sensors

MDPI

Article

Robotic Arm Trajectory Planning for Tunnel Lighting Cleaning Based on the CAW-PSO Algorithm

Zhibin Yao [1], Taibo Song [1], Hui Li [1], Hongwei Zhang [2] and Zhanlong Li [2,3,*]

[1] School of Mechanical Engineering, Taiyuan University of Science and Technology, Taiyuan 030024, China; yzb@tyust.edu.cn (Z.Y.); songtaibo@tyust.edu.cn (T.S.); s202312210108@tyust.edu.cn (H.L.)
[2] Shanxi Provincial Intelligent Transportation Laboratory Co., Ltd., Taiyuan 030032, China; 2021260702@mail.nwpu.edu.cn
[3] School of Vehicle and Transportation Engineering, Taiyuan University of Science and Technology, Taiyuan 030024, China
* Correspondence: lizl@tyust.edu.cn

Abstract

Tunnel lighting cleaning is of significant practical importance for improving driving safety. To address the low operational efficiency of tunnel lighting cleaning tasks, a trajectory planning method based on the chaotic adaptive whale–particle swarm optimization (CAW-PSO) algorithm is proposed. Taking the SIASUN GCR16-2000 robotic arm as the research object, the trajectory is constructed using a 3-5-3 polynomial interpolation, with the objective of achieving time-optimal trajectory planning. In the CAW-PSO algorithm, a tent chaotic map is introduced to improve the quality of the population; a linearly decreasing inertia weight is designed to strike a balance between local and global search; dynamic learning factors are defined to strengthen the individual learning ability and global cognitive capability of particles; finally, the exploitation mechanism of the whale optimization algorithm is employed to avoid getting trapped in local optima and improve convergence accuracy. The simulation time is 3.661 s, a reduction of 69.94%. The experimental results yielded a mean relative error of 1.16%, indicating good agreement with the simulation results. The results of the simulation and experiment indicate that the CAW-PSO effectively reduces the motion time of the robotic arm, exhibiting superior applicability in trajectory planning for tunnel lighting cleaning robotic arms.

Keywords: chaotic mapping; trajectory planning; dynamic learning factor; robotic arm; whale optimization algorithm; inertia weight

Academic Editor: Enrico Meli

Received: 9 February 2026
Revised: 4 March 2026
Accepted: 4 March 2026
Published: 9 March 2026

1. Introduction

With the development of robotics, its implementation in hazardous and complex environments has become feasible, such as glass façade cleaning [1], fire emergency rescue [2], and material handling [3]. Cleaning tunnel luminaires can improve their illumination performance, thereby enhancing driving safety. In tunnel luminaire cleaning, replacing manual operations with a robotic manipulator can significantly improve operational efficiency.

In tunnel lighting cleaning, the core objective of using robotic arms is to identify a motion trajectory that satisfies the cleaning task requirements [4]. The robotic manipulator needs to follow a predefined trajectory to perform the cleaning task, ensuring effective cleaning of the luminaire surface. By employing trajectory planning methods, the motion trajectory of the robotic manipulator is optimized to generate a time-optimal trajectory and

reduce the manipulator's motion time. It is important for enhancing cleaning efficiency to implement time-optimal trajectory planning.

The number of lighting fixtures in tunnels is large, and they are densely distributed. Owing to variations in installation height and orientation, their spatial arrangement is highly complex. Under such conditions, the performance of existing trajectory planning algorithms is limited, and even their improved variants cannot be directly applied. To address the challenges of trajectory planning in tunnel environments, it is therefore necessary to develop targeted improvements to existing algorithms.

In the complex tunnel environment, the cleaning operation requires a relatively long execution time. To address this challenge, a chaotic adaptive whale–particle swarm optimization (CAW-PSO) algorithm is proposed. The developed algorithm integrates 3-5-3 polynomial interpolation to achieve time-optimal trajectory planning for a robotic manipulator performing tunnel lighting cleaning tasks. The main innovations of the presented algorithm are outlined below: a tent chaotic map is introduced to generate a more uniformly distributed initial particle population, thereby improving population diversity and overall solution quality; a linearly decreasing inertia weight strategy is designed to strengthen global exploration in the early stages and local exploitation in the later stages; dynamic learning factors are defined to enhance individual learning during the early stages of the algorithm and global cognition in the later stages; finally, the optimization mechanism of the whale optimization algorithm is employed to escape local optima and enhance convergence accuracy.

