Robotica (2025), **43**, pp. 3488–3518
doi:10.1017/S0263574725102208

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Improved dung beetle optimization algorithm based inverse kinematics solution for robotic arm

Yunpeng Lv[1], Hongbing Li[2,3], Siqi Zhu[2,3], Siyun Tan[2,3], Peng Yang[1] and Chunzhe Zhao[2,3]

[1]Chongqing Key Laboratory of Geological Environment Monitoring and Disaster Early Warning in the Three Gorges Reservoir Area, Chongqing Three Gorges University, Chongqing, China
[2]Municipal Key Laboratory of Intelligent Information Processing and Control, Chongqing Three Gorges University, Chongqing, China
[3]Chongqing Engineering Research Center for Internet of Things and Intelligent Control Technology, Chongqing Three Gorges University, Chongqing, China
Corresponding author: Hongbing Li; Email: cqlhb@qq.com

Received: 23 April 2025; **Revised:** 30 June 2025; **Accepted:** 28 July 2025; **First published online:** 30 September 2025

Keywords: dung beetle optimization algorithm; directional perturbation; inverse kinematics solution; Robotic Hands; Serial Manipulator Design and Kinematics

Abstract

Aiming at the issues of more difficult to solve and lower precision of six-axis robotic arm in inverse kinematics (IK) solution, a multi-strategy improved dung beetle optimization algorithm (ECDBO) is proposed. It improves performance in four aspects: population initiation, global search capability, search direction perturbation and jumping out of local optima. Sobol sequence strategy was introduced to initialize the dung beetle population, resulting in a more even distribution of individual dung beetles and increasing the diversity of initial population. Boundary optimization strategy is adopted to balance the requirements on search capability at different times. This approach enhances global search capability at the beginning and local search capability at the end of an iteration. Propose hybrid directional perturbation strategy to change the search direction of rolling dung beetles and stealing dung beetles. It allows for more detailed exploration and improves convergence accuracy. The Levy flight strategy is incorporated to perturb current optimal solution, enhancing algorithm's ability to jump out of the local optimum. In order to verify performance of ECDBO algorithm, CEC2017 function tests and robotic arm IK solving experiments were conducted and compared with other algorithms. ECDBO ranked first on 21 functions in the 30 dimensions tested in CEC2017 and on 27 functions in the 100 dimensions. ECDBO performs well in the IK solving experiments of two robotic arms with better accuracy than other algorithms. The experimental results show that the ECDBO algorithm significantly improves the convergence and accuracy, and also performs excellently on the IK solving problem.

1. Introduction

Solving the inverse kinematics (IK) problem is a fundamental task in the field of robotics and automation, as it directly determines the joint configurations required to achieve a desired end-effector pose. IK refers to determining the joint angles and postures of a robotic arm based on the given position coordinates of its end-effector. This serves as the foundation for trajectory planning and control in robotics [1].

Although the IK problem has long been a fundamental aspect of robot control, the growing complexity and diversity of robotic applications in modern industrial environments have brought new challenges to IK solution methods, prompting continued research into more robust and adaptable algorithms. From traditional analytical and numerical methods [2] to modern machine learning and intelligent algorithms, each approach has its own strengths and limitations. Analytical IK methods involve solving complex kinematic equations derived from the robot's geometry, which can be mathematically intensive and are typically only applicable to robots with specific structural characteristics [3]. In contrast to analytical

© The Author(s), 2025. Published by Cambridge University Press.

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

approaches, optimization-based numerical methods solve the IK problem by formulating it as a nonlinear optimization task and iteratively approximating the joint solutions [4].

In recent years, many scholars have worked on solving the IK of robotic arms using different methods. The kinematic relationships of a robotic arm are usually nonlinear, machine learning is able to effectively capture and model these complex nonlinear relationships to improve solution accuracy [5,6]. However, IK solution inaccuracies can occur due to nonlinear mapping complexities, model approximation errors, and dataset mapping errors in machine learning methods [7]. By extending the two-by-two method and comparing it with multiple meta-heuristics, Alireza Motahari verifies the advantages of the new method in terms of computation time and error control, as well as reveals the effect of the number of pending modules on the performance, which provides an efficient and scalable solution strategy for IK solving of discrete hyper-redundant robotic arms [8]. Dongsheng Guo et al. discretize an existing Jacobi pseudo-inverse-based method and adopt a novel five-point discretization formulation to improve the numerical computation accuracy. The new method is theoretically shown to be able to compute IK solutions with high accuracy and good convergence performance [9]. Intelligent optimization algorithms have also made significant progress in solving the IK of robotic arms. Many scholars have applied intelligent algorithms to inverse kinematics solving [10–12]. Compared to machine learning, optimization algorithms do not rely on precise mathematical models and are easy to implement and adapt. Despite the rapid advancements in IK, increasingly complex tasks and demanding work environments impose higher requirements for both solution accuracy and speed. Achieving high-precision and fast solutions remains one of the key challenges in this field today.

Swarm intelligent optimization algorithms have been a hot research topic in recent years, especially in solving engineering problems. Xiao Y et al. proposed a Multi-strategy Boosted Snow Ablation Optimizer, which incorporates high-quality population, greedy selection strategy, differential evolutionary mechanism and dynamic lenticular inverse learning on the basis of the original algorithm, which effectively improves the algorithm's global exploration ability and local exploitation ability [13]. To cope with more complex engineering problems, Xiao Y et al. proposed an Artificial Lemming Algorithm based on the migration, burrow digging, foraging and escaping behaviors of lemmings, where long-distance migration and burrow digging are used for global exploration, and foraging and escaping from predators are used for local exploitation, and an energy decay mechanism is introduced to dynamically adjust the balance between exploration and exploitation, which improves the algorithm's ability to avoid local optimums and quickly converge to the global optimal solution [14]. Rui Zhong et al. proposed an enhanced version of the RIME algorithm called SRIME, which significantly improves the performance of the algorithm in solving complex optimization problems by combining Latin Hypercubic Sampling, an improved Hard Freezing Search Strategy, and an embedded Distance-Based Selection Mechanism, and demonstrates excellent competitiveness in benchmarking and real engineering optimization problems [15]. Shortly after, Rui Zhong et al. proposed a novel meta-heuristic algorithm called Hybrid Remora Crayfish Optimization Algorithm (HRCOA), which effectively solves the original COA algorithm's problems of uneven exploration and exploitation, easy to premature and stagnant problems [16]. Dung Beetle Optimization (DBO) algorithm [17], introduced by Xue J et al. in 2022, is a novel swarm intelligence optimization algorithm with strong optimization and convergence capabilities, showing effective results in solving complex problems. However, according to the "No Free Lunch" theorem [18], no single swarm intelligence algorithm can efficiently solve all optimization problems. The DBO algorithm also suffers from the same problem of imbalance between global and local exploitation capabilities and the tendency to fall into local optima. To address the limitations of the original DBO algorithm, many researchers have proposed improvements [19–22]. Although the performance of the DBO algorithm is improved, it still suffers from poor convergence accuracy and slow convergence speed.

To improve the optimization capability of DBO algorithm and solution accuracy of the IK solution problem, improved DBO algorithm is proposed. Main contributions of the work are as follows.

- Introducing Sobol sequence initialization and Levy flight strategies. Making Dung Beetle Populations More Evenly Distributed, Improving Population Diversity and the Ability to Jump Out of Local Optimality.

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

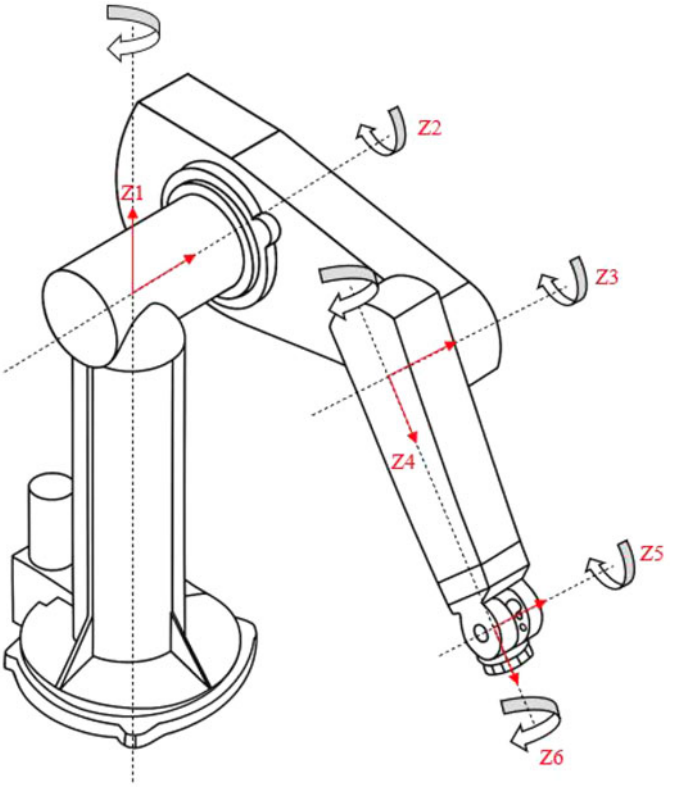


Figure 1. *Link structure diagram of PUMA560 robotic arm.*

- Boundary optimization strategies are proposed for balancing the search capability requirements at different times. Having a larger search area upfront of the algorithm improves the algorithm's global search capability. In the later stages of the algorithm, dung beetles are clustered to search in smaller areas to improve the local exploitation of the algorithm.
- Hybrid directional perturbation strategy is proposed to improve the convergence speed and accuracy of the algorithms by alternating dynamic weight perturbation and T-distribution perturbation to change the search paths of the rolling dung beetle and stealing dung beetle.

This work is organized as follows. Section 2 conducts a kinematic analysis of the PUMA560 six-degree-of-freedom robotic arm and constructs the fitness function for solving IK. Section 3 introduces the original DBO algorithm. Section 4 provides a detailed explanation of the improved ECDBO algorithm. Section 5 tests the ECDBO algorithm for benchmark function comparison and IK solving examples, and makes a comparative analysis with other algorithms. Section 6 concludes.

2. Related work

2.1. Kinematic analysis of robotic arm

This section mainly focuses on the kinematic analysis of the PUMA560 and xArm6 six-degree-of-freedom robotic arms. Figure 1 illustrates the link structure of the PUMA560 robotic arm.

The Denavit-Hartenberg (D-H) parameters of the PUMA560 robotic arm are constructed based on its kinematic parameters, as shown in Table I.

Figure 2 shows the xArm6 robotic arm. Like the PUMA560 robotic arm, this arm is also a 6-axis robotic arm, with three joints used for position adjustment and three joints used for posture adjustment.

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

Table I. *DH parameters with symbolic variables.*

Link	α_i **(rad)**	d_i **(m)**	a_i **(m)**	θ_i **(rad)**
1	0	0	a_2	θ_2
2	0	0	a_2	θ_2
3	$-\pi/2$	d_3	a_3	θ_3
4	$\pi/2$	d_4	0	θ_4
5	$-\pi/2$	0	0	θ_5
6	0	0	0	θ_6

The key parameters are as follows: $d_3 = 0.149$, $d_4 = 0.433$, $a_2 = 0.4318$, $a_3 = 0.0203$.

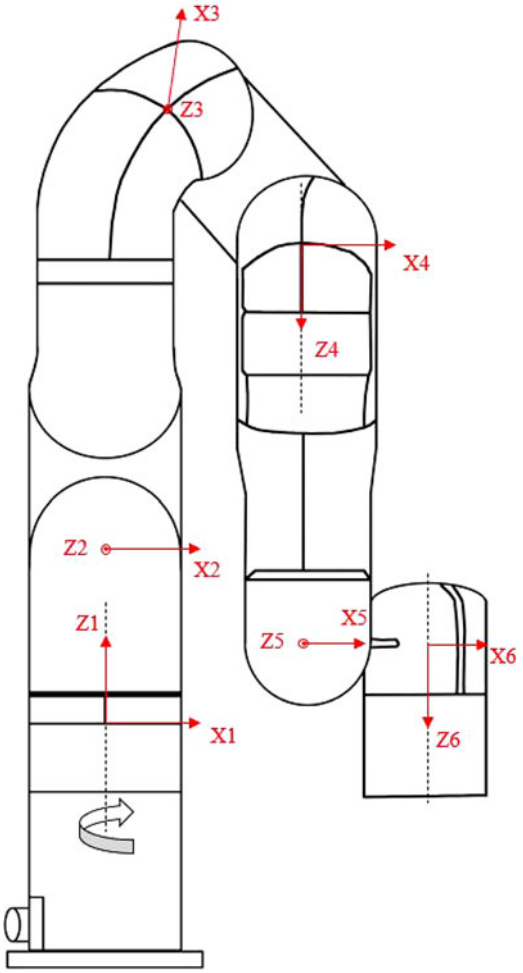


Figure 2. *Link structure diagram of xArm6 robotic arm.*

The Denavit-Hartenberg (D-H) parameters of the xArm6 robotic arm are constructed based on its kinematic parameters, as shown in Table II.

Since both the PUMA560 and xArm6 robotic arms are six-degree-of-freedom robotic arms, their link transformation formulas are the same, as given by Equation (1).

$$ {}^{i-1}_{i}T = \begin{pmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1} $$

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

Table II. *DH parameters of the xArm robot.*

Link	θ_i (rad)	d_i (mm)	α_i (rad)	a_i (mm)	Offset
1	0	267	$-\pi/2$	0	0
2	0	0	0	289.48	T2_offset
3	0	0	$-\pi/2$	77.5	T3_offset
4	0	342.5	$\pi/2$	0	0
5	0	0	$-\pi/2$	76	0
6	0	97	0	0	0

Tx_offset is the offset angle from the mathematical position to the mechanical position.

where $s\theta_i$ is the abbreviation for sin θ_i, $c\theta_i$ for cos θ_i, $s\alpha_i$ for sin α_i, and $c\alpha_i$ for cos α_i. By substituting the D-H parameters into the link transformation equation, the transformation matrix from link 6 to link 0 can be obtained, as shown in Equation (2).

$$\begin{cases} {}^0_6T = {}^0_1T^1_2T^2_3T^3_4T^4_5T^5_6T \\ {}^0_6T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & P_x \\ r_{21} & r_{22} & r_{23} & P_y \\ r_{31} & r_{32} & r_{33} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{3\times3} & P \\ 0 & 1 \end{bmatrix} \end{cases} \tag{2}$$

where ${}^0_6T$ represents the end-effector pose matrix of the robotic arm, $R_{3\times3}$ is the rotation matrix of the end-effector, and P is the translation vector of the end-effector.

2.2. Fitness function for inverse kinematics solution

Although analytical solutions to the IK problem exist for certain robot configurations – such as manipulators with a spherical wrist structure like the PUMA – these solutions are highly dependent on the specific kinematic architecture and may not be applicable to robots with more complex or arbitrary structures. To improve the generality and flexibility of the solution method, this study reformulates the IK problem as a nonlinear optimization problem. The improved DBO algorithm is then employed to numerically solve the joint configurations that minimize the pose error of the robot end-effector.

For the minimization problem, a fitness function must first be constructed. Since the robotic arm's pose parameters consist of a 3×3 rotation matrix and a 3×1 translation vector, the fitness function is defined as shown in Equation (3).

$$\begin{cases} \mathrm{rot_{diff}} = \|\mathrm{T_{rot}} - \mathrm{At_{rot}}\| \\ \mathrm{pos_{diff}} = \|\mathrm{T_{pos}} - \mathrm{At_{pos}}\| \\ \mathrm{F} = \mathrm{rot_{diff}} + \mathrm{pos_{diff}} \end{cases} \tag{3}$$

where T_{rot} represents the actual rotational state of the robotic arm, At_{rot} is the target rotational state, rot_{diff} is the difference between the actual and target rotational states; T_{pos} represents the actual position vector of the end-effector in world coordinates, At_{pos} is the target position vector of the end-effector in world coordinates, pos_{diff} is the difference between the actual and target positions; F represents the total difference between the actual pose matrix and the target pose matrix.

As derived from Equation (3), when the actual pose of the robotic arm matches or closely approximates the target pose, the fitness function F should equal 0 or approach 0.

