F_7	ChOA	0.0012203	0.00668386	0.0144182	0.0789718
	OIChOA	0.0001276	0.00069883	0.0003562	0.0019509
	MChOA	0.000117	0.0006428	0.000286	0.001566
F_8	ChOA	-2852.91	15626.0385	-5446.948	29834.165
	OIChOA	-458.8776	2513.37614	-682.97939	3740.832
	MChOA	-672.248	3682.05242	-1054.5842	5776.1954
F_9	ChOA	0.9879299	5.41111509	2.4385032	13.356232
	OIChOA	4.24E-13	2.32E-12	5.34E-12	2.92E-11
	MChOA	1.39E-08	7.62E-08	1.08E-05	5.89E-05
F_{10}	ChOA	0.6692698	3.66574175	0.6699706	3.66958
	OIChOA	9.35E-11	5.12E-10	8.47E-09	4.64E-08
	MChOA	0.689648	3.7773598	0.6910202	3.7848734
F_{11}	ChOA	0.0025382	0.01390209	0.033427	0.1830873
	OIChOA	6.29E-17	3.45E-16	6.55E-13	3.59E-12
	MChOA	1.51E-10	8.29E-10	5.79E-08	3.17E-07
F_{12}	ChOA	0.0372097	0.20380586	0.0383294	0.209939
	OIChOA	0.0292241	0.16006698	0.0326019	0.178568
	MChOA	0.026374	0.14445364	0.029169	0.159765
F_{13}	ChOA	1.6541416	9.06010669	3.490482	19.118157
	OIChOA	1.5791169	8.64917945	3.2222336	17.6489
	MChOA	1.550627	8.4931347	3.203639	17.54705

*Note: When using MATLAB for computation, encountering results displayed as "inf" may indicate that, during position updates using Formula (21), the denominator has reached zero. This suggests that the global optimal position has likely been found in practice, and this inference is supported by the numerical values returned in the workspace when running the function.

It is evident that when addressing high-dimensional function problems, OIChOA and MChOA continue to exhibit significant improvements compared to the standard ChOA. Moreover, MChOA, employing multiple improvement strategies, demonstrates improved optimization speed and global search capability with increasing dimensions.

4.3.3. Comparative Analysis of Optimization Trends with Different Improvement Strategies

To reflect the dynamic convergence characteristics of MChOA, Figure 5 presents the average convergence curves for the 23 benchmark test functions and inverse kinematics functions. Comparatively, MChOA, optimized with multiple strategies, exhibits excellent performance in various scenarios, demonstrating persistent optimization capability, particularly in functions such as [mention function name] where it excels.

During times when MChOA performs exceptionally well, it can be observed that the optimization strategy associated with OIChOA plays a significant positive role in the final optimization results. This optimization strategy enhances the overall range of the search for the optimal individual in MChOA, mitigating the negative impact of local optima. It increases an individual's search capability in the surrounding environment when updating positions, reducing the likelihood of getting trapped in local optima.