The remainder of this paper is organized as follows: Section 2 provides an overview of related work, discusses their shortcomings, and introduces the improvements proposed in this study. Section 3.1 introduces the modeling method of the robotic manipulator. Section 3.2 presents the principle of the 3-5-3 polynomial interpolation. Section 3.3 describes the particle swarm optimization (PSO) algorithm. Section 3.4 provides a detailed description of the improved PSO algorithm. Section 3.5 illustrates the workflow of the CAW-PSO algorithm. In Section 4.1, the performance of the CAW-PSO algorithm is assessed using the CEC2017 test functions. Section 4.2 conducts simulation analysis by MATLAB (Version R2023a). Section 4.3 conducts real-world experiments. Section 5 summarizes the paper and outlines future research directions.

2. State of the Art

Particle swarm optimization (PSO) [5] has many advantages, such as broad adaptability and strong global search capability. However, PSO also has certain limitations, such as low adaptability and a tendency to become trapped in local optima. In practical applications, numerous researchers have proposed various improvements to address the shortcomings of the PSO algorithm.

Chen et al. [6] designed a novel migration mechanism into the particle swarm algorithm and proposed a diversity-migration-based quantum particle swarm optimization method (DM-QPSO), effectively addressing the decline in search performance and becoming trapped in a local optimum in high-dimensional optimization problems. The algorithm was optimized solely for benchmark test functions and did not take into account the kinematic and dynamic constraints of the robotic arm. Luo et al. [7] defined an improved Lévy chaotic particle swarm optimization clustering routing protocol (LCPSO-CRP), significantly enhancing convergence speed and search efficiency, thereby effectively extending the operational lifespan of industrial IoT systems. The direct application of the adopted fitness function may lead to discontinuities in the motion trajectory; therefore, the optimization objective needed to be redesigned. Wang et al. [8] introduced an adaptive multi-objective particle swarm optimization algorithm (ADMOPSO), which solves the

challenge of determining the global best particle during the process of multi-objective optimization. ADMOPSO focused on the optimization of discrete variables without addressing temporal continuity. Mebarka et al. [9] put forward an adaptive particle swarm optimization algorithm for t-SNE (t-SNE-PSO) that optimizes t-SNE by dynamically updating cognitive and social coefficients, overcoming the limitations of stochastic gradient descent and enhancing global optimization capability. The t-SNE-PSO was designed to optimize static datasets and lacked adaptive updating mechanisms for dynamic trajectory planning. Jiao et al. [10] proposed a learning-based adaptive chaotic particle swarm optimization (LCPSO) algorithm to escape from local optima and enhance the speed of convergence. In LCPSO, the reward–penalty function was difficult to enforce strictly, and tuning the penalty factor was challenging. Ahmad et al. [11] presented a novel adaptive inertia weight method based on population success rate feedback, which dynamically modifies the state of the particle in the search space. To improve the performance of the algorithm, Bilal et al. [12] introduced a chaotic sequence during each random number generation and designed a parameter-adaptive particle swarm optimization algorithm based on chaotic mapping. The improved method proposed by Ahmad and Bilal was unable to adapt to dynamically changing trajectory planning tasks and exhibited low optimization efficiency. Pan et al. [13] designed an improved hybrid algorithm (IGSAPSO) that combines gravitational search and particle swarm optimization, optimizing the charging and discharging power allocation of electric vehicles. IGSAPSO had a relatively large parameter space, and its direct application to trajectory planning required extensive parameter tuning, increasing the difficulty of engineering implementation. Sun et al. [14] developed a hybrid chaotic evolutionary particle swarm optimization algorithm (HCEPSO) that integrates evolutionary strategies with chaotic optimization mechanisms into particle swarm optimization, effectively addressing the two-dimensional bin packing problem involving conflicts and load balancing in the logistics domain. The integer encoding adopted by HCEPSO did not represent continuous time-trajectory parameters, rendering the optimized trajectory infeasible for execution by the robotic manipulator. Shu et al. [15] proposed a multi-strategy fusion enhanced particle swarm optimization algorithm (MSFPSO), which incorporates a Cauchy variant-based migration mechanism to improve search efficiency and effectiveness, employs a combined opposition-based selection strategy to broaden the exploration of the solution space, and introduces an attract–reject optimization strategy to further balance exploitation and exploration, effectively preventing algorithm stagnation. The CEC benchmark functions tested by MSFPSO were mostly unconstrained or simply bounded mathematical functions, and the 50 engineering problems concentrated on static parameter optimization. Its performance in dynamic and high-dimensional complex optimization scenarios remains to be further evaluated. Yan et al. [16] combined Lévy flight quantum particle swarm optimization with quantum particle swarm optimization to put forward an enhanced Lévy flight quantum particle swarm optimization algorithm (ELQPSO), effectively mitigating the adverse effects of limited sensor placement on overall optimization performance. The discrete optimization characteristics of ELQPSO did not satisfy the requirements of continuous linear trajectory planning. Wang et al. [17] defined a collaborative knowledge-transfer-based multi-objective multi-task particle swarm optimization algorithm (CKT-MMPSO). This method introduces the CKT-MMPSO framework, establishes a dual-space knowledge inference mechanism, and integrates an entropy-based collaborative knowledge transfer strategy, effectively addressing the issue of the EMTO algorithm neglecting latent correlations in the objective space during optimization. The multi-task framework of CKT-MMPSO was redundant in single-task trajectory planning, and its knowledge transfer mechanism was not effectively utilized, increasing computational complexity. Tao et al. [18] presented a multi-strategy improved particle swarm optimization algorithm (MSIPSO), resolving problems of path