3. Dung beetle optimization algorithm

DBO Algorithm is an intelligent optimization algorithm proposed to simulate the behaviors of dung beetles, such as rolling dung balls, reproducing, foraging, and stealing in nature. Different dung beetle

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

populations were defined based on four behaviors: ball-rolling dung beetles, breeding dung beetles, juvenile dung beetles, and thief dung beetles. Each dung beetle is positionally updated by a different formula to find the globally optimal solution as much as possible.

3.1. Rolling behavior of dung beetles

In nature, dung beetles roll animal dung into balls and use polarized light for navigation, rolling the dung balls in a straight line toward an ideal location.

To simulate the behavior of dung beetles rolling dung balls, rolling beetles move in a specific direction across the entire search space. The position update equation for rolling beetles is given by Equation (4).

$$\begin{cases} x_i^{t+1} = x_i^t + \alpha \cdot k \cdot x_i^{t-1} + b \cdot \Delta x \\ \Delta x = \left| x_i^t - x_{\text{worst}}^t \right| \end{cases} \tag{4}$$

where t is the current iteration, x_i^t is the position of the i-th beetle at iteration t, k is a constant deflection coefficient with a value range of $[0, 0.2]$, b is a constant between $[0, 1]$, α is a natural coefficient, either 1 or -1, and x_{worst}^t is the worst global position at iteration t.

When rolling beetles encounter obstacles, they perform a "dancing" behavior on top of the dung ball to determine a new direction before continuing to roll. This behavior is simulated using the tangent function, as shown in Equation (5).

$$x_i^{t+1} = x_i^t + \tan(\theta) \left| x_i^t - x_i^{t-1} \right| \tag{5}$$

When $\theta = 0, \pi/2$ or π, $\tan(\theta)$ becomes undefined, meaning the beetle's position remains unchanged.

3.2. Breeding behavior of dung beetles

Female dung beetles roll dung balls to safe locations and hide them to create a suitable environment for offspring. This behavior inspired the boundary constraint strategy used to simulate the egg-laying area of dung beetles, expressed in Equation (6).

$$\begin{cases} \text{Lb}^* = \max\{\text{x}^* \cdot (1 - \text{R}), \text{Lb}\} \\ \text{Ub}^* = \min\{\text{x}^* \cdot (1 + \text{R}), \text{Ub}\} \end{cases} \tag{6}$$

where $R = 1 - t/T$, and T is the maximum number of iterations, Lb and Ub are the lower and upper bounds of the search space, x^* is the global optimal position in the current generation, Lb^* and Ub^* are the lower and upper bounds of the egg-laying region.

Once the egg-laying region is determined, female beetles lay a single breeding ball within this area during each iteration. The position of the breeding ball changes dynamically with the egg-laying region, as given by Equation (7).

$$B_i^{t+1} = \text{x}^\wedge + \text{b}_1 \cdot \left(B_i^t - \text{Lb}^*\right) + \text{b}_2 \cdot \left(B_i^t - \text{Ub}^*\right) \tag{7}$$

where B_i^t is the position of the i-th breeding ball at iteration t, b_1 and b_2 are two independent random vectors of size 1xD, where D is the problem dimension.

As shown in Figure 3, the blue points represent the positions of the breeding balls, the black points represent the local optimum x^*, and the red points represent the upper and lower boundaries of the region.

3.3. Foraging behavior of juvenile beetles

Once the breeding balls successfully hatch, juvenile beetles emerge and forage within the local area. As the iterations progress, the foraging area of juvenile beetles dynamically updates, as shown in Equation (8).

$$\begin{cases} \text{Lb}^\wedge = \max\{\text{x}^\wedge \cdot (1 - \text{R}), \text{Lb}\} \\ \text{Ub}^\wedge = \min\{\text{x}^\wedge \cdot (1 + \text{R}), \text{Ub}\} \end{cases} \tag{8}$$

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

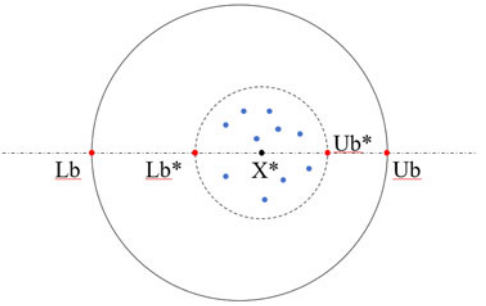


Figure 3. *Boundary selection strategy diagram.*

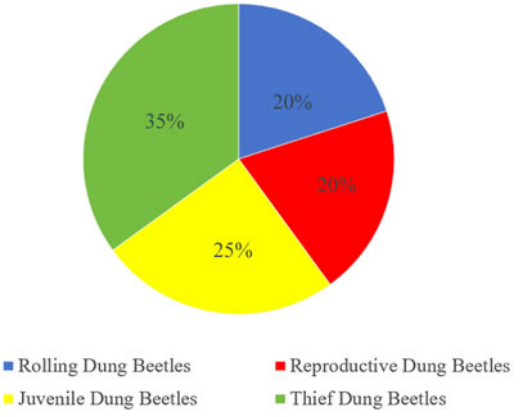


Figure 4. *Various dung beetle types in the population.*

where $R = 1 - t/T$, and T is the maximum number of iterations, $x^{\wedge}$ is the current local optimal position, $Lb^{\wedge}$ and $Ub^{\wedge}$ represent the lower and upper bounds of the foraging area.

The position update for juvenile beetles is given by Equation (9).

$$x_i^{t+1} = x_i^t + C_1 \cdot \left(x_i^t - \mathrm{Lb}^{\wedge}\right) + C_2 \left(x_i^t - \mathrm{Ub}^{\wedge}\right) \tag{9}$$

where x_i^t is the position of the i-th juvenile beetle at iteration t, C_1 is a random number following a normal distribution $C_1 \sim N(0, 1)$, C_2 is a random vector with values in the range (0,1).

3.4. Stealing behavior of dung beetles

In the dung beetle world, the thieves are referred to as stealing beetles, which steal dung balls rolled by other beetles. From the equations provided earlier, x^* and $x^{\wedge}$ represent the optimal positions for stealing dung balls. The position update for the stealing beetles is expressed as Equation (10).

$$x_i^{t+1} = \mathrm{x}^{\wedge} + \mathrm{S} \cdot \mathrm{g} \cdot \left(\left|x_i^t - \mathrm{x}^*\right| + \left|x_i^t - \mathrm{x}^{\wedge}\right|\right) \tag{10}$$

where x_i^t represents the position of the i-th stealing beetle at iteration t, g is a random vector of size 1xD drawn from a normal distribution $g \sim N(0, 1)$, S is a constant.

Figure 4 illustrates the proportion of different types of dung beetles in the population. The blue section represents rolling beetles, accounting for 20% of the total population. The red section corresponds to breeding beetles, also comprising 20%. The yellow section indicates juvenile beetles, which make up 25%, and the green section represents stealing beetles, constituting 35% of the overall population.

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

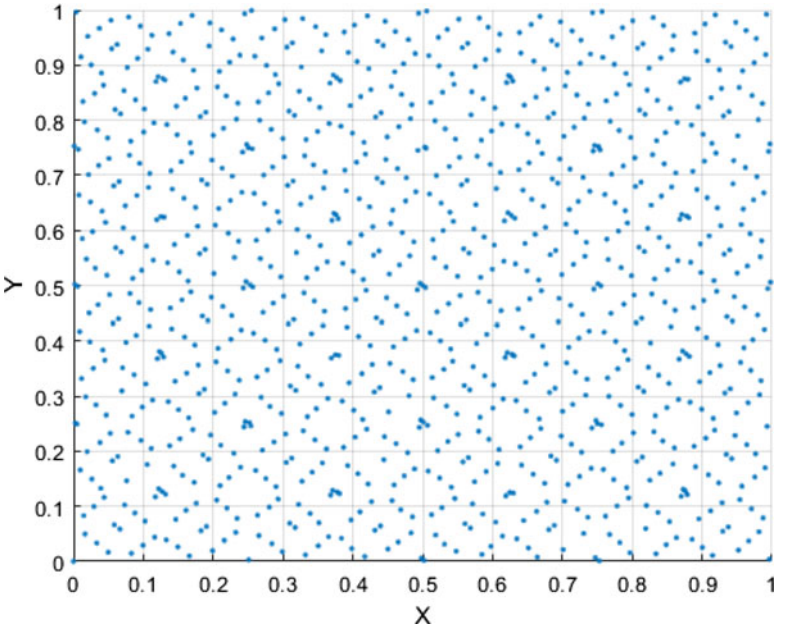


Figure 5. *Sobol sequence.*

4. Improving dung beetle optimization algorithm

4.1. Sobol sequence initialization

Traditional swarm intelligence optimization algorithms use generation of random numbers distributed in the solution space when generating the initial population. In contrast, random numbers generated based on modern CPUs are pseudo-random, with duplication and aggregation, resulting in reduced population diversity. Sobol sequences are low-discrepancy sequences that are more evenly distributed in space than pseudo-random sequences. Good overall coverage is ensured by precisely controlling the position in space of each newly generated point.

As shown in Figures 5 and 6, a comparison of using Sobol sequences and pseudo-random sequences to generate population initiation locations in 2D space. Where Figure 4 shows the location of dung beetle populations generated using Sobol sequences, it can be seen that they are able to be evenly distributed and somewhat differentiated in any area. Figure 5 shows the distribution of populations generated using pseudo-random sequences, and it can be seen that multiple population clusters occur in the figure, as well as some areas that are not covered.

Sobol sequences are a class of low-discrepancy sequences based on binary reflections that are commonly used to generate uniformly distributed sampling points in high-dimensional spaces. For a Sobol sequence of dimension d, the generating process primitive polynomials and the number of directions. The formula for the mth point of the Sobol sequence is shown in Equation (11).

$$x_m = \sum_{i=1}^{n} \frac{v_{i,j}}{2^i} \tag{11}$$

where $v(i, j)$ denotes the number of directions in the j dimension and i denotes the order of the binary bits.

4.2. Boundary optimization strategy

From the convergence factor $R = 1 - t/T$ in the reproductive dung beetle boundary formula, we know that as the number of iterations increases, the area available for reproduction decreases. In the early

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

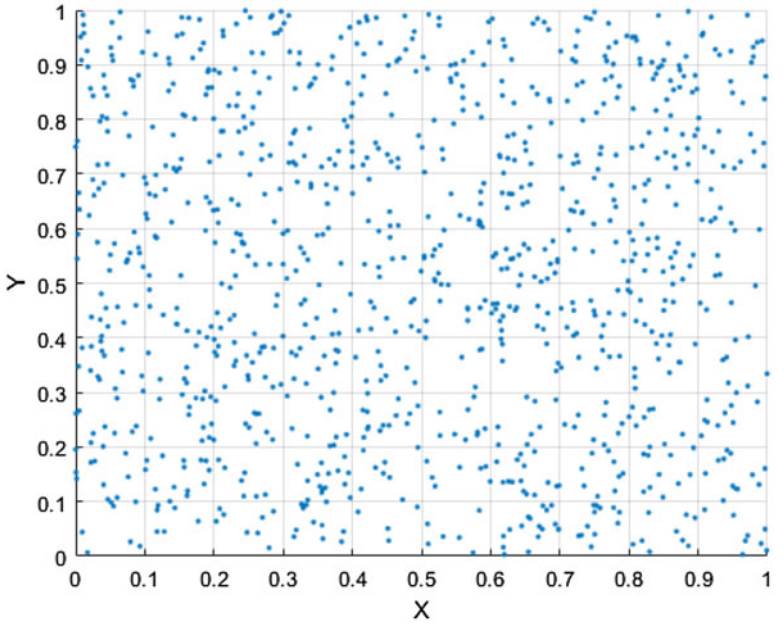


Figure 6. *Pseudorandom sequence.*

iterations, the value of R is larger, leading to a larger optimal breeding area, enhancing the global search ability of the algorithm. As R decreases over time, the search area shrinks, which strengthens the local search ability in later iterations.

To balance global search in the early stages and local search in the later stages, we redefine the convergence factor as a piecewise function. Initially, the convergence remains unchanged, preserving the global search capability. Later, a logistic function is introduced to adjust the convergence speed of R, enhancing the algorithm's local search capacity. The logistic function is described in Equation (12).

$$\begin{cases} \mathrm{R} = 1 - \frac{\mathrm{t}}{\mathrm{T}}, \mathrm{t} \le \mathrm{T}/2 \\ \mathrm{R} = 1 - \dfrac{1}{1 + \mathrm{e}^{-\mathrm{k}\cdot\left(\frac{\mathrm{T}}{2} - \mathrm{t}\right)}}, t > T/2 \end{cases} \tag{12}$$

where T is the number of iterations, t is the current iteration, and k is a constant, set to –0.01 in this study.

As shown in Figure 7, the logistic function maintains an appropriate convergence speed during the early iterations, resulting in a larger optimal search area and improved global search capability. In the later stages, the logistic function accelerates the search process, shrinking the optimal foraging area and enhancing local search, thereby improving the convergence speed.

4.3. Hybrid directional perturbation strategies

During the search process, the Rolling Dung Beetle and Stealing Dung Beetle will explore forward along a straight line path based on the paths generated by Equation (4) and Equation (10). Such a search method greatly diminishes their ability to explore, and they tend to miss opportunities to find better solutions. A hybrid directional perturbation strategy is proposed for this problem, which perturbs the forward direction of the rolling dung beetle and the stealing dung beetle to increase the chances of finding an optimal solution.

Conventional normally distributed perturbations can produce moderate perturbations that improve search. However, when faced with complex optimization problems, especially with multiple local optimal solutions, the normal distribution may not be able to jump out of the local optimum. T-distributed

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

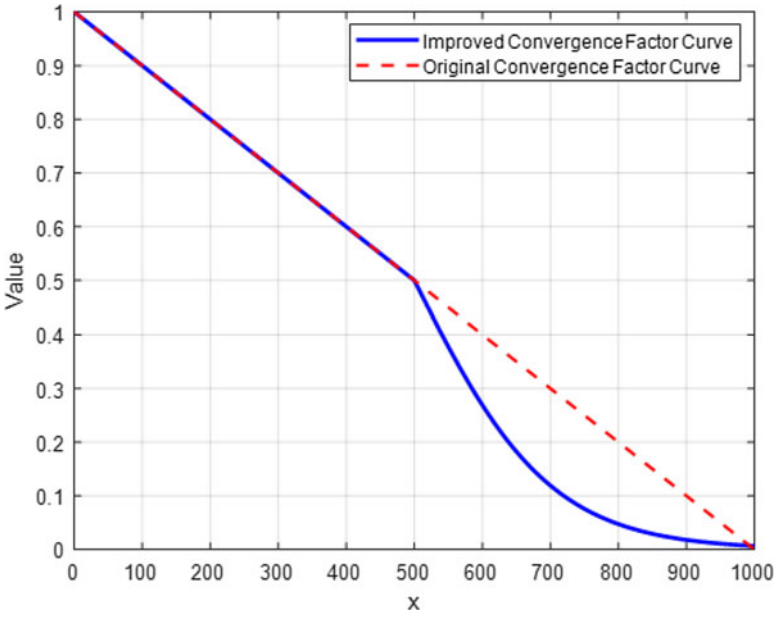


Figure 7. *Variation curve of convergence factor.*

perturbations have thicker tails compared to normal-distributed perturbations, increasing the probability of generating large perturbations during the search process and improving the algorithm's ability to jump out of local optima. T-distribution is a symmetric distribution with a probability density function as shown in Equation (13). where M is the degree of freedom, which determines the shape of the distribution, the t-distribution gradually converges to a normal distribution as the degree of freedom increases.

$$f(n) = \frac{\Gamma\left(\frac{m+1}{2}\right)}{\sqrt{m\pi}\,\Gamma\left(\frac{m}{2}\right)}\left(1+\frac{n^2}{m}\right)^{-\frac{m+1}{2}} \tag{13}$$

During the iteration process, the t-distribution perturbation is introduced into the update of the search position with a perturbation factor formulated as equation (14).

$$w = \mu_w + \sigma_w \cdot t_m \tag{14}$$

where μ_w is the offset. σ_w is the scale parameter, which controls the magnitude of the perturbation. t_m is a random variable drawn from a t-distribution with m degrees of freedom.

The variation curve of the disturbance factor at this time is shown in Figure 8.