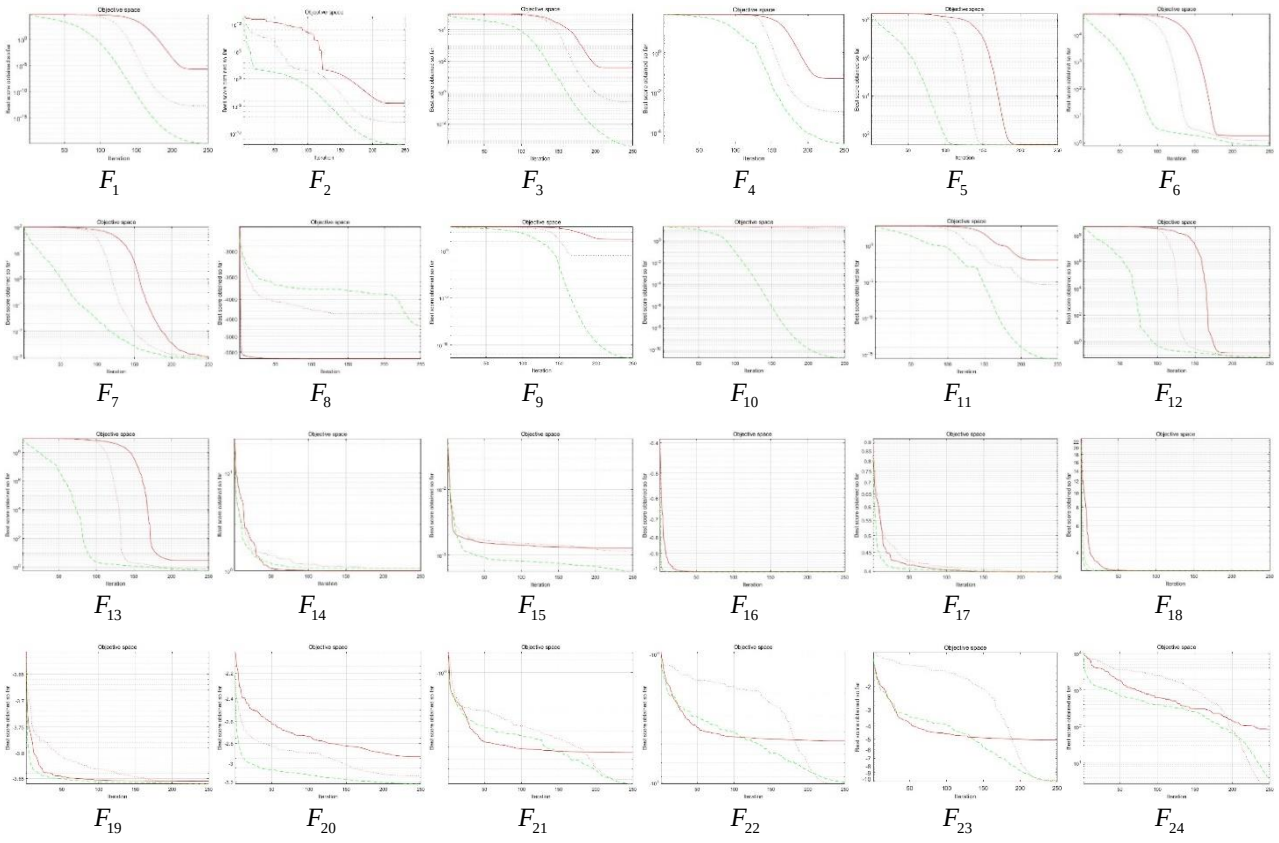

Figure 5. Comparative Convergence Curves of Different Improvement Strategies for Test Functions

5. CONCLUSION

Through a comparative analysis of multiple algorithms, it is evident that MChOA, derived from standard ChOA through the incorporation of multiple improvement strategies, has made breakthroughs in both convergence speed and global search capabilities. It has also demonstrated exceptional performance in solving inverse kinematics problems for robotic arms.

The utilization of the Infinite Folding Iteration Chaos Mapping (Iteration Mapping) for population initialization, coupled with coefficient optimization based on reverse learning concepts, has not only enhanced the optimization precision of the original algorithm but has also significantly improved convergence speed. This approach offers new avenues for enhancing heuristic optimization algorithms further.

The efficacy of the MChOA improvement strategy has been demonstrated through experiments on 13 complex test functions, both in low and high dimensions, as well as comparisons with various algorithms on 10 fixed-dimensional multi-modal complex functions. However, there is still room for further improvement in convergence accuracy when MChOA encounters function F8.

Furthermore, applying MChOA to solve complex problems such as inverse kinematics has yielded favorable results, highlighting its practical applicability in engineering scenarios.

REFERENCES

[1] Serkan Dereli ,Raşit Köker. A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: quantum behaved particle swarm algorithm. Artificial Intelligence Review, 2020, 53(2): 949-964.

[2] Serkan Dereli, Raşit Köker,İsmail Öylek, et al. A comprehensive research on the use of swarm algorithms in the inverse kinematics solution. Politeknik Dergisi, 2019, 22(1): 75-79.

[3] Serkan Dereli,Raşit Köker. IW-PSO approach to the inverse kinematics problem solution of a 7-DOF serial robot manipulator. Sigma Journal of Engineering and Natural Sciences, 2018, 36(1): 77-85.

[4] Minh Tuan Nguyen,Cadmus Yuan,Jin Huang Huang. Kinematic analysis of A 6-DOF robotic arm//Advances in Mechanism and Machine Science: Proceedings of the 15th Iftomm World Congress on Mechanism and Machine Science 15: Springer, 2019: 2965-2974.

[5] Malek Alkayyali,Tarek A Tutunji. PSO-based algorithm for inverse kinematics solution of robotic arm manipulators//2019 20th International Conference on Research and Education in Mechatronics (rem): Ieee, 2019: 1-6.

[6] C.-H.Liu, G.-R.Wang, Y.-L.Chin, et al. Adaptive Particle Swarm Optimization Algorithm for Solving Inverse Kinematics of Mobile Welding Robot . Modular Machine Tool & Automatic Manufacturing Technique, 2023, (3): 50-53, 58

[7] L.-X.Sun, Q.-L.Keng, J.-H.t'ang, et al. Research on Inverse Kinematics of Redundant Manipulatorwith Joint Lmitation . Modern Manufacturing Engineering, 2022, (8): 46-52

[8] G.-S.Meng, D.-D.Kao. Improved Butterfly Optimization Algorithm and Its Application in Solving Inverse Kinematicsproblem of Redundant Manipulators. Manufacturing Technology & Machine Tool, 2022, (8): 91-96

[9] S. Wang, Z.-X. Chang, P.-Z. Chao, et al. Improved Fruit Fly Optimization Algorithm for AnalyzingInverse Kinematics of Redundant Manipulator. Machine Design and Research, 2022, 38(5): 47-53

[10] M. Khishe, M.R. Mosavi. Chimp optimization algorithm. Expert Systems with Applications, 2020, 149: 113338.

[11] Shangce Gao,Yang Yu,Yirui Wang, et al. Chaotic local search-based differential evolution algorithms for optimization. Ieee Transactions on Systems, Man, and Cybernetics: Systems, 2019, 51(6): 3954-3967.

[12] Q.Huang,S.Liu,M.-M.Li, et al. Multi-Strategy Chimp Optimization Algorithm and Its Application of Engineering Problem . Computer Engineering and Applications, 2022, 58(19): 174-183

[13] Hamid R Tizhoosh. Opposition-based learning: a new scheme for machine intelligence//International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (cimca-iawtic'06): Ieee, 2005: 695-701.

[14] Tizhoosh H.-R, Ventresca Mario, Rahnamayan Shahryar. Oppositional Concepts in Computational Intelligence: Springer Berlin Heidelberg, 2008: 11-28.