planning for autonomous drone delivery. In trajectory planning, the strong randomness of the local deadlock escape strategy in the algorithm led to blind particle search, reducing convergence efficiency. Xu et al. [19] designed a hybrid Gaussian quantum particle swarm optimization and adaptive genetic algorithm (HGQPSO-AGA), addressing the scheduling problem. Li et al. [20] developed a novel particle swarm optimization algorithm (PSO-PE), overcoming dynamic optimization problems. The optimization objective of PSO-PE only considers the optimal solution and does not take trajectory smoothness into account.

Numerous researchers have addressed the algorithm's shortcomings—namely, low convergence accuracy and susceptibility to local optima—by tuning algorithm parameters, introducing new mechanisms, or hybridizing it with other algorithms. The improved algorithms were evaluated using benchmark functions, and their effectiveness was successfully validated. The improved algorithms effectively addressed the engineering problems they encountered. The improved PSO algorithms proposed by researchers mainly focus on functions with simple or unconstrained boundaries, without considering optimization problems involving complex constraints. The engineering problems they addressed were primarily discrete problems with simple constraints, and the proposed improved algorithms are incapable of solving continuous trajectory planning problems under complex constraints.

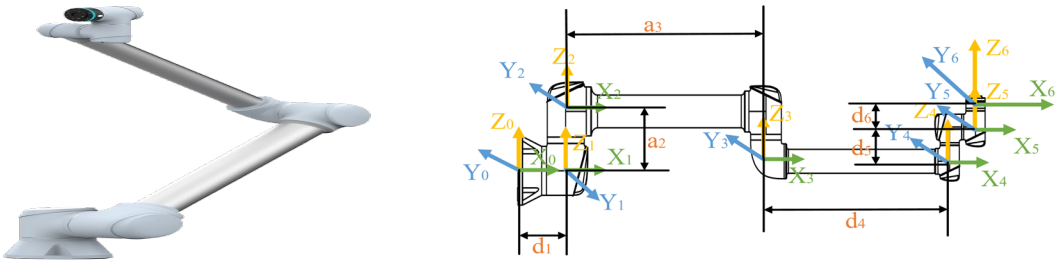
To address the continuous trajectory planning problem in complex operating environments, a chaotic adaptive whale–particle swarm optimization (CAW-PSO) algorithm is proposed. Initially, tent chaotic mapping is utilized for population initialization, thereby significantly improving the ergodicity and diversity of the particle distribution. Subsequently, a linearly decreasing inertia weight is adopted to balance local search and global search. Sinusoidal dynamic learning factors enhance the particles' individual learning capability and global cognitive capability. The synergistic effect of the two effectively balances convergence accuracy and convergence speed. Finally, the search mechanism of the whale optimization algorithm is integrated, and the particle velocity update strategy is improved by incorporating its spiral updating and random search strategies to escape local optima.