Dynamic weight perturbations can change the perturbation factor as the number of iterations increases. Having a perturbation factor with a strong effect at the beginning of the iteration enables exploration over a large range. With the increase in the number of sum iterations, the perturbation factor gradually converges to 1, the perturbation effect is weakened, and a more detailed search can be carried out within a certain range, increasing the convergence accuracy. The dynamic weight perturbation factor formula is shown in equation (15). The dynamic perturbation factor w curve is shown in Figure 9.

$$w = e^{-\frac{T-t}{T}} \tag{15}$$

where T is the maximum number of iterations and t is the current number of iterations.

T-distribution has thicker tails compared to the normal distribution, which means a greater probability of producing values far from the mean. The ability to introduce large random fluctuations allows the ball-rolling dung beetle to have a greater chance of jumping out of the current local optimum region

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

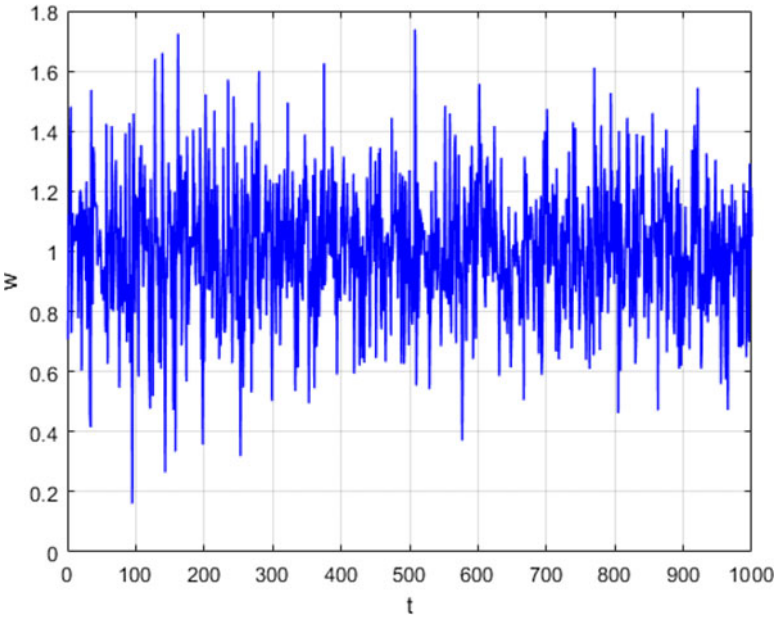


Figure 8. *w perturbation curve.*

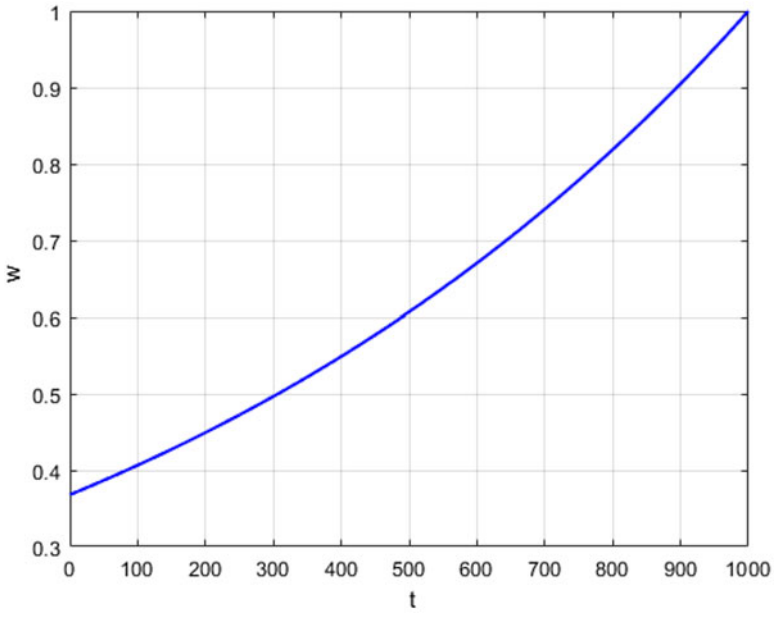


Figure 9. *Disturbance factor w.*

and exploring unknown regions. The dynamic weight perturbations decay exponentially, and the magnitude of the perturbations is on the small side, which enables the Stealing Dung Beetle to explore the region in greater detail and helps the algorithm to approximate the global optimum more quickly. The two perturbation strategies were used in combination, with two consecutive t-distribution perturbations followed by a single dynamic weight perturbation on the parameters k and b for the ball-rolling dung beetle and on the parameter S for the stealing dung beetle, which in turn controlled the search routes of

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

the two dung beetle types. As shown in Equation (16).

$$\begin{cases} k = k * w \\ b = b * w \\ S = S * w \end{cases} \tag{16}$$

Two consecutive perturbations of the t-distribution, while increasing the dung beetle's global search ability, may also cause the search to be too far off course. A dynamic weight perturbation at this point can correct the search direction to avoid the problem of too much deviation. At the same time, the addition of dynamic weight perturbation makes the dung beetle search sufficiently in a small area and accelerates the convergence of the algorithm, and the hybrid strategy works better than any of the perturbation strategies alone.

4.4. Levy flight strategy

In order to solve the problem that the original DBO algorithm suffers from the tendency to fall into local optimums, the Levy flight strategy is introduced. The Levy flight strategy is a stochastic wandering method with heavy-tailed properties. Unlike Brownian motion, Levy flight allows for episodic large step-size jumps, enabling the dung beetle population to search over a larger area, improving the algorithm's global search capability and enhancing the ability to jump out of local optima.

The step size s of the Levy flight is generated by Equation (17).

$$s = \frac{u}{v^{\frac{1}{\beta}}} \tag{17}$$

where v and u obey a normal distribution: $v \sim (0, 1)$, $u \sim N(0, \sigma_u^2)$, σ_u is the scale parameter, determined by Equation (18).

$$\sigma_u = \left(\frac{\Gamma(1+\beta) \sin \frac{\pi\beta}{2}}{\Gamma\left(\frac{1+\beta}{2}\right) \beta 2^{\frac{\beta-1}{2}}} \right)^{\frac{1}{\beta}} \tag{18}$$

Γ is the gamma function. β is the Levy distribution index, and $\beta = 1.5$.

During the iteration of the algorithm, the Levy flight strategy is used to generate new candidate solutions that jump out of the local optimum by perturbing the current optimal solution. In contrast to other perturbation methods, the Levy flight strategy has a perturbation rule for short-range perturbations and occasional long-range perturbations. The short-range perturbation ensures that the algorithm is able to perform a detailed search of the local space and avoids frequent perturbations that affect the convergence performance of the algorithm. And occasional long-range perturbations help the algorithm to go beyond the local optimum and explore a larger range. Levy flight jump trajectory is shown in Figure 10.

4.5. Complexity of ECDBO algorithm

In the initial phase of the algorithm, a Sobol sequence initiation strategy is introduced, which generates a Sobol sequence initial solution for some of the populations with a complexity of O(N ∗ D), where N is the population size and D is the problem dimension. The bounded optimization strategy is used to change the complexity to O(N ∗ D). Dynamic weight perturbations and t-distribution perturbations are used to update the parameters, and since the dynamic weights w are obtained by simple scalar computation and the generation of t-distributed random numbers is constant time in expectation, the complexity of

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

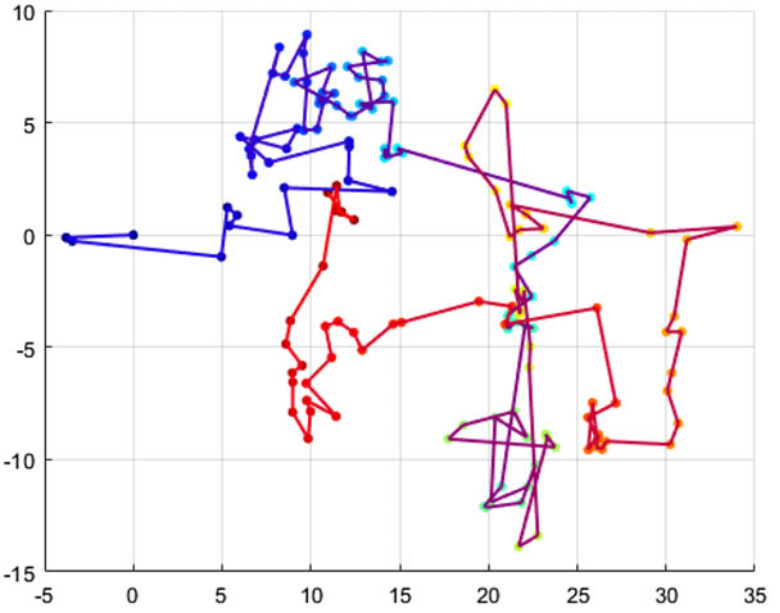


Figure 10. *Levy flight path.*

this part is O(1), which has little impact on the overall time complexity. The Levy flight strategy is used to perturb the globally optimal solution, which involves operations on D-dimensional vectors with O(D) complexity. Taken together, the main time consumption of the ECDBO algorithm in each iteration remains the updating and fitness assessment of the population with a complexity of $O(N * D)$, and the fitness ranking with a complexity of O(N log N). Thus, the overall time complexity of the algorithm is $O(T * (N * D + N \log N))$, where T is the number of iterations. The overall time complexity of the DBO algorithm is also $O(T * (N * D + N \log N))$. Compared to the DBO algorithm, the convergence speed and accuracy of the algorithm are substantially improved without increasing the time cost.

4.6. ECDBO algorithm workflow

The flow of Enhanced Chaotic Dung Beetle Optimization (ECDBO) algorithm can be summarized in the following pseudocode.

5. Experimental results and analysis

This section tests the performance of ECDBO using seven comparison algorithms and 30 test functions. The test function is CEC2017 and two dimensions 30,100 are selected for testing, and the comparison algorithms are WOA [23], DBO, BOA [24], PSO [25], iCSPM [26], MDBO and GSA [27], which are validated in many literatures. For the fairness of the experiment, in this paper, the population size of all algorithms is uniformly set to 30, and the number of iterations is 1000. To avoid chance in the experiment, each algorithm was run independently 30 times, and the mean and standard deviation were ranked for comparison.

Benchmark function testing for intelligent optimization algorithms aims to evaluate their performance and adaptability in solving different problems. The experiments were conducted on a Windows 11 Professional operating system, with a 12th Gen Intel(R) Core(TM) i5-12400F?processor (2.50 GHz), 32 GB of memory (3200 MHz), and an NVIDIA GeForce GTX 4060 GPU. MATLAB 2023a was used as the simulation software.

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

Algorithm 1 ECDBO Algorithm

Input: objective function, relevant parameters
Output: the optimal solution value

Use Equation (11) to initialize the dung beetle population positions
while $t < T$ **do**
  Levy flight strategies perturb the current optimal solution
  Perturbing the k, b, S **parameters**
  for $i = 1{:}N$ **do**
    Update the positions of the worst-performing dung beetles using Equations (13) and (14)
    if i ∈ Rolling Dung Beetles **then**
      if $\delta < 0.9$ **then**
        Update the positions of the rolling dung beetles using Equation (4)
      else
        Update the positions of the rolling dung beetles using Equation (5)
      end if
    end if
    Dynamically determine the optimal region using Equation (12)
    if $i \in$ Reproductive Dung Beetles **then**
      Update the positions of the reproductive dung beetles using Equation (7)
    end if
    if $i \in$ Juvenile Dung Beetles **then**
      Update the positions of juvenile dung beetles using Equation (9)
    end if
    if $i \in$ Thief Dung Beetles **then**
      Update the positions of thief dung beetles using Equation (10)
    end if
  end for
end while

5.1. CEC2017 function experimentation and analysis

The CEC2017 test set consists of 30 single-objective functions, including single-peak functions (F1-F3), simple multi-peak functions (F4-F10), hybrid functions (F11-F20), and combined functions (F21-F30). Figures 11 and 12 show the ranking of the average results of the eight algorithms after 30 independent runs in different dimensions.

5.1.1. Numerical analysis of CEC2017 results

The experimental results of ECDBO algorithm in 30 and 100 dimensions are recorded in Tables III and IV. It shows the mean, standard deviation, and ranking of each algorithm. The data is analyzed as follows:

- ECDBO algorithm is ahead or even significantly ahead of the other seven algorithms in both the 30- and 100-dimensional tests for the single-peak functions F1-F3, which indicates that the ECDBO algorithm excels in its ability to find the global optimum. Overall, the ECDBO algorithm performs well in solving the single-peaked function problem.
- The results of the simple multi-peak function F4-F10 test show that the ECDBO algorithm outperforms the other seven algorithms. In the 30-dimensional test, the ECDBO algorithm outperforms the other algorithms on the functions F4, F7, and F8, and ranks first. It is slightly weaker than the DBO algorithm on functions F5 and F9 and ranks second. It is only weaker than the iCSPM algorithm in functions F6 and F10, and ranks second. In the 100-dimensional test, the ECDBO algorithm outperforms the other seven algorithms in functions F4, F5, F7, F8 and

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

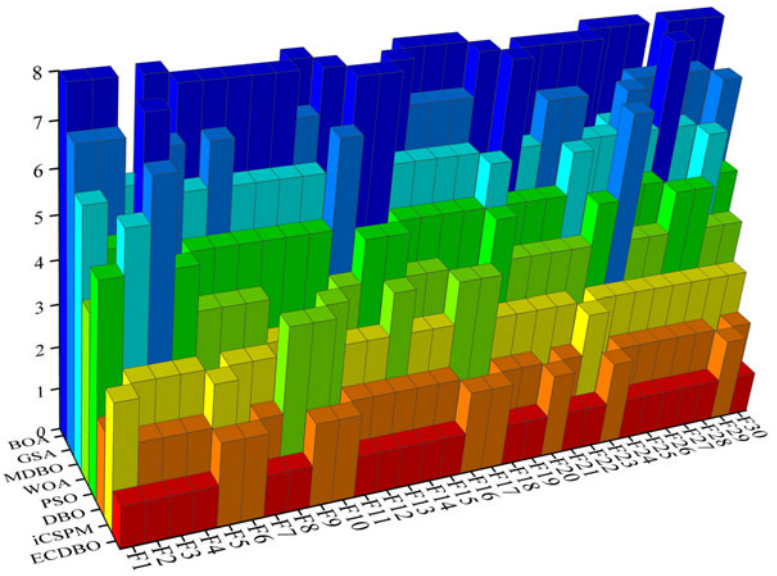


Figure 11. *Average ranking under 30 dimensions.*

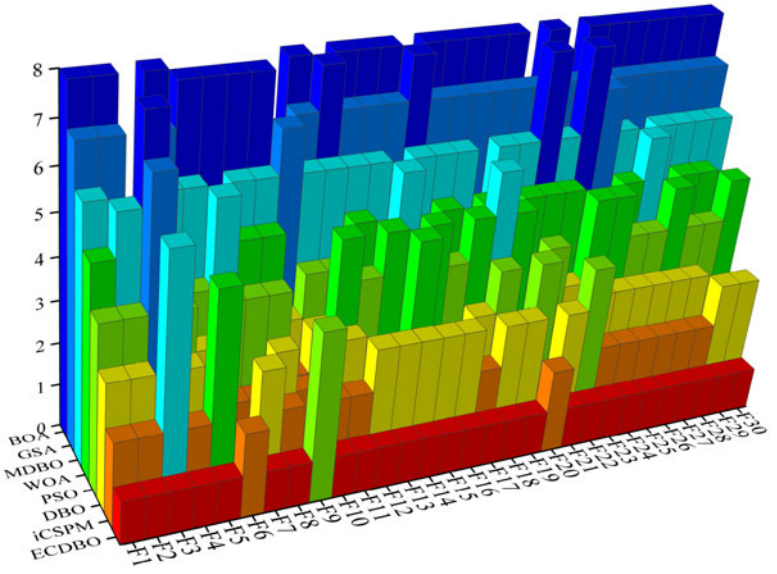


Figure 12. *Average ranking under 100 dimensions.*

F10, and ranks first, and is only slightly weaker than the DBO algorithm in function F6, and is weaker than the DBO, PSO and WOA algorithms in function F9, and ranks fourth. These results verify the superiority of ECDBO algorithm in solving multi-objective problems, showing very good performance.

- ECDBO has very good convergence performance on hybrid functions F11-F20. In the 30-dimensional test, ECDBO tops the list of 7 hybrid functions, which is attributed to the improved search strategy and the ability to jump out of local optima. In the 100-dimensional test, ECDBO outperforms the other algorithms in 9 functions, and is almost equal to the iCPSM algorithm only in function F20.

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

Table III. *CEC2017 test results, dim = 30.*

		ECDBO	WOA	DBO	BOA	PSO	iCSPM	MDBO	GSA
F1	Degree	**1**	4	2	8	5	3	6	7
	Average	**8.18E + 03**	1.54E + 09	6.58E + 07	5.30E + 10	1.54E + 09	2.77E + 08	3.52E + 10	5.08E + 10
	STD	**7.93E + 03**	6.22E + 08	6.82E + 07	7.75E + 09	2.12E + 09	4.74E + 07	3.03E + 09	8.04E + 09
F2	Degree	**1**	4	3	8	6	2	5	7
	Average	**8.28E + 12**	1.46E + 32	2.79E + 30	7.04E + 52	2.57E + 40	3.27E + 19	7.58E + 39	9.72E + 50
	STD	**1.98E + 13**	5.47E + 32	7.44E + 30	1.75E + 53	9.53E + 40	5.41E + 19	4.15E + 40	5.32E + 51
F3	Degree	**1**	8	3	5	7	2	4	6
	Average	**3.56E + 04**	2.61E + 05	6.62E + 04	8.14E + 04	1.02E + 05	5.48E + 04	7.28E + 04	9.37E + 04
	STD	**9.72E + 03**	6.55E + 04	9.98E + 03	8.79E + 03	6.71E + 04	1.41E + 04	7.99E + 03	7.98E + 02
F4	Degree	**1**	4	3	8	5	2	7	6
	Average	**4.95E + 02**	8.12E + 02	5.56E + 02	2.10E + 04	8.77E + 02	5.38E + 02	1.50E + 04	1.14E + 04
	STD	**2.15E + 01**	1.59E + 02	5.47E + 01	3.31E + 03	4.59E + 02	2.86E + 01	1.74E + 03	3.62E + 03
F5	Degree	2	5	**1**	7	4	3	6	8
	Average	7.06E + 02	8.31E + 02	**6.88E + 02**	9.07E + 02	7.17E + 02	7.08E + 02	8.84E + 02	9.47E + 02
	STD	5.05E + 01	5.80E + 01	**3.65E + 01**	2.63E + 01	4.34E + 01	2.90E + 01	3.97E + 01	3.87E + 01
F6	Degree	2	5	3	7	4	**1**	7	8
	Average	6.34E + 02	6.77E + 02	6.39E + 02	6.88E + 02	6.53E + 02	**6.23E + 02**	6.86E + 02	6.95E + 02
	STD	1.23E + 01	1.50E + 01	1.01E + 01	6.95E + 00	1.21E + 01	**9.34E + 00**	1.29E + 01	6.15E + 00
F7	Degree	**1**	5	3	7	4	2	6	8
	Average	**9.22E + 02**	1.30E + 03	1.02E + 03	1.39E + 03	1.09E + 03	9.47E + 02	1.31E + 03	1.47E + 03
	STD	**5.01E + 01**	8.43E + 01	7.71E + 01	5.84E + 01	7.73E + 01	1.87E + 01	5.12E + 01	4.67E + 01
F8	Degree	**1**	5	2	7	3	4	6	8
	Average	**9.50E + 02**	1.04E + 03	9.63E + 02	1.14E + 03	9.82E + 02	1.01E + 03	1.05E + 03	1.15E + 03
	STD	**3.89E + 01**	4.17E + 01	3.17E + 01	1.88E + 01	4.05E + 01	2.62E + 01	6.13E + 01	2.62E + 01
F9	Degree	2	5	**1**	7	3	4	6	8
	Average	5.58E + 03	9.78E + 03	**5.29E + 03**	1.07E + 04	5.95E + 03	7.08E + 03	1.05E + 04	1.15E + 04
	STD	2.19E + 03	2.72E + 03	**1.83E + 03**	1.33E + 03	2.09E + 03	2.57E + 03	1.27E + 03	1.45E + 03
F10	Degree	2	5	4	8	3	**1**	6	7
	Average	5.70E + 03	6.55E + 03	6.18E + 03	9.01E + 03	6.11E + 03	**5.55E + 03**	8.09E + 03	8.72E + 03
	STD	8.90E + 02	6.79E + 02	9.58E + 02	3.19E + 02	9.49E + 02	**4.65E + 02**	1.00E + 03	5.73E + 02
F11	Degree	**1**	7	3	6	4	2	5	8
	Average	**1.31E + 03**	7.50E + 03	1.53E + 03	7.34E + 03	4.44E + 03	1.34E + 03	4.55E + 03	1.21E + 04
	STD	**7.45E + 01**	3.56E + 03	1.34E + 02	2.27E + 03	8.27E + 03	3.46E + 01	8.38E + 02	2.78E + 03
F12	Degree	**1**	4	3	7	5	2	8	6
	Average	**1.17E + 06**	2.75E + 08	3.27E + 07	1.24E + 10	5.66E + 08	1.12E + 07	1.28E + 10	9.89E + 09
	STD	**9.98E + 05**	1.87E + 08	3.97E + 07	3.66E + 09	9.56E + 08	6.24E + 06	2.12E + 09	3.90E + 09
F13	Degree	**1**	3	4	7	5	2	8	6
	Average	**1.96E + 05**	2.35E + 06	3.97E + 06	1.08E + 10	2.26E + 08	2.72E + 05	1.32E + 10	2.24E + 09
	STD	**8.05E + 05**	2.01E + 06	9.19E + 06	5.33E + 09	7.83E + 08	3.87E + 05	3.92E + 09	2.60E + 09
F14	Degree	**1**	5	3	7	4	2	6	8
	Average	**6.60E + 04**	2.09E + 06	1.49E + 05	5.11E + 06	6.37E + 05	1.24E + 05	2.60E + 06	1.51E + 07
	STD	**4.64E + 04**	1.86E + 06	2.17E + 05	5.98E + 06	1.87E + 06	1.16E + 05	7.86E + 05	4.07E + 07
F15	Degree	**1**	5	3	8	4	2	6	7
	Average	**9.15E + 03**	2.14E + 06	3.16E + 04	6.41E + 08	5.42E + 04	1.56E + 04	2.52E + 08	3.08E + 08
	STD	**1.13E + 04**	7.54E + 06	2.58E + 04	4.00E + 08	5.26E + 04	1.06E + 04	1.33E + 08	2.15E + 08
F16	Degree	2	5	4	8	3	**1**	6	7
	Average	2.83E + 03	4.16E + 03	3.26E + 03	7.41E + 03	3.23E + 03	**2.64E + 03**	4.31E + 03	6.02E + 03
	STD	2.94E + 02	6.64E + 02	3.95E + 02	1.38E + 03	5.26E + 02	**2.95E + 02**	5.26E + 02	1.09E + 03
F17	Degree	2	5	4	8	3	**1**	6	7
	Average	2.45E + 03	2.77E + 03	2.61E + 03	1.21E + 04	2.59E + 03	**2.16E + 03**	3.55E + 03	4.47E + 03
	STD	2.46E + 02	2.53E + 02	2.85E + 02	1.35E + 04	3.95E + 02	**1.64E + 02**	2.71E + 02	3.43E + 03
F18	Degree	**1**	6	3	7	5	2	4	8
	Average	**5.36E + 05**	7.58E + 06	1.75E + 06	7.40E + 07	6.36E + 06	8.14E + 05	6.31E + 06	1.05E + 08
	STD	**6.02E + 05**	7.68E + 06	3.48E + 06	6.26E + 07	2.26E + 07	6.71E + 05	8.33E + 06	1.58E + 08
F19	Degree	**1**	5	3	7	4	2	8	6
	Average	**6.85E + 03**	1.08E + 07	1.65E + 05	6.66E + 08	1.25E + 06	1.88E + 04	8.25E + 08	4.42E + 08
	STD	**6.02E + 03**	1.07E + 07	2.66E + 05	5.02E + 08	4.05E + 06	1.44E + 04	3.39E + 08	3.97E + 08
F20	Degree	2	5	3	7	4	**1**	6	8
	Average	2.58E + 03	2.89E + 03	2.60E + 03	3.03E + 03	2.68E + 03	**2.44E + 03**	2.91E + 03	3.07E + 03
	STD	1.82E + 02	2.74E + 02	1.69E + 02	1.21E + 02	2.14E + 02	**1.51E + 02**	1.10E + 02	1.67E + 02
F21	Degree	**1**	5	3	6	4	2	7	8
	Average	**2.46E + 03**	2.62E + 03	2.51E + 03	2.70E + 03	2.51E + 03	2.50E + 03	2.72E + 03	2.73E + 03
	STD	**3.39E + 01**	6.18E + 01	3.46E + 01	1.07E + 02	4.59E + 01	3.31E + 01	3.66E + 01	5.50E + 01

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

Table III. *Continued*

		ECDBO	WOA	DBO	BOA	PSO	iCSPM	MDBO	GSA
F22	Degree	**1**	6	2	5	4	3	7	8
	Average	**2.44E + 03**	7.89E + 03	2.99E + 03	6.64E + 03	5.74E + 03	4.50E + 03	8.11E + 03	9.83E + 03
	STD	**7.79E + 02**	1.82E + 03	1.64E + 03	1.29E + 03	2.31E + 03	2.32E + 03	5.63E + 02	6.94E + 02
F23	Degree	2	4	3	7	5	**1**	6	8
	Average	2.85E + 03	3.13E + 03	3.01E + 03	3.46E + 03	3.32E + 03	**2.83E + 03**	3.41E + 03	3.50E + 03
	STD	5.12E + 01	9.96E + 01	1.02E + 02	1.34E + 02	2.63E + 02	**6.64E + 01**	1.75E + 02	1.63E + 02
F24	Degree	**1**	4	3	8	7	2	6	5
	Average	**3.01E + 03**	3.22E + 03	3.14E + 03	4.01E + 03	3.64E + 03	3.05E + 03	3.64E + 03	3.62E + 03
	STD	**6.00E + 01**	8.94E + 01	8.15E + 01	2.55E + 02	1.49E + 02	5.45E + 01	1.50E + 02	1.88E + 02
F25	Degree	**1**	5	3	8	4	2	7	6
	Average	**2.90E + 03**	3.10E + 03	2.93E + 03	5.82E + 03	2.99E + 03	2.93E + 03	4.77E + 03	4.46E + 03
	STD	**1.35E + 01**	5.14E + 01	2.85E + 01	5.95E + 02	4.74E + 01	1.54E + 01	2.73E + 02	3.46E + 02
F26	Degree	**1**	5	3	8	4	2	6	7
	Average	**4.23E + 03**	8.56E + 03	6.03E + 03	1.19E + 04	7.17E + 03	4.62E + 03	9.24E + 03	1.14E + 04
	STD	**1.64E + 03**	9.07E + 02	1.64E + 03	8.75E + 02	2.14E + 03	1.40E + 03	5.69E + 02	1.16E + 03
F27	Degree	**1**	4	3	6	5	2	8	7
	Average	**3.24E + 03**	3.42E + 03	3.32E + 03	4.17E + 03	3.59E + 03	3.24E + 03	4.32E + 03	4.22E + 03
	STD	**1.71E + 01**	1.03E + 02	7.32E + 01	3.21E + 02	2.14E + 02	2.63E + 01	5.98E + 02	4.26E + 02
F28	Degree	**1**	4	3	8	5	2	6	7
	Average	**3.22E + 03**	3.57E + 03	3.35E + 03	8.06E + 03	3.53E + 03	3.29E + 03	5.67E + 03	6.40E + 03
	STD	**2.13E + 01**	1.05E + 02	5.79E + 01	6.48E + 02	3.23E + 02	4.19E + 01	1.65E + 03	6.26E + 02
F29	Degree	2	6	3	8	4	**1**	5	7
	Average	4.07E + 03	5.25E + 03	4.46E + 03	1.29E + 04	4.58E + 03	**4.00E + 03**	4.62E + 03	8.67E + 03
	STD	2.81E + 02	5.25E + 02	3.64E + 02	8.78E + 03	5.20E + 02	**1.93E + 02**	6.92E + 02	2.47E + 03
F30	Degree	**1**	5	3	8	4	2	7	6
	Average	**4.88E + 04**	4.61E + 07	3.06E + 06	1.33E + 09	4.79E + 06	2.23E + 05	1.30E + 09	1.18E + 09
	STD	**8.28E + 04**	4.48E + 07	5.54E + 06	8.47E + 08	9.27E + 06	1.24E + 05	5.62E + 08	1.01E + 09

- ECDBO also performs well in the combined functions of F21-F30. In the 30-dimension test, it ranks first in 8 test functions, and is only slightly weaker than the iCSPM algorithm in F23 and F29. In the 100-dimension test, it ranks first in 10 test functions ahead of the other algorithms. These results show that the ECDBO algorithm is highly competitive in solving difficult and complex problems. Its excellent global exploration and local exploitation capabilities, as well as its remarkable ability to jump out of local optima, provide strong support for solving complex problems.

5.1.2. CEC2017 convergence curve analysis

Algorithm convergence analysis is an important metric for evaluating the algorithm's global and local search capabilities. Figures 13 and 14 show the convergence curves of ECDBO with the other seven comparison algorithms on the 30 functions of the CEC2017 test set.

In the 30-dimensional test, the ECDBO algorithm is significantly ahead of the other seven algorithms on functions F1, F2, F4, F12, F13, F14, F19, F21, F22, F24, F26, and F30, which can be attributed to the strong global exploration and local development capability of the ECDBO algorithm. Although the other algorithms may converge faster than ECDBO at the beginning of the iteration, they are soon overtaken by ECDBO, and the gap is getting wider and wider due to the lack of development. For functions F3, F7, F8, F11, F15, F18, F24, F25, F27, F28, and F29, the ECDBO algorithm and other better-performing algorithms (e.g., iCSPM, DBO, and PSO algorithms) alternately take the lead at the beginning of the iteration, and at the later stage of the iteration, the ECDBO algorithm eventually takes the lead due to its excellent development capability.