3. Materials and Methods

The standard Denavit–Hartenberg (D-H) method is adopted for modeling. The 3-5-3 polynomial interpolation is used for trajectory construction, and the polynomial coefficients are solved. A detailed description of the improved PSO algorithm is provided, and the flow of the proposed algorithm is presented.

3.1. Manipulator Modeling

This study utilizes the SIASUN GCR16-2000 (SIASUN Robot & Automation Co., Ltd., Shenyang, China) collaborative robot as the research platform. The robot comprises six links and six joints. The physical structure is shown in Figure 1a, and the coordinate system established for the links is shown in Figure 1b. The selected collaborative robot is modeled using the standard Denavit–Hartenberg (D-H) method. Table 1 lists the corresponding D-H parameters.



(**a**) (**b**)

Figure 1. SIASUN GCR16-2000 collaborative robot: (**a**) Physical structure of the robotics; (**b**) Coordinate system diagram of the robotic arm linkage.

Table 1. Standard D-H parameters of the SIASUN GCR16-2000.

p	θ_p/rad	d_p/mm	α_p/mm	α_p/°
1	θ_1	235	0	90
2	θ_2	0	217.5	−180
3	θ_3	0	977.5	180
4	θ_4	896.5	0	−90
5	θ_5	126	0	90
6	θ_6	113	0	0

In Table 1, the joint angle is denoted by θ, the joint offset is represented by d, the link length is indicated by a, and the link twist angle is represented by α. According to the known D-H parameter, the transformation matrix from coordinate frame $\{p-1\}$ to frame $\{p\}$ is derived as shown in Equation (1). After multiplying the transformation matrix of each joint, the overall transformation matrix from the base coordinate frame to the end-effector frame is acquired, as shown in Equation (2).

$$ {}^{p-1}_{p}S = \begin{bmatrix} \cos\theta_p & -\sin\theta_p & 0 & a_{p-1} \\ \sin\theta_p\cos\alpha_{p-1} & \cos\theta_p\cos\alpha_{p-1} & -\sin\alpha_{p-1} & -d_p\sin\alpha_{p-1} \\ \sin\theta_p\sin\alpha_{p-1} & \cos\theta_p\sin\alpha_{p-1} & \cos\alpha_{p-1} & d_p\cos\alpha_{p-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

$$ {}^{0}_{6}S = {}^{0}_{1}S^{1}_{2}S^{2}_{3}S^{3}_{4}S^{4}_{5}S^{5}_{6}S = \begin{bmatrix} m_x & r_x & f_x & h_x \\ m_y & r_y & f_y & h_y \\ m_z & r_z & f_z & h_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

where m_x, m_y, and m_z represent the direction vector components of the x-axis of the robot arm end-effector coordinate system relative to the fixed coordinate frame; r_x, r_y, and r_z are the direction vector components of the y-axis of the robot arm end-effector coordinate system relative to the fixed coordinate frame; f_x, f_y, and f_z are the direction vector components of the z-axis of the robot arm end-effector coordinate system relative to the fixed coordinate frame; h_x, h_y, and h_z are the direction vector components of the robot arm end-effector relative to the fixed coordinate frame.

Kinematic analysis is the foundation of manipulator trajectory planning, mainly consisting of inverse and forward kinematics. Robot motion problems are primarily addressed using inverse kinematics, with solutions typically obtained through analytical methods. The solution process is described as follows.