In the 100-dimensional test, the ECDBO algorithm performs well on functions F1, F4, F5, F10, F12, F13, F14, F15, F16, F18, F19, F21, F22, F23, F25, F28, and F30, and the final convergence result is significantly ahead of the other algorithms. On functions F2, F3, F5, F7, F11, F17, F20, F24, F26, F27, and F29, the ECDBO algorithm and other algorithms show alternating leads, but soon the ECDBO algorithm exceeds the other algorithms in terms of convergence speed and accuracy, and maintains a

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

Table IV. *CEC2017 test results, dim = 100.*

		ECDBO	WOA	DBO	BOA	PSO	iCSPM	MDBO	GSA
F1	Degree	**1**	5	3	8	4	2	6	7
	Average	**1.77E + 08**	6.40E + 10	2.50E + 10	2.61E + 11	4.19E + 10	1.69E + 10	2.44E + 11	2.56E + 11
	STD	**9.53E + 07**	7.97E + 09	8.93E + 09	1.28E + 10	7.54E + 09	1.50E + 09	1.34E + 10	1.54E + 10
F2	Degree	**1**	6	3	8	4	2	5	7
	Average	**1.64E + 112**	7.92E + 174	1.44E + 145	2.89E + 199	9.64E + 161	1.71E + 118	3.67E + 165	3.48E + 180
	STD	**8.99E + 112**	INF	7.82E + 145	INF	INF	7.83E + 118	INF	INF
F3	Degree	**1**	8	2	5	7	6	4	3
	Average	**3.52E + 05**	9.64E + 05	3.61E + 05	3.70E + 05	7.16E + 05	6.14E + 05	3.68E + 05	3.66E + 05
	STD	**4.34E + 04**	1.34E + 05	6.22E + 04	3.86E + 04	2.61E + 05	5.08E + 04	2.01E + 04	2.67E + 03
F4	Degree	**1**	5	3	8	4	2	6	7
	Average	**9.43E + 02**	1.14E + 04	3.37E + 03	1.14E + 05	6.30E + 03	2.01E + 03	7.91E + 04	9.88E + 04
	STD	**7.96E + 01**	2.39E + 03	1.35E + 03	7.04E + 03	2.27E + 03	2.63E + 02	8.57E + 03	1.45E + 04
F5	Degree	**1**	4	2	7	3	5	6	8
	Average	**1.33E + 03**	1.87E + 03	1.43E + 03	2.09E + 03	1.79E + 03	1.89E + 03	2.00E + 03	2.11E + 03
	STD	**7.79E + 01**	1.36E + 02	7.87E + 01	2.57E + 01	9.33E + 01	1.49E + 02	1.50E + 02	5.08E + 01
F6	Degree	2	6	**1**	7	4	3	5	8
	Average	6.66E + 02	7.05E + 02	**6.65E + 02**	7.11E + 02	6.91E + 02	6.89E + 02	7.01E + 02	7.15E + 02
	STD	6.61E + 00	1.21E + 01	**5.73E + 00**	2.91E + 00	7.62E + 00	8.40E + 00	1.23E + 01	3.34E + 00
F7	Degree	**1**	5	3	7	4	2	6	8
	Average	**2.35E + 03**	3.67E + 03	2.97E + 03	3.95E + 03	3.46E + 03	2.44E + 03	3.76E + 03	4.05E + 03
	STD	**2.40E + 02**	1.55E + 02	3.18E + 02	7.70E + 01	2.15E + 02	9.96E + 01	1.78E + 02	6.66E + 01
F8	Degree	**1**	5	2	7	4	3	6	8
	Average	**1.74E + 03**	2.35E + 03	1.82E + 03	2.55E + 03	2.20E + 03	2.17E + 03	2.36E + 03	2.63E + 03
	STD	**8.09E + 01**	1.54E + 02	1.12E + 02	4.25E + 01	9.36E + 01	1.15E + 02	1.99E + 02	6.55E + 01
F9	Degree	4	3	**1**	6	2	8	7	5
	Average	7.31E + 04	6.94E + 04	**4.95E + 04**	8.25E + 04	6.11E + 04	1.09E + 05	8.33E + 04	7.71E + 04
	STD	8.63E + 03	1.63E + 04	**1.76E + 04**	4.05E + 03	8.46E + 03	1.62E + 04	4.45E + 03	5.08E + 03
F10	Degree	**1**	4	2	8	3	5	6	7
	Average	**2.18E + 04**	2.73E + 04	2.29E + 04	3.32E + 04	2.56E + 04	2.85E + 04	3.18E + 04	3.29E + 04
	STD	**4.52E + 03**	1.69E + 03	5.16E + 03	4.79E + 02	1.82E + 03	6.35E + 02	1.28E + 03	1.12E + 03
F11	Degree	**1**	4	3	7	5	2	6	8
	Average	**3.62E + 04**	2.06E + 05	1.23E + 05	2.42E + 05	2.15E + 05	6.84E + 04	2.18E + 05	3.16E + 05
	STD	**1.06E + 04**	6.73E + 04	4.38E + 04	3.61E + 04	1.43E + 05	1.76E + 04	3.58E + 04	9.62E + 04
F12	Degree	**1**	5	2	8	4	3	6	7
	Average	**9.10E + 07**	1.34E + 10	2.92E + 09	1.95E + 11	1.34E + 10	3.22E + 09	1.64E + 11	1.81E + 11
	STD	**4.81E + 07**	3.86E + 09	1.51E + 09	1.49E + 10	7.39E + 09	5.04E + 08	1.50E + 10	1.95E + 10
F13	Degree	**1**	4	2	8	5	3	6	7
	Average	**7.10E + 04**	5.23E + 08	7.12E + 07	4.50E + 10	1.40E + 09	1.82E + 08	4.24E + 10	4.37E + 10
	STD	**4.73E + 04**	2.50E + 08	1.05E + 08	4.83E + 09	1.66E + 09	3.52E + 07	2.85E + 09	6.47E + 09
F14	Degree	**1**	6	5	8	2	3	4	7
	Average	**2.85E + 06**	1.61E + 07	7.94E + 06	1.58E + 08	5.63E + 06	6.77E + 06	7.25E + 06	6.82E + 07
	STD	**1.67E + 06**	8.90E + 06	6.77E + 06	8.70E + 07	3.97E + 06	3.45E + 06	5.97E + 06	4.56E + 07
F15	Degree	**1**	4	2	7	5	3	6	8
	Average	**3.23E + 04**	1.04E + 08	2.16E + 07	2.12E + 10	5.93E + 08	2.67E + 07	1.47E + 10	2.35E + 10
	STD	**2.50E + 04**	8.77E + 07	4.01E + 07	4.37E + 09	9.39E + 08	8.57E + 06	2.18E + 09	4.03E + 09
F16	Degree	**1**	5	2	8	4	3	6	7
	Average	**6.22E + 03**	1.56E + 04	7.35E + 03	2.49E + 04	1.02E + 04	8.56E + 03	2.09E + 04	2.43E + 04
	STD	**8.02E + 02**	2.27E + 03	8.82E + 02	2.58E + 03	1.32E + 03	5.99E + 02	1.16E + 03	3.86E + 03
F17	Degree	**1**	4	3	8	5	2	6	7
	Average	**5.98E + 03**	1.24E + 04	8.71E + 03	1.66E + 07	2.95E + 04	6.75E + 03	3.37E + 05	8.01E + 06
	STD	**6.15E + 02**	4.58E + 03	1.21E + 03	1.03E + 07	1.07E + 05	4.21E + 02	5.00E + 05	1.18E + 07
F18	Degree	**1**	5	4	8	6	3	2	7
	Average	**2.94E + 06**	1.48E + 07	1.28E + 07	2.55E + 08	1.49E + 07	1.15E + 07	8.49E + 06	1.18E + 08
	STD	**1.46E + 06**	7.73E + 06	1.01E + 07	1.39E + 08	2.91E + 07	5.97E + 06	4.30E + 06	7.96E + 07
F19	Degree	**1**	4	2	8	5	3	6	7
	Average	**4.95E + 04**	1.32E + 08	2.60E + 07	2.32E + 10	8.30E + 08	3.83E + 07	1.73E + 10	2.08E + 10
	STD	**6.93E + 04**	6.04E + 07	1.99E + 07	4.50E + 09	1.17E + 09	8.72E + 06	1.75E + 09	4.48E + 09
F20	Degree	2	5	4	8	3	**1**	6	7
	Average	5.95E + 03	7.06E + 03	6.31E + 03	7.98E + 03	6.06E + 03	**5.95E + 03**	7.65E + 03	7.77E + 03
	STD	6.53E + 02	5.90E + 02	8.94E + 02	2.98E + 02	5.61E + 02	**5.15E + 02**	3.19E + 02	3.54E + 02
F21	Degree	**1**	5	2	6	4	3	8	7
	Average	**3.23E + 03**	4.42E + 03	3.66E + 03	4.84E + 03	4.15E + 03	3.66E + 03	5.17E + 03	4.87E + 03
	STD	**1.30E + 02**	2.74E + 02	1.54E + 02	1.43E + 02	2.02E + 02	9.88E + 01	1.62E + 02	2.17E + 02

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

Table IV. *Continued*

		ECDBO	WOA	DBO	BOA	PSO	iCSPM	MDBO	GSA
F22	Degree	**1**	5	2	8	3	4	6	7
	Average	**2.47E + 04**	3.11E + 04	2.67E + 04	3.52E + 04	2.78E + 04	3.06E + 04	3.43E + 04	3.45E + 04
	STD	**4.09E + 03**	1.68E + 03	5.33E + 03	7.95E + 02	1.62E + 03	1.12E + 03	8.56E + 02	9.53E + 02
F23	Degree	**1**	4	3	6	5	2	8	7
	Average	**4.03E + 03**	5.22E + 03	4.66E + 03	6.39E + 03	5.97E + 03	4.15E + 03	8.07E + 03	6.74E + 03
	STD	**2.07E + 02**	1.97E + 02	1.87E + 02	2.71E + 02	4.42E + 02	1.17E + 02	4.95E + 02	5.99E + 02
F24	Degree	**1**	4	3	8	5	2	7	6
	Average	**4.59E + 03**	6.39E + 03	5.91E + 03	1.19E + 04	9.27E + 03	4.59E + 03	1.09E + 04	1.02E + 04
	STD	**2.72E + 02**	3.84E + 02	3.90E + 02	1.43E + 03	7.78E + 02	1.16E + 02	1.30E + 03	1.17E + 03
F25	Degree	**1**	5	3	8	4	2	6	7
	Average	**3.58E + 03**	8.08E + 03	5.39E + 03	2.98E + 04	7.50E + 03	4.94E + 03	1.70E + 04	2.71E + 04
	STD	**1.02E + 02**	8.78E + 02	5.36E + 02	1.59E + 03	8.80E + 02	2.22E + 02	1.83E + 03	2.35E + 03
F26	Degree	**1**	6	3	8	4	2	5	7
	Average	**1.86E + 04**	3.65E + 04	2.64E + 04	5.77E + 04	3.17E + 04	1.90E + 04	3.54E + 04	5.31E + 04
	STD	**5.54E + 03**	3.49E + 03	3.75E + 03	2.33E + 03	5.47E + 03	1.08E + 03	3.87E + 03	3.23E + 03
F27	Degree	**1**	4	3	8	5	2	6	7
	Average	**3.79E + 03**	5.81E + 03	4.31E + 03	1.48E + 04	7.14E + 03	3.87E + 03	1.03E + 04	1.28E + 04
	STD	**1.76E + 02**	8.45E + 02	2.85E + 02	1.23E + 03	1.72E + 03	7.87E + 01	3.59E + 03	2.32E + 03
F28	Degree	**1**	5	3	8	4	2	6	7
	Average	**3.63E + 03**	1.12E + 04	8.18E + 03	3.65E + 04	9.38E + 03	4.74E + 03	1.29E + 04	3.03E + 04
	STD	**5.22E + 01**	1.30E + 03	3.31E + 03	1.82E + 03	1.48E + 03	2.96E + 02	5.10E + 03	1.82E + 03
F29	Degree	**1**	5	2	8	4	3	6	7
	Average	**7.71E + 03**	1.75E + 04	9.70E + 03	1.01E + 06	1.53E + 04	9.95E + 03	9.17E + 04	5.52E + 05
	STD	**6.43E + 02**	2.90E + 03	1.13E + 03	4.96E + 05	5.79E + 03	6.69E + 02	4.10E + 04	6.88E + 05
F30	Degree	**1**	4	2	8	5	3	6	7
	Average	**1.26E + 06**	1.42E + 09	9.28E + 07	4.13E + 10	1.44E + 09	1.43E + 08	3.37E + 10	3.87E + 10
	STD	**1.44E + 06**	6.33E + 08	5.74E + 07	5.09E + 09	1.76E + 09	4.88E + 07	4.83E + 09	7.15E + 09

better convergence speed and accuracy in the later iterations. On functions F2, F17, F20, F24, and F26, the ECDBO algorithm and the iCSPM algorithm eventually converge with almost the same iteration speed and accuracy, ahead of the other six algorithms. On function F20, the iCSPM algorithm performs better, while on function F9, the DBO algorithm performs better.

The following results can be obtained from the above analysis:

- ECDBO algorithm achieves the best results on a larger number of functions, both in terms of mean value and convergence speed. This shows that the ECDBO algorithm has a better ability to cope with a wide variety of optimization problems, reflecting its strong competitiveness.
- Unlike the other compared algorithms, the ECDBO algorithm finds optimal solutions in multiple functions, reflecting the fact that ECDBO possesses greater adaptability to different problems.
- When confronted with complex problems, ECDBO has more powerful exploration and exploitation capabilities, and is more likely to allow the algorithm to eventually converge to a globally optimal solution.

5.2. Parameter sensitivity analysis

In order to verify the parameter sensitivity of the proposed improvement strategy in the ECDBO algorithm, the parameter k of the boundary optimization strategy and the parameter β of the Lévy flight strategy are selected for analysis in this paper. To ensure the comprehensiveness and representativeness of the experimental conclusions, the better-performing functions in the single-peak, multi-peak, hybrid and combined functions (F1, F4, F13 and F22, respectively), as well as the corresponding worse-performing functions (F3, F9, F20 and F29, respectively), were selected from the set of test functions of the CEC 2017, and the stability and luberance of the above eight functions were deeply analyzed by solving the functions with different combinations of parameters under different conditions. By solving the above eight functions under different parameter combinations, the performance of the improved algorithm in terms of stability and robustness is analyzed in depth.