The first motion joint is obtained by Equation (3).

$$ \theta_1 = a\tan\left(2(h_y - l_5 f_y, h_x - l_5 f_x)\right) \tag{3} $$

The second joint is obtained from Equation (4).

$$\theta_2 = a\tan 2(U_2, \pm\sqrt{1-U_2^2}) - \eta_2 \tag{4}$$

with

$$U_2 = \frac{l_2^2 - l_3^2 - l_4^2 + \left(\cos\theta_1(h_x - l_5 f_x) + \sin\theta_1(h_y - l_5 f_y)\right)^2 + (p_z - l_1 - l_5 a_z)^2}{2l_2} \tag{5}$$

$$\eta_2 = a\tan 2(h_z - l_1 - l_5 f_z, \cos\theta_1(h_x - l_5 f_x) + \sin\theta_1(h_y - l_5 f_y)) \tag{6}$$

The third joint is determined by Equation (7).

$$\begin{aligned}\theta_3 = \; & a\tan 2(\cos\theta_1(h_x - l_5 f_x) + \sin\theta_1(h_y - l_5 f_y) - l_2\sin\theta_2, h_z - l_1 - l_5 f_z - l_2\cos\theta_2) \\ & -\theta_2 - \eta_3\end{aligned} \tag{7}$$

with

$$\eta_3 = a\tan 2(l_4, l_3) \tag{8}$$

The fourth joint variable is derived from Equation (9).

$$\theta_4 = a\tan 2(U_{41}, U_{42}) \tag{9}$$

with

$$U_{41} = \frac{-(f_x\cos\theta_1 + f_y\sin\theta_1)\sin(\theta_2+\theta_3) - f_z\cos(\theta_2+\theta_3)}{\sin\theta_5}, \tag{10}$$

$$U_{42} = \frac{f_y\cos\theta_1 - f_x\sin\theta_1}{\sin\theta_5} \tag{11}$$

The fifth joint is obtained from Equation (12).

$$\theta_5 = a\tan 2(\pm\sqrt{1-U_5^2}, U_5) \tag{12}$$

with

$$U_5 = (f_x\cos\theta_1 + f_y\sin\theta_1)\cos(\theta_2+\theta_3) - f_z\sin(\theta_2+\theta_3) \tag{13}$$

The sixth motion joint is calculated according to Equation (14).

$$\theta_6 = a\tan 2(U_{62}, U_{61}) \tag{14}$$

with

$$U_{61} = \frac{-(m_x\cos\theta_1 + m_y\sin\theta_1)\cos(\theta_2+\theta_3) + m_z\sin(\theta_2+\theta_3)}{\sin\theta_5} \tag{15}$$

$$U_{62} = \frac{(r_x\cos\theta_1 + r_y\sin\theta_1)\cos(\theta_2+\theta_3) - r_z\sin(\theta_2+\theta_3)}{\sin\theta_5} \tag{16}$$

where l_1 = 235 mm, l_2 = 977.5 mm, l_3 = 896.5 mm, l_4 = 126 mm, l_5 = 239 mm.

3.2. 3-5-3 Polynomial Interpolation

To achieve smooth motion trajectories, in this study, 3-5-3 polynomial interpolation is employed for trajectory construction, as expressed in Equation (17).

$$\begin{cases} \theta_{p1}(t) = b_{10} + b_{11}t + b_{12}t^2 + b_{13}t^3 \\ \theta_{p2}(t) = b_{20} + b_{21}t + b_{22}t^2 + b_{23}t^3 + b_{24}t^4 + b_{25}t^5 \\ \theta_{p3}(t) = b_{30} + b_{31}t + b_{32}t^2 + b_{33}t^3 \end{cases} \tag{17}$$

where $\theta_{p1}(t)$, $\theta_{p2}(t)$, and $\theta_{p3}(t)$ represent the angular displacements of the p-th joint during the first, second, and third trajectory segments, respectively.

Constraints on Equation (17) are imposed through Equations (18)–(20).

$$\begin{cases} \theta_{p1}(0) = w_{p0} \\ \theta_{p1}(t_1) = \theta_{p2}(0) = w_{p1} \\ \theta_{p2}(t_2) = \theta_{p3}(1) = w_{p2} \\ \theta_{p3}(t_3) = w_{p3} \end{cases} \tag{18}$$

$$\begin{cases} \dot{\theta}_{p1}(0) = 0 \\ \dot{\theta}_{p1}(t_1) = \dot{\theta}_{p1}(0) \\ \dot{\theta}_{p2}(t_2) = \dot{\theta}_{p2}(0) \\ \dot{\theta}_{p3}(t_3) = 0 \end{cases} \tag{19}$$

$$\begin{cases} \ddot{\theta}_{p1}(0) = 0 \\ \ddot{\theta}_{p1}(t_1) = \ddot{\theta}_{p1}(0) \\ \ddot{\theta}_{p2}(t_2) = \ddot{\theta}_{p2}(0) \\ \ddot{\theta}_{p3}(t_3) = 0 \end{cases} \tag{20}$$

In Equations (4)–(6), $\dot{\theta}_{p1}$, $\dot{\theta}_{p2}$, $\dot{\theta}_{p3}$, $\ddot{\theta}_{p1}$, $\ddot{\theta}_{p2}$, and $\ddot{\theta}_{p3}$ stand for the angular velocities and angular accelerations of the p-th joint of robots during the first, second, and third trajectory segments, respectively.

The angle constraints ensure positional continuity at the junctions of adjacent trajectory segments. The angular velocity constraints guarantee velocity continuity at the segment connections, resulting in smooth motion. The angular acceleration constraints ensure smooth variation of force at the segment junctions.

Based on the above constraints, matrices A (Equation (21)) and B (Equation (22)) can be obtained.

$$A = \begin{bmatrix} t_1^3 & t_1^2 & t_1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 3t_1^2 & 2t_1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 6t_1 & 2 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & t_2^5 & t_2^4 & t_2^3 & t_2^2 & t_2 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 5t_2^4 & 4t_2^3 & 3t_2^2 & 2t_2 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 20t_2^3 & 12t_2^2 & 6t_2 & 2 & 0 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_3^3 & t_3^2 & t_3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3t_3^2 & 2t_3 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6t_3 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{21}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & x_3 & 0 & 0 & x_0 & 0 & 0 & x_2 & x_1 \end{bmatrix} \tag{22}$$

where t_1, t_2, and t_3 denote the times of the three interpolation trajectory segments, respectively. x represents the interpolated displacement.

The coefficient b of the 3-5-3 polynomial interpolation is solved by Equation (23).

$$b = A^{-1} \cdot B \tag{23}$$

3.3. Particle Swarm Optimization Algorithm

Inspired by the predatory behavior of birds, the particle swarm optimization (PSO) algorithm was proposed to resolve complex optimization problems. In the PSO algorithm, when a bird finds an approximate location of the food, it is analogous to a particle approaching the optimal solution. The particles then communicate this information to other members, leading the entire swarm to converge gradually towards the target position. In PSO-based trajectory planning, the particle position represents the sequence of trajectory points of the robotic arm, and the particle velocity determines the direction and magnitude of trajectory updates. The optimal trajectory is continuously searched through the fitness function.

The population size is M, and the dimensionality of the search space is E. Particles update their velocities and positions according to Equations (24) and (25), respectively.

$$V_m(n+1) = \tau V_m(n) + (Pb_{best}(n) - L_m(n))\lambda_1 r_1 + (Gb_{best}(n) - L_m(n))\lambda_2 r_2 \tag{24}$$

$$L_m(n+1) = L_m(n) + V_m(n+1) \tag{25}$$

where $Pb_{best}(n)$ means individual historical best positions; $Gb_{best}(n)$ stands for global best positions; τ denotes the inertia weight; the learning factor is denoted by λ_1 and λ_2; random numbers are denoted by r_1 and r_2, each ranging from [0, 1].

The particle velocity update in Equation (24) consists of three components: the first term is the inertia component, which enables the particle to retain its previous velocity and reflects its "memory" characteristic; the second term is the cognitive component, which reflects the search behavior of particle based on its own historical best position; the third term is the social component, which captures the information exchange and cooperative interaction among particles, thereby guiding them toward the global optimal solution.