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

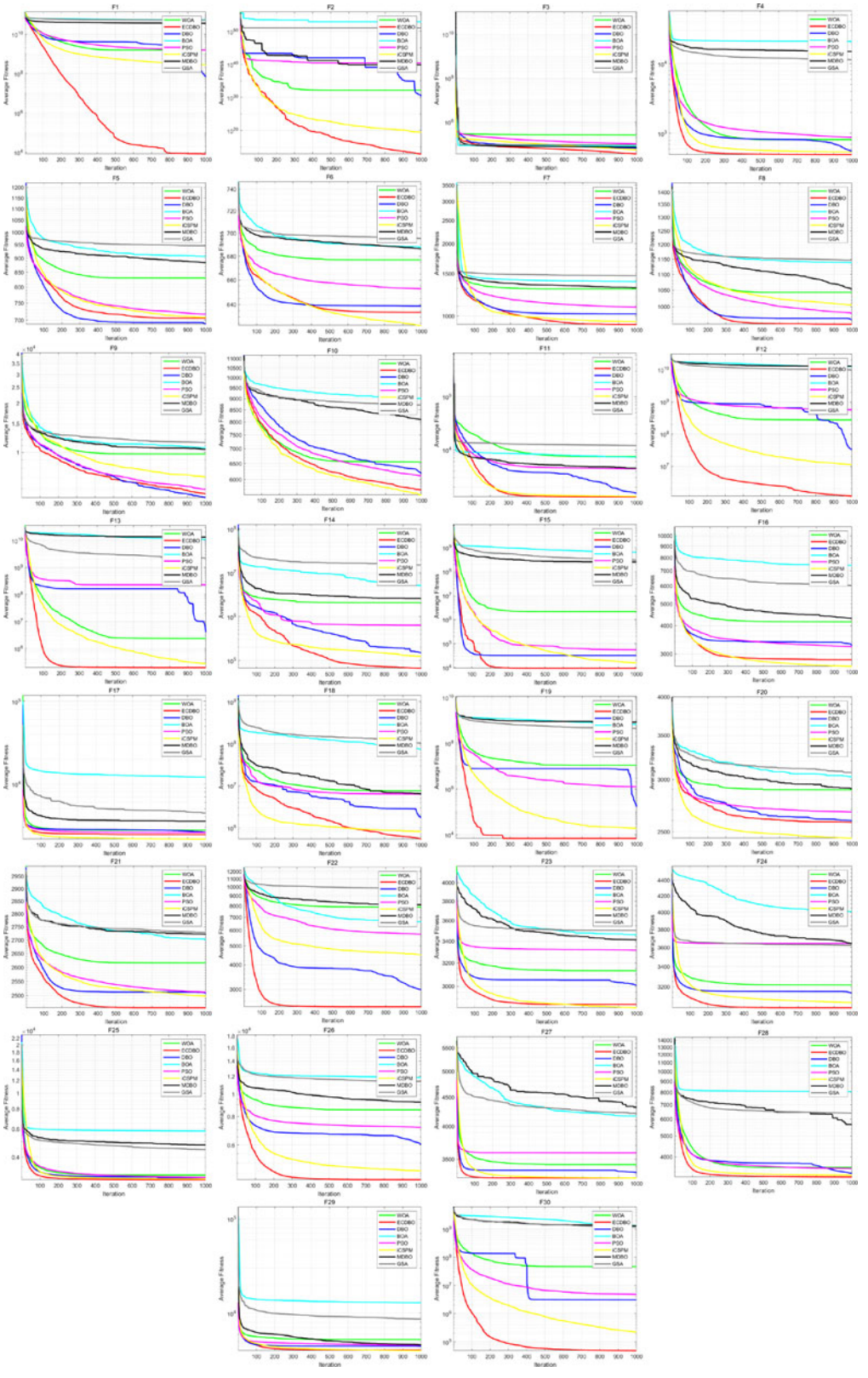


Figure 13. *30-dimensional CEC2017 convergence curve.*

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

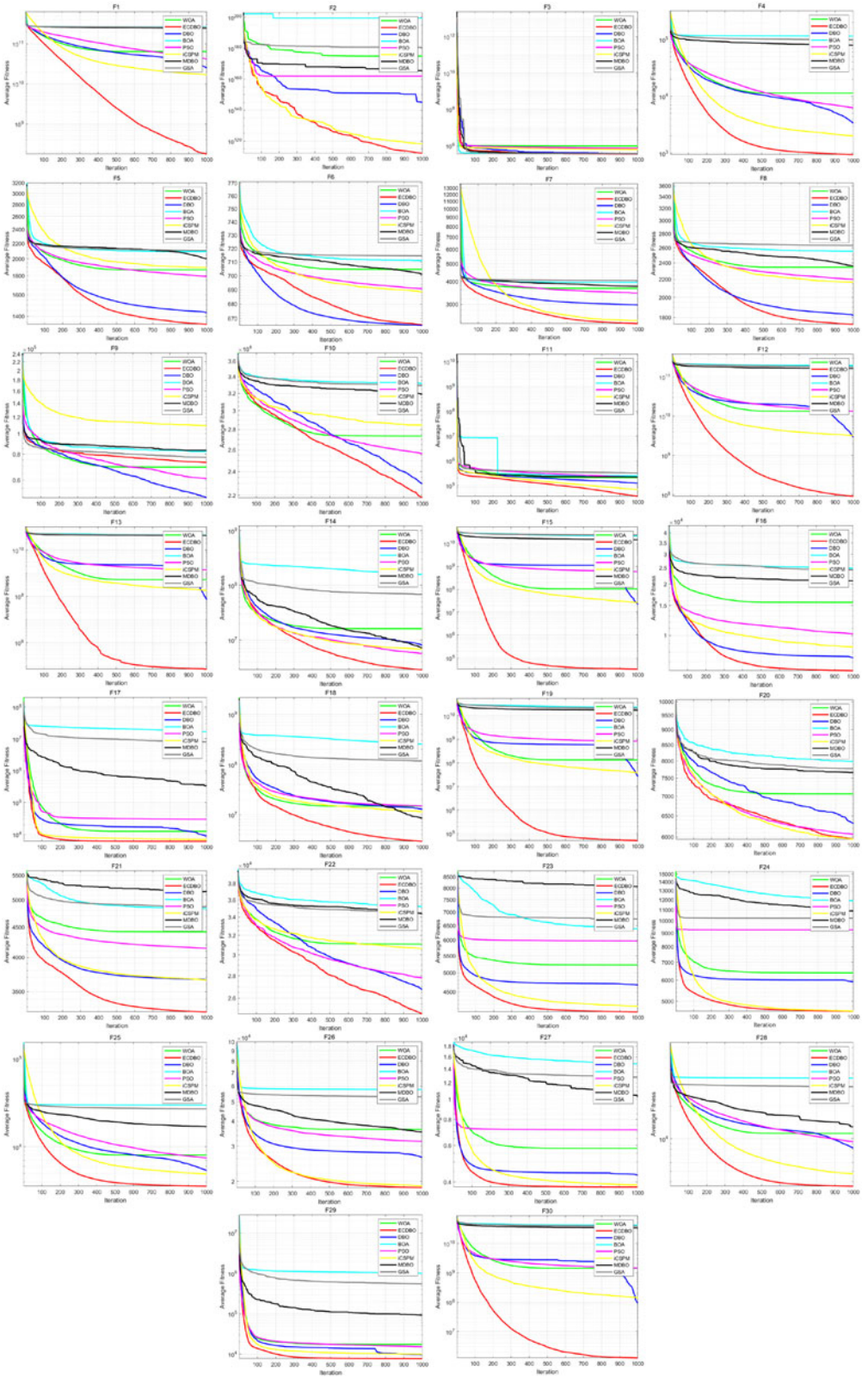


Figure 14. *100-dimensional CEC2017 convergence curve.*

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

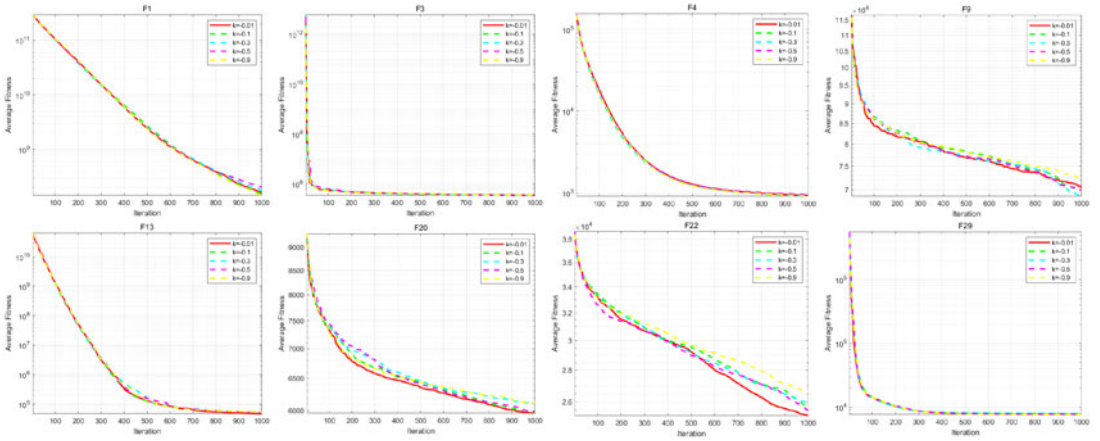


Figure 15. *Experimental results at different k values.*

To investigate the sensitivity of the proposed algorithm to the parameter k, a series of experiments was conducted using different values: $k = -0.01, -0.1, -0.3, -0.5, -0.9$. The average fitness curves across eight representative benchmark functions (F1, F3, F4, F9, F13, F20, F22, and F29) are shown in Figure 15.

For functions F1, F3, F4, F13, and F29, the algorithm exhibits consistent convergence behavior for all values of k. The algorithm maintains a stable search dynamics regardless of the change in k values. In F9 and F20, the setting $k = -0.01$ consistently outperforms others in terms of final fitness. In F22, the curves begin to diverge after 400 iterations, with $k = -0.01$ again achieving the best convergence, and $k = -0.9$ performing the worst. These results indicate that smaller-magnitude negative values of k (closer to 0) are better suited to maintaining exploitation capability in complex search spaces, whereas larger negative values may hinder convergence precision. While small performance differences appear in certain cases, especially for complex multimodal functions, the algorithm remains effective under all tested settings. This highlights the robustness and adaptability of the algorithm with respect to parameter k.

To further assess the influence of key parameters on algorithm performance, we conducted a sensitivity analysis on the parameter β, with selected values $\beta = 1.0, 1.3, 1.5, 1.7, 1.9$. The results of the average convergence curves for different settings in the benchmarking functions F1, F3, F4, F9, F13, F20, F22 and F29 are shown in Figure 16. The following insights can be drawn: In F1 and F13, $\beta = 1.5$ achieves superior convergence precision, while lower values converge more slowly and stagnate at suboptimal fitness levels. In F4, all curves show similar trends initially, but the later convergence performance is notably better with larger β values ($\beta = 1.5$ to 1.9), especially beyond 300 iterations. In F3 and F29, the curves for all β values rapidly converge and almost entirely overlap, suggesting the optimization difficulty is low and β has negligible impact in these scenarios. Although $\beta = 1.7$, 1.9 outperforms the default configuration ($\beta = 1.5$) for functions F9, F20, and F22, the differences between all functions are not significant. Therefore, $\beta = 1.5$ provides a robust balance between the different problem types and was chosen as the default configuration. The convergence behavior is significantly more sensitive to the parameter β compared to k.

5.3. Wilcoxon rank sum test

Wilcoxon rank sum test was used to test whether the ECDBO algorithm was significantly different from the other seven pairwise algorithms. Using the test results of the eight algorithmic CEC2017 functions as input samples, and tested at a significance level of 0.05. Tables IV and V demonstrate the results of the Wilcoxon rank sum test.

It is assumed that there is no significant difference between the two groups of data to be tested, and the hypothesis is rejected when $P < 0.05$ proves that there is a significant difference between the two groups

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

Table V. *30-dimensional CEC2017 Wilcoxon rank sum test.*

	WOA	DBO	BOA	PSO	iCSPM	MDBO	GSA
F1	3.02E-11 < 0.05	1.33E-10 < 0.05	4.46E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F2	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F3	3.02E-11 < 0.05	1.46E-10 < 0.05	4.08E-11 < 0.05	2.03E-09 < 0.05	5.60E-07 < 0.05	6.07E-11 < 0.05	3.02E-11 < 0.05
F4	3.02E-11 < 0.05	2.38E-07 < 0.05	3.02E-11 < 0.05	8.15E-11 < 0.05	3.81E-07 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F5	1.07E-09 < 0.05	0.137	3.02E-11 < 0.05	0.395	0.807	7.39E-11 < 0.05	3.02E-11 < 0.05
F6	5.49E-11 < 0.05	0.102	3.02E-11 < 0.05	2.88E-06 < 0.05	1.17E-03 < 0.05	4.50E-11 < 0.05	3.02E-11 < 0.05
F7	3.02E-11 < 0.05	8.84E-07 < 0.05	3.02E-11 < 0.05	4.62E-10 < 0.05	5.32E-03 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F8	1.69E-09 < 0.05	0.162	3.02E-11 < 0.05	6.97E-03 < 0.05	2.49E-06 < 0.05	9.83E-08 < 0.05	3.02E-11 < 0.05
F9	2.57E-07 < 0.05	0.610	2.61E-10 < 0.05	0.501	2.15E-02 < 0.05	3.82E-10 < 0.05	6.07E-11 < 0.05
F10	5.87E-04 < 0.05	0.0615	3.02E-11 < 0.05	0.0933	0.684	7.77E-09 < 0.05	3.02E-11 < 0.05
F11	3.02E-11 < 0.05	2.23E-09 < 0.05	3.02E-11 < 0.05	4.20E-10 < 0.05	1.06E-03 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F12	3.02E-11 < 0.05	1.78E-10 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F13	3.82E-10 < 0.05	2.57E-07 < 0.05	3.02E-11 < 0.05	5.19E-07 < 0.05	8.48E-09 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F14	8.99E-11 < 0.05	0.0519	3.02E-11 < 0.05	0.429	0.0824	3.02E-11 < 0.05	3.02E-11 < 0.05
F15	3.69E-11 < 0.05	2.13E-05 < 0.05	3.02E-11 < 0.05	1.56E-08 < 0.05	4.46E-04 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F16	5.49E-11 < 0.05	3.16E-05 < 0.05	3.02E-11 < 0.05	6.55E-04 < 0.05	2.61E-02 < 0.05	4.50E-11 < 0.05	3.02E-11 < 0.05
F17	2.13E-05 < 0.05	3.39E-02 < 0.05	3.02E-11 < 0.05	0.133	5.86E-06 < 0.05	3.69E-11 < 0.05	8.99E-11 < 0.05
F18	3.96E-08 < 0.05	0.773	4.50E-11 < 0.05	0.464	4.64E-03 < 0.05	6.55E-04 < 0.05	3.02E-11 < 0.05
F19	3.02E-11 < 0.05	3.83E-06 < 0.05	3.02E-11 < 0.05	3.65E-08 < 0.05	1.39E-06 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F20	2.00E-05 < 0.05	0.695	1.61E-10 < 0.05	0.109	4.43E-03 < 0.05	1.56E-08 < 0.05	2.61E-10 < 0.05
F21	3.02E-11 < 0.05	5.46E-06 < 0.05	3.50E-09 < 0.05	2.13E-05 < 0.05	5.97E-05 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F22	4.08E-11 < 0.05	4.20E-10 < 0.05	1.21E-10 < 0.05	8.99E-11 < 0.05	1.96E-10 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F23	3.34E-11 < 0.05	1.31E-08 < 0.05	3.02E-11 < 0.05	3.34E-11 < 0.05	0.252	3.02E-11 < 0.05	3.02E-11 < 0.05
F24	1.21E-10 < 0.05	2.03E-07 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	2.27E-03 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F25	3.02E-11 < 0.05	8.29E-06 < 0.05	3.02E-11 < 0.05	3.69E-11 < 0.05	5.09E-08 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F26	8.99E-11 < 0.05	1.53E-05 < 0.05	3.02E-11 < 0.05	1.73E-07 < 0.05	1.76E-02 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F27	6.70E-11 < 0.05	8.84E-07 < 0.05	3.02E-11 < 0.05	4.98E-11 < 0.05	0.62	9.51E-06 < 0.05	3.02E-11 < 0.05
F28	3.02E-11 < 0.05	9.92E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	1.86E-09 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F29	1.78E-10 < 0.05	5.61E-05 < 0.05	3.02E-11 < 0.05	1.25E-05 < 0.05	0.438	1.78E-04 < 0.05	3.02E-11 < 0.05
F30	3.02E-11 < 0.05	7.12E-09 < 0.05	3.02E-11 < 0.05	1.78E-10 < 0.05	1.43E-08 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05

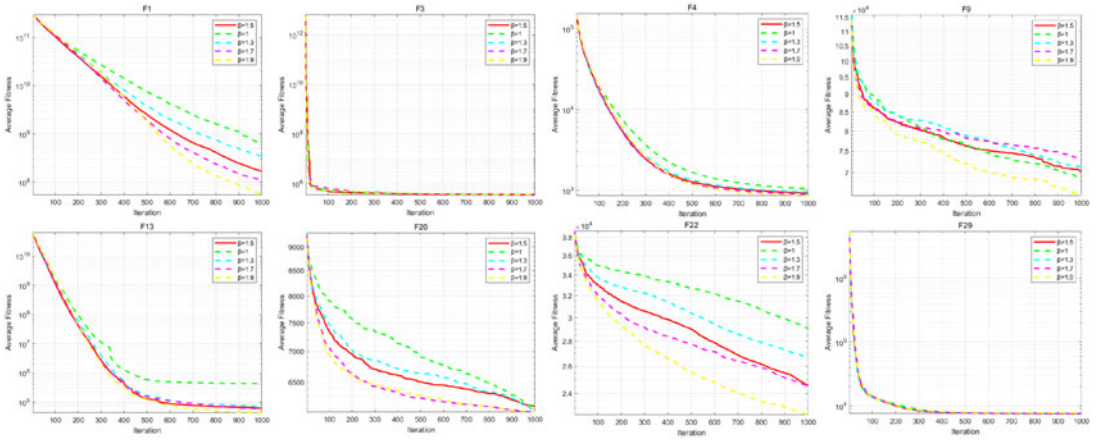


Figure 16. *Experimental results at different β values.*

of data. As can be seen from Table V and VI, the ECDBO algorithm realizes a significant performance difference in comparison with the results of the other seven algorithms, which indicates that the ECDBO algorithm outperforms the other algorithms in terms of the selected evaluation metrics, and the results are statistically significant. The next step will be to continue to explore the advantages of the algorithm in practical applications.

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

Table VI. *100-dimensional CEC2017 Wilcoxon rank sum test.*

	WOA	DBO	BOA	PSO	iCSPM	MDBO	GSA
F1	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F2	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	8.15E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F3	3.02E-11 < 0.05	0.819	5.08E-03 < 0.05	6.70E-11 < 0.05	3.34E-11 < 0.05	9.52E-04 < 0.05	5.46E-06 < 0.05
F4	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F5	3.02E-11 < 0.05	2.88E-06 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.34E-11 < 0.05	3.02E-11 < 0.05
F6	4.08E-11 < 0.05	0.888	3.02E-11 < 0.05	8.15E-11 < 0.05	2.61E-10 < 0.05	5.49E-11 < 0.05	3.02E-11 < 0.05
F7	3.02E-11 < 0.05	1.29E-09 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	0.0575	3.02E-11 < 0.05	3.02E-11 < 0.05
F8	3.02E-11 < 0.05	3.02E-03 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	5.49E-11 < 0.05	3.02E-11 < 0.05
F9	4.51E-02 < 0.05	4.42E-06 < 0.05	2.39E-08 < 0.05	1.60E-07 < 0.05	5.57E-10 < 0.05	4.31E-08 < 0.05	4.36E-02 < 0.05
F10	3.57E-06 < 0.05	0.429	3.02E-11 < 0.05	9.07E-03 < 0.05	4.62E-10 < 0.05	6.70E-11 < 0.05	3.02E-11 < 0.05
F11	3.02E-11 < 0.05	4.98E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	4.57E-09 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F12	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F13	3.02E-11 < 0.05	4.08E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F14	8.99E-11 < 0.05	4.35E-05 < 0.05	3.02E-11 < 0.05	1.60E-03 < 0.05	5.19E-07 < 0.05	4.35E-05 < 0.05	3.02E-11 < 0.05
F15	3.02E-11 < 0.05	3.34E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F16	3.02E-11 < 0.05	1.17E-05 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	8.15E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F17	3.69E-11 < 0.05	1.61E-10 < 0.05	3.02E-11 < 0.05	2.15E-10 < 0.05	3.57E-06 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F18	8.15E-11 < 0.05	1.70E-08 < 0.05	3.02E-11 < 0.05	4.98E-04 < 0.05	1.96E-10 < 0.05	2.60E-08 < 0.05	3.02E-11 < 0.05
F19	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F20	1.31E-08 < 0.05	0.149	3.69E-11 < 0.05	0.549	0.935	1.46E-10 < 0.05	5.49E-11 < 0.05
F21	3.02E-11 < 0.05	1.78E-10 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	6.07E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F22	5.46E-09 < 0.05	0.0905	3.02E-11 < 0.05	7.62E-03 < 0.05	1.01E-08 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F23	3.02E-11 < 0.05	9.92E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	4.43E-03 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F24	3.02E-11 < 0.05	3.34E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	0.888	3.02E-11 < 0.05	3.02E-11 < 0.05
F25	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F26	3.02E-11 < 0.05	2.83E-08 < 0.05	3.02E-11 < 0.05	6.12E-10 < 0.05	0.728	3.02E-11 < 0.05	3.02E-11 < 0.05
F27	3.34E-11 < 0.05	7.38E-10 < 0.05	3.02E-11 < 0.05	3.34E-11 < 0.05	5.32E-03 < 0.05	7.69E-08 < 0.05	3.02E-11 < 0.05
F28	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	6.74E-06 < 0.05	3.02E-11 < 0.05
F29	3.02E-11 < 0.05	1.55E-09 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	6.70E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05
F30	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05	3.02E-11 < 0.05

5.4. Inverse kinematics solution experiment

5.4.1. PUMA560 robotic arm experiment

To minimize the possibility of random experimental outcomes, fitness value F was used as the evaluation metric, with ECDBO, IDBO, DBO, and PSO algorithms as the subjects of study. The target pose matrix A_T was used to set up the simulation experiment. At this time, the position and posture of the PUMA560 robotic arm are as shown in Figure 17. The experiment is set with a population size of 60, a maximum iteration count of 1000, and other parameters kept at default values. By comparing the mean value and standard deviation of the fitness values, superiority of the ECDBO algorithm in solving the IK problem was validated. In this experiment, the population size was consistently set to 60, with 1000 iterations, and each algorithm was independently run 30 times. The test results are shown in Table VII, and the fitness value curves of the 30 independent runs are illustrated in Figure 18.

$$A_T = \begin{bmatrix} 0.5322 & -0.1137 & -0.8390 & 0.0833 \\ -0.4697 & -0.8641 & -0.1809 & 0.1890 \\ -0.7044 & 0.4903 & -0.5133 & -0.4925 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

As can be seen from Table VII, the adaptation value of ECDBO is significantly smaller than the other three algorithms and is ranked first. This shows that the ECDBO algorithm has higher solution accuracy and is more suitable for IK solution problems.

Figure 18 shows the fitness value F curves for 30 independent runs of the four algorithms. From the figure, it is evident that the PSO algorithm's F values fluctuate within the range of 0 to 0.6, with large amplitude variations, indicating poor stability. The F-value of the DBO algorithm fluctuates within the

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

Table VII. *Statistical comparison of fitness values.*

Algorithm	ECDBO	IDBO	DBO	PSO
Degree	1	2	3	4
Average	0.014212	0.028578	0.068387	0.249841
STD	0.00507	0.01855	0.046551	0.164579

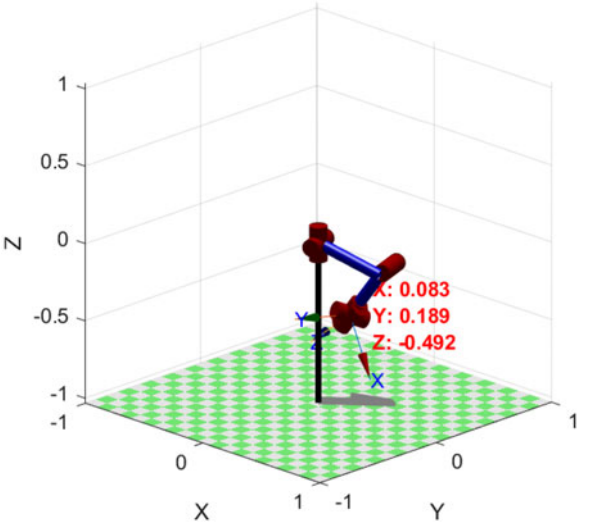


Figure 17. *Target position of robotic arm.*

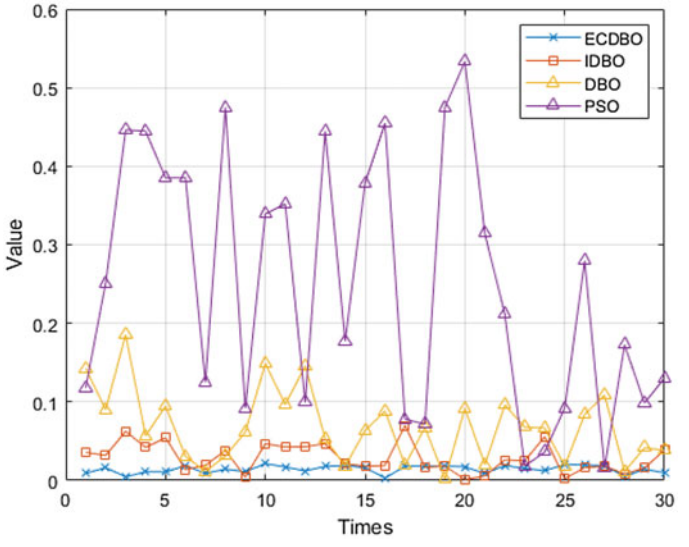


Figure 18. *Comparison of F-value.*

range of 0 to 0.2, and the amplitude variation is smaller than that of the PSO, but it is still relatively large, indicating that the stability is average. The IDBO algorithm shows fluctuations in the F value within the range of 0 to 0.1, with smaller amplitude variations than PSO and DBO, demonstrating moderate stability. In contrast, the ECDBO algorithm maintains its F values within the range of 0 to 0.022, with minimal fluctuations, reflecting strong stability.

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

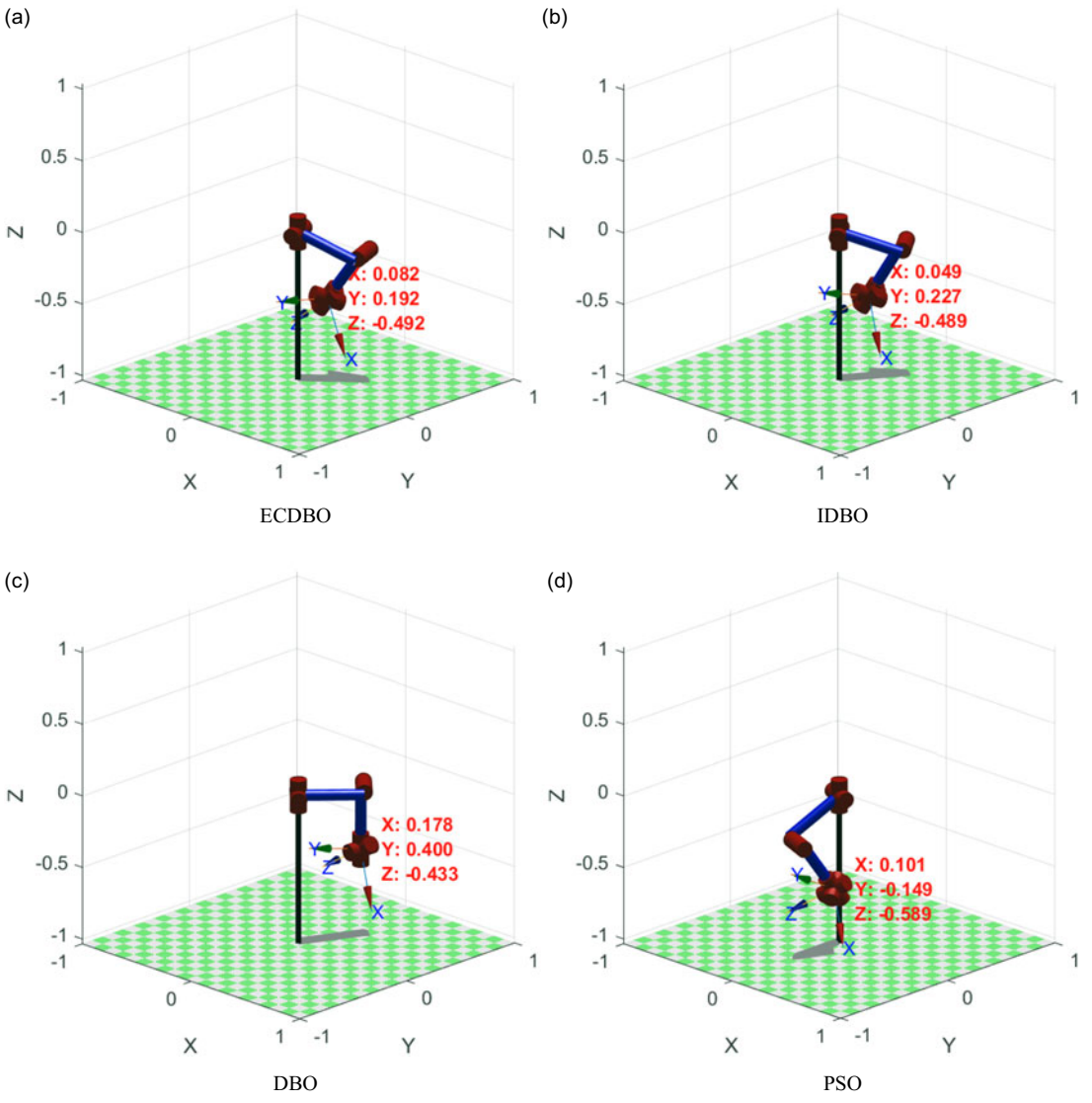


Figure 19. *Actual position of robotic arm.*

To verify the feasibility and effectiveness of the ECDBO algorithm for IK solution of robotic arms, PUMA560 robotic arm is used as a research object, A_T is the target position matrix to establish the simulation experiment. IK solution using ECDBO, IDBO, DBO and PSO algorithms, deriving six joint angles and actual pose matrix. PUMA560 robotic arm was modeled in MATLAB, and resulting positional errors were verified through the Robotics Toolbox.

The joint angles obtained by each algorithm are shown in Table VIII. Input the six joint angles obtained from different algorithms into the robot toolbox, and the actual pose of the robotic arm is shown in Figure 19.

Comparing Figures 17 and 19, it can be seen that the results sought by the ECDBO algorithm are the closest to the initially set target, both in terms of the end-effector position and the robotic arm attitude. ECDBO outperforms the other three compared algorithms in handling IK solution problems. This indicates that the improved ECDBO algorithm provides a solution much closer to the target values.

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

Table VIII. *Joint angle optimization results of three algorithms.*

Joint Angles	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
ECDBO	0.3699	0.3442	0.3466	0.3917	0.3606	0.3206
IDBO	0.6602	0.2940	0.3259	0.8427	0.5211	0.0782
DBO	−1.6432	−3.1415	−3.1415	1.2812	−1.0276	−2.5317
PSO	3.1415	2.5507	−3.1415	2.7155	0.4625	0.8821

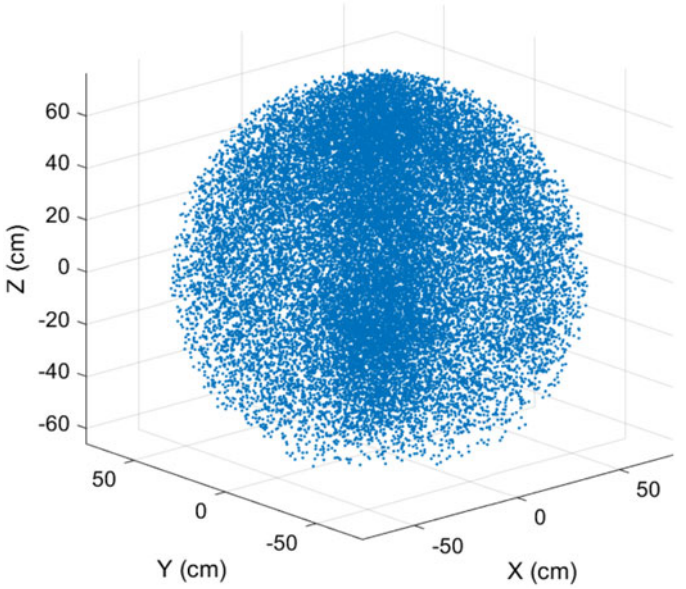


Figure 20. *Target position of robotic arm.*

5.4.2. xArm6 robotic arm experiment

To comprehensively evaluate the performance of the proposed IK solving algorithm in continuous trajectory tasks, this paper designs a path tracking experiment based on the xArm6 robotic arm.

Firstly, a point cloud map of the workspace of the xArm6 robotic arm is generated according to the xArm6 robotic arm DH parameters, as shown in Figure 20. An initial path is generated in this point cloud map with a starting point of [30,0,50] and an end point of [−10, −20,50], and the generated initial path is a smooth and continuous 3D curve covering a representative range of attitude changes in the working space of the robotic arm. To facilitate calculation and comparison, 50 points are randomly and uniformly sampled from this path as the target end position sequence for IK solution, and four algorithms are used to perform continuous IK solution and record the experimental results.

The initial population number of each algorithm is set to 60, when the fitness value reaches 0.01 and below, the condition is met, the iteration stops and the results are recorded, or stops when the maximum number of iterations reaches 1,000, at which time the preset condition is not met, the fitness value and the number of times the condition is not met are recorded and used as a comparative analysis.

Table IX demonstrates the running time and number of failures of the four algorithms. Observing the data in the table, it can be learnt that the ECDBO algorithm outperforms the IDBO and DBO algorithms in terms of solution time and is the only algorithm among the four algorithms that passes all 50 IK solutions. Although the PSO algorithm outperforms the ECDBO algorithm in terms of solution time, it also has the relatively highest number of failures, with 11 failures out of 50 solutions.

Figure 21 shows the variation of the fitness values for 50 IK solvers, in which ECDBO maintains very low and stable errors at all path points, indicating that its IK solver has the highest accuracy and is the most stable. IDBO has a smaller overall error, but large fluctuations occur at individual path points,

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

Table IX. *Execution time and failure counts of different algorithms.*

Algorithm	**ECDBO**	**IDBO**	**DBO**	**PSO**
Time	8.48s	12.79s	11.82s	5.10s
Number of Failures	0	8	5	11

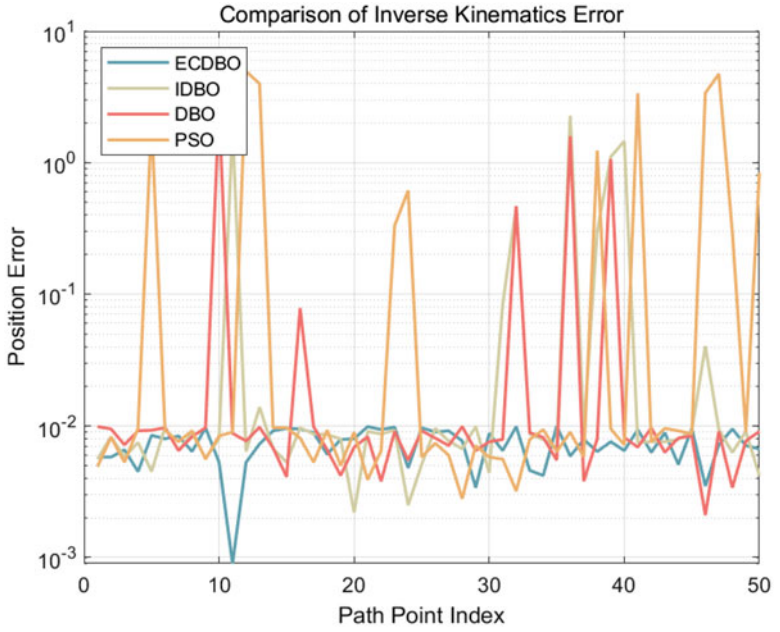


Figure 21. *Comparison of inverse kinematics error.*

showing a certain degree of instability. DBO has a good performance at most points, but there are equally some points where the DBO performs well in most points, but there are also some points with sudden increase in error, which indicates that its convergence ability is insufficient in some complex attitudes. PSO shows the greatest error variability and the poorest stability, suggesting a tendency to get trapped in local optima on complex inverse-kinematics tasks.