3.4. The CAW-PSO

The optimization fitness function of the p-th joint, together with the velocity and acceleration constraint conditions, is given in Equation (26).

$$\begin{cases} f(t) = min \sum_{t=0}^{n} (t_{p1} + t_{p2} + t_{p3}) \\ max\{|v_{pj}|\} \le v_{max} \\ max\{|a_{pj}|\} \le a_{max} \end{cases} \tag{26}$$

where the maximum velocity of robotics is denoted by v_{max}; a_{max} stands for the maximum acceleration of robotics; v_{pj} represents the velocity of the p-th joint in the j-th polynomial segment; a_{pj} denotes the acceleration of the p-th joint in the j-th polynomial segment; t_{p1}, t_{p2}, and t_{p3} stand for the time durations of each trajectory segment. According to the SIASUN GCR16-2000 manual, the maximum velocity is 4 m/s, and the maximum acceleration is 4 m/s^2.

In this study, the improvements to the standard PSO proposed are mainly reflected in the following four aspects.

3.4.1. Tent Chaotic Map

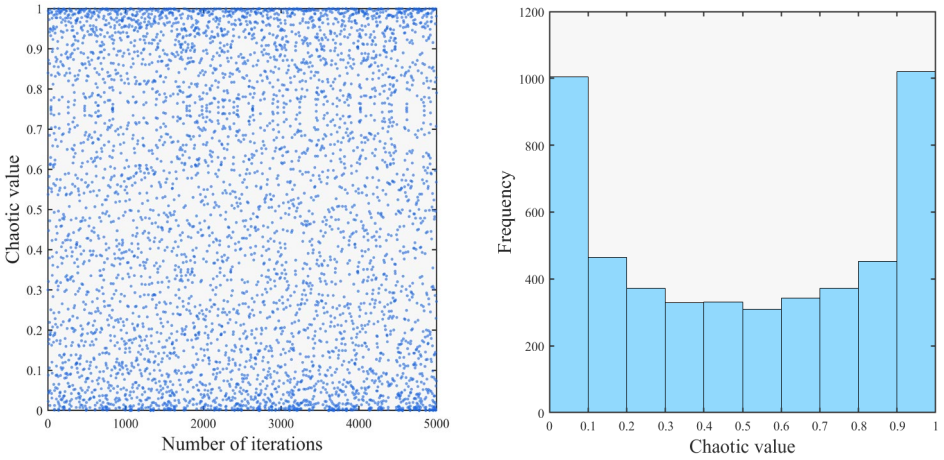
In the PSO algorithm, particle positions and velocities are typically initialized randomly, which may result in uneven initial population distribution. Moreover, when the initialized population is overly concentrated in certain regions of the search space, the algo-

rithm is prone to getting trapped in local optima. Compared to the standard PSO, chaotic initialization enhances the diversity and coverage of the particle swarm, thereby improving population quality and the global search ability. There are various types of chaotic maps, commonly including the logistic map, tent map, and Hénon map, among others. The tent map features a simple structure, ease of analysis, good ergodicity, high computational efficiency, and strong chaotic behavior, making it widely used in the initialization phase of optimization algorithms. Based on these considerations, this study selects the tent map, whose mathematical expression is given in Equation (27).

$$T_{n+1} = \begin{cases} \frac{S_n}{\beta} & 0 \le S_n < \beta \\ \frac{1-S_n}{1-\beta} & \beta \le S_n \le 1 \end{cases} \tag{27}$$

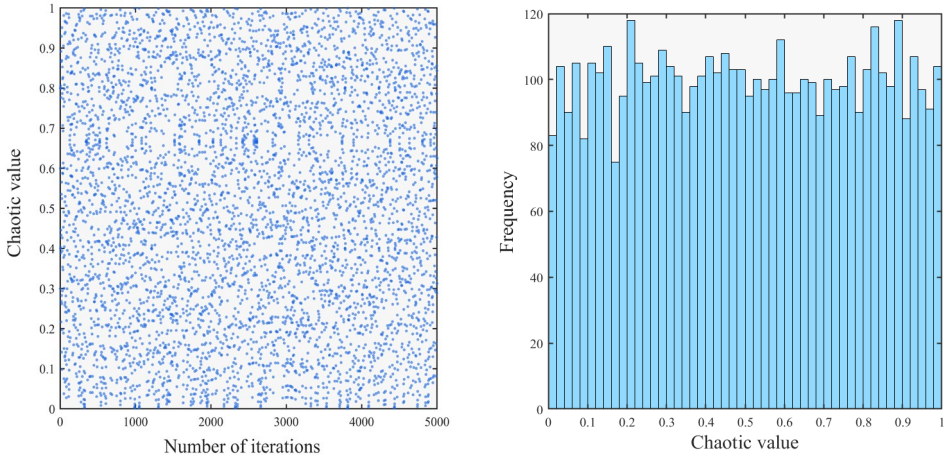
where S_n denotes the chaotic number at the n-th iteration; β is a user-defined parameter with a value range of [0, 1].