The Figure 22 shows the overlap between the reconstructed path (red dashed line) and the original path (blue solid line) in 3D space for each algorithm. As can be seen from the figure, the path generated by ECDBO is almost completely overlapped with the original path, with very high accuracy, and the overall trajectory is smooth and continuous, which indicates that the algorithm has strong convergence ability and global search ability. The paths generated by IDBO and DBO are also basically close to the target path, but there is a small deviation near some points, which shows a certain degree of instability. In contrast, the paths generated by PSO have obvious jitter and deviation phenomena, and the errors are large at several locations and even fail to accurately approximate the target point, which shows that it is easy to fall into the local optimum in the high-dimensional complex space.

This experiment clearly demonstrates the superior performance of the proposed ECDBO algorithm on the complex path inverse solution problem, and further proves its feasibility and practicability in practical scenarios such as industrial robot path control and attitude tracking.

In order to further verify the usability of the IK solution results in practical control, the 50 joint angle solution sequences computed by the ECDBO algorithm are sequentially imported into the solid xArm6 robotic arm. As shown in the Figure 23, in the experiment, the robotic arm moves point by point according to the inverse solution results in a set order, and finally realizes the overall trajectory tracking from the starting point of the path to the end point smoothly, and completes the path execution task stably under the condition of no external interference, with good continuity of trajectory, no obvious

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

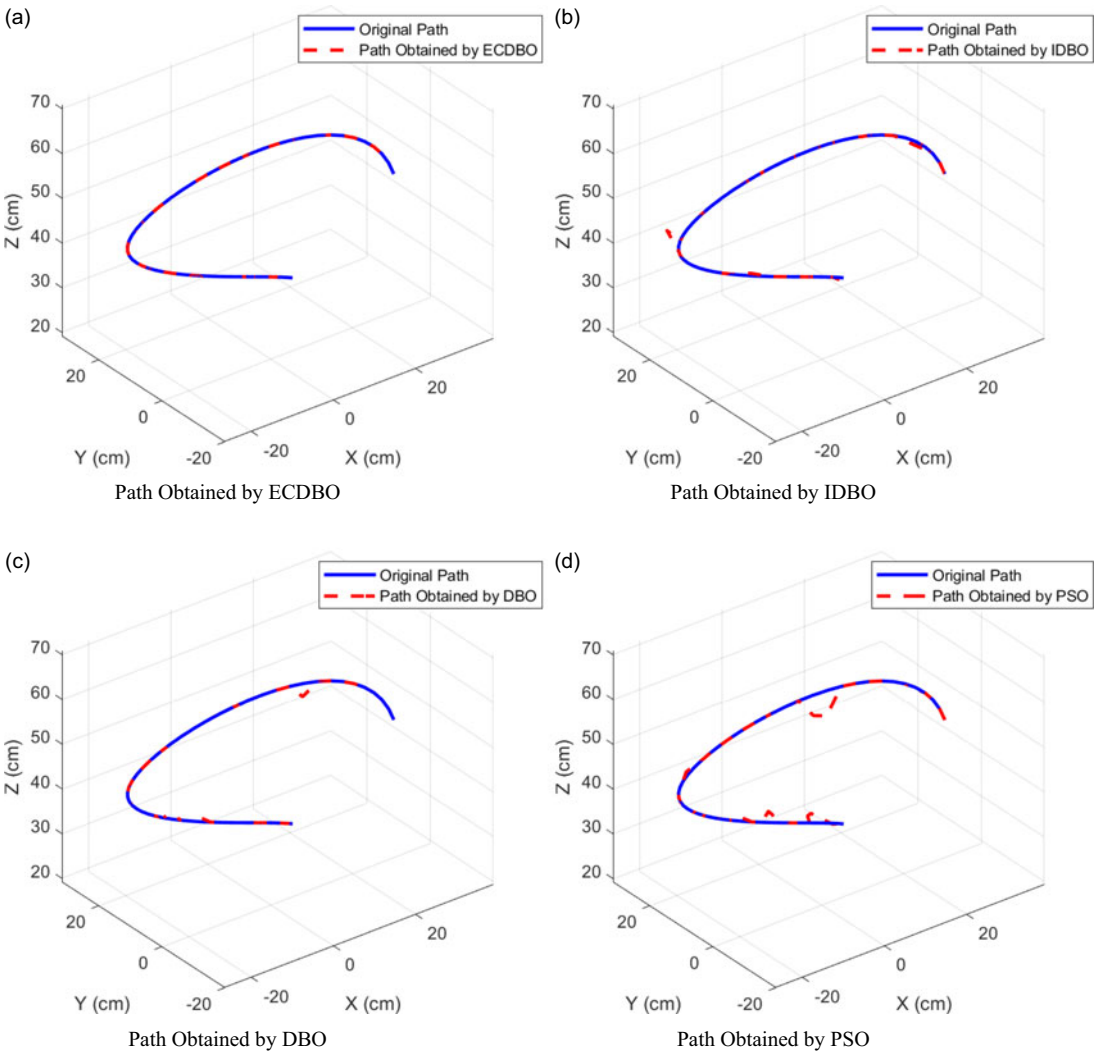


Figure 22. *Comparison of path.*

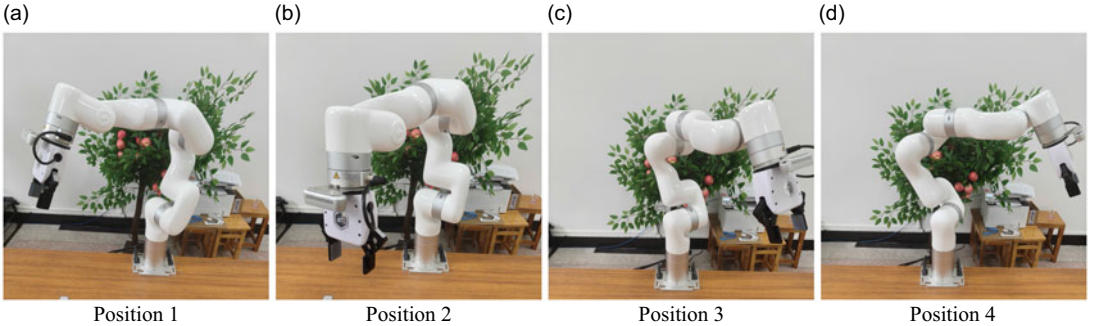


Figure 23. *Comparison of path.*

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

stutter or offset, and the execution process is smooth, which verifies its better engineering adaptability and control effect.

6. Conclusions and future works

An improved DBO algorithm, ECDBO, is proposed to address the problem of difficult IK of six-degree-of-freedom robotic arms with low solution accuracy. Improvements were made in four main areas: (1) Introducing Sobol sequences to initiate dung beetle populations to distribute individual dung beetles more evenly across the solution space and increase population diversity. (2) Propose boundary optimization strategies to improve the global exploration capability at the early stage and the local exploitation capability at the later stage, and balance the requirements for search capability at different periods. (3) Proposing a hybrid directional perturbation strategy to change the search direction of rolling dung beetles and stealing dung beetles for more detailed exploration and improved convergence accuracy. (4) Introducing a Levy flight strategy to perturb the globally optimal solution and prevent the algorithm from falling into a local optimum.

The performance of the improved DBO algorithm was validated using the CEC2017 test functions and an example of IK solving of a robotic arm. Test function experimental results evidence that the ECDBO algorithm has the advantages of fast convergence speed and high convergence accuracy. Uniform population distribution and the bounded optimization strategy enhance the global search capability to rapidly approach the optimal region and thus achieve fast convergence. Hybrid directional perturbation strategy provides more detailed search paths and ensures that the algorithm achieves a high level of accuracy at convergence. An example of solving IK of the robotic arm shows that the ECDBO algorithm can solve the nonlinear equations in the IK more efficiently, and the fitness value error has a significant decrease compared with other algorithms, and the position matrix is closer to the target position matrix, which effectively improves the position accuracy of the robotic arm.

Despite the excellent performance of the ECDBO algorithm, it has to be admitted that there are several limitations. Firstly, as can be seen in Tables III and IV, although the ECDBO algorithm performs well in the 100 -dimension test, there is still some room for improvement in the 30-dimension test. Second, the algorithm has not been directly deployed at the end of the robotic arm, and a host computer is still required to perform the algorithmic calculations. In future work, we will continue to explore the performance of the algorithm in lower dimensions and deploy it to the robotic arm end to achieve a substantial transformation from theoretical research to engineering applications.

Author contributions. YUNPENG LV: Software, Validation, Writing – original draft, Writing – review and editing. HONGBING LI: Funding, Software, Methodology, Conceptualization, Writing-review. SIQI ZHU: Data curation, Software, Visualization. SIYUN TAN: Formal analysis, Visualization. PENG YANG: Formal analysis. Chunzhe Zhao: Funding, Writing – review and editing.

Financial support. The work was supported by National Key R&D Program of China (2021YFB3901400), Natural Science Foundation of Chongqing municipality (2022NSCQ-MSX4084, CSTB2023NSCQ-BHX0089), Scientific and Technological Research Program of Chongqing Municipal Education Commission (KJZD-M202201204, KJZD-M202301203, KJQN202301247, KJQN202401212, KJQN202301215, KJQN202401226, KJQN202401237, 22SKGH333), Open Fund of Chongqing Key Laboratory of Geoenvironment Monitoring and Disaster Early Warning of Three Gorges Reservoir Area (MP2020B0202).

Competing interests. The authors declare no conflicts of interest exist.

Ethical approval. Not applicable.

References

[1] Y. Y. Luo, D. X. Li and C. Z. Fan, "An optimization method for solution selection based on the inverse kinematics problem of a mechanical arm," *Appl. Mech. Mater.* **740**, 211–217 (2015).

[2] J. Angeles, "On the numerical solution of the inverse kinematic problem," *Int. J. Robot. Res.* **4**(2), 21–37 (1985).

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press

[3] S. Uzuner, N. Akkuş and M. Toz, "5-DOF serial robot manipulator design, application and inverse kinematic solution through analytical method and simple search technique," *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi* **26**(2), 392–401 (2020).
[4] S. Xie, L. Sun, G. Chen, Z. Wang and Z. Wang, "A novel solution to the inverse kinematics problem of general 7R robots," *IEEE Access* **10**, 67451–67469 (2022).
[5] Y. Bai, M. Luo and F. Pang, "An algorithm for solving robot inverse kinematics based on FOA optimized BP neural network," *Appl. Sci.* **11**(15), 7129 (2021).
[6] S. Luo, D. Chu, Q. Li and Y. He, "Inverse kinematics solution of 6-DOF manipulator based on multi-objective full-parameter optimization PSO algorithm," *Front. Neurorobotics* **16**, 791796 (2022).
[7] M. J. Thomas, M. M. Sanjeev, A. Sudheer and M. L. J., "Comparative study of various machine learning algorithms and Denavit–Hartenberg approach for the inverse kinematic solutions in a 3- PP SS parallel manipulator," *Ind. Robot.* **47**(5), 683–695 (2020).
[8] A. Motahari, "Solving the inverse kinematics problem of discretely actuated hyper-redundant manipulators using the multi-module searching method," *Robotica* **42**(3), 911–930 (2024).
[9] D. Guo, A. Li, J. Cai, Q. Feng and Y. Shi, "Inverse kinematics of redundant manipulators with guaranteed performance," *Robotica* **40**(1), 170–190 (2022).
[10] S. Dereli and R. Köker, "Simulation based calculation of the inverse kinematics solution of 7-DOF robot manipulator using artificial bee colony algorithm," *SN Appl. Sci.* **2**(1), 27 (2020).
[11] H. Deng and C. Xie, "An improved particle swarm optimization algorithm for inverse kinematics solution of multi-DOF serial robotic manipulators," *Soft Comput.* **25**(21), 13695–13708 (2021).
[12] J. Zhong, T. Wang and L. Cheng, "Collision-free path planning for welding manipulator via hybrid algorithm of deep reinforcement learning and inverse kinematics," *Complex Intell. Syst.* **8**(3), 1899–1912 (2022).
[13] Y. Xiao, H. Cui, R. A. Khurma and P. A. Castillo, "Artificial lemming algorithm: A novel bionic meta-heuristic technique for solving real-world engineering optimization problems," *Artif. Intell. Rev.* **58**(3), 84 (2025).
[14] Y. Xiao, H. Cui, A. G. Hussien and F. A. Hashim, "MSAO: A multi-strategy boosted snow ablation optimizer for global optimization and real-world engineering applications," *Adv. Eng. Inform.* **61**(3), 102464 (2024).
[15] R. Zhong, J. Yu, C. Zhang and M. Munetomo, "SRIME: A strengthened RIME with Latin hypercube sampling and embedded distance-based selection for engineering optimization problems," *Neural Comput. Appl.* **36**(12), 6721–6740 (2024).
[16] R. Zhong, Q. Fan, C. Zhang and J. Yu, "Hybrid remora crayfish optimization for engineering and wireless sensor network coverage optimization," *Cluster Comput.* **27**(7), 10141–10168 (2024).
[17] J. Xue and B. Shen, "Dung beetle optimizer: A new meta-heuristic algorithm for global optimization," *J. Supercomput.* **79**(7), 7305–7336 (2023).
[18] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.* **1**(1), 67–82 (1997).
[19] F. Zhu, G. Li, H. Tang, Y. Li, X. Lv and X. Wang, "Dung beetle optimization algorithm based on quantum computing and multi-strategy fusion for solving engineering problems," *Expert Syst. Appl.* **236**(2), 121219 (2024).
[20] R. Zhang and Y. Zhu, "Predicting the mechanical properties of heat-treated woods using optimization-algorithm-based BPNN," *Forests* **14**(5), 935 (2023).
[21] L. Li, L. Liu, Y. Shao, X. Zhang, Y. Chen, C. Guo and H. Nian, "Enhancing swarm intelligence for obstacle avoidance with multi-strategy and improved dung beetle optimization algorithm in mobile robot navigation," *Electronics* **12**(21), 4462 (2023).
[22] W. Zilong and S. Peng, "A multi-strategy dung beetle optimization algorithm for optimizing constrained engineering problems," *IEEE Access* **11**, 98805–98817 (2023).
[23] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.* **95**(5), 51–67 (2016).
[24] S. Arora and S. Singh, "Butterfly optimization algorithm: A novel approach for global optimization," *Soft Comput.* **23**(3), 715–734 (2019).
[25] 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.
[26] B. H. Abed-alguni, "Island-based cuckoo search with highly disruptive polynomial mutation," *Int. J. Artif. Intell.* **17**(1), 57–82 (2019).
[27] E. Tanyildizi and G. Demir, "Golden sine algorithm: A novel math-inspired algorithm," *Adv. Electr. Comput. Eng.* **17**(2), 71–79 (2017).

Cite this article: Y. Lv, H. Li, S. Zhu, S. Tan, P. Yang and C. Zhao (2025). "Improved dung beetle optimization algorithm based inverse kinematics solution for robotic arm", Robotica **43**, 3488–3518. https://doi.org/10.1017/S0263574725102208

https://doi.org/10.1017/S0263574725102208 Published online by Cambridge University Press