Assuming the particle swarm undergoes 5000 iterations, the population is initialized using logistic and tent chaotic maps, respectively, with their distribution illustrated in Figures 2 and 3.



(**a**) (**b**)

Figure 2. (**a**) Logistic chaotic mapping distributions; (**b**) Logistic chaotic mapping frequency distributions.



(**a**) (**b**)

Figure 3. (**a**) Distributions of the tent chaotic map; (**b**) Frequency distributions of the tent chaotic map.

As shown in Figure 3, when β = 0.5, the initial values across all dimensions are more widely and evenly distributed within the search space, enhancing the search coverage and the likelihood of locating the global optimum. Therefore, in this study, β is set to 0.5.

3.4.2. Inertia Weight

The solution accuracy and convergence speed of PSO are influenced by the inertia weight τ. A larger inertia weight makes the global search capabilities strong but may reduce its local search effectiveness; a smaller inertia weight exhibits the opposite effect. To balance local and global search capability, a linearly decreasing inertia weight is designed, with its update formula shown in Equation (28).

$$\tau(n) = \tau_{\max} - (\tau_{\max} - \tau_{\min})\left(\frac{k_n}{K_{\max}}\right) \tag{28}$$

where τ_{max} denotes the maximum inertia weight; the minimum inertia weight is represented by τ_{min}; k_n and K_{max} denote the n-th iteration and the maximum number of iterations, respectively.

Shi et al. [21] experimentally investigated the impact of different inertia weight values on the PSO algorithm. They found that, when the maximum value exceeds 0.9, the particle velocity increases too rapidly, causing particles to overshoot the optimal solution and making convergence difficult or even leading to divergence. Conversely, when the minimum value is below 0.4, the particle velocity quickly decays to zero, resulting in a rapid loss of population diversity and premature convergence to local optima. According to their study, the maximum inertia weight is set to 0.9, while the minimum inertia weight is set to 0.4. In the early stages, a larger inertia weight is applied to strengthen global exploration. It is then gradually reduced in the later stages to improve local exploitation. This approach balances global exploration with local search, ensuring favorable convergence performance.

3.4.3. Learning Factor

In the PSO, the constant learning factors limit the adaptability and flexibility of the algorithm. Fixed learning factors make it difficult to balance convergence speed and solution accuracy. Larger learning factors may induce oscillations in the algorithm, resulting in unstable convergence, whereas smaller learning factors can maintain stability but tend to reduce the convergence speed. In this study, the learning factors λ_1 and λ_2 are defined as dynamically varying parameters and are adjusted using a sinusoidal function. As the algorithm iterates, λ_1 gradually decreases, while λ_2 gradually increases, enhancing the individual learning capabilities in the early stages, strengthening their global cognitive capability in the later stages, thereby improving the convergence performance and search efficiency. The expressions of λ_1 and λ_2 are given in Equations (29) and (30), respectively.

$$\lambda_1 = 2\sin^2\left[\frac{\pi}{2}\left(1 - \frac{k_n}{K_{\max}}\right)\right] \tag{29}$$

$$\lambda_2 = 2\sin^2\left(\frac{k_n \pi}{2K_{\max}}\right) \tag{30}$$

3.4.4. WOA Optimization Mechanism

During the exploitation and exploration process of the standard PSO, particle trajectories are mainly driven by linear attraction toward the global best and the individual historical best solutions, which causes particles to rapidly cluster around the current global best during iterations, becoming trapped in local optima. In response to these problems, the optimization mechanism of the whale optimization algorithm (WOA) is incorporated in this study.

During the prey search phase, the random foraging behavior of a whale population is simulated, in which an individual randomly selects another whale as a target and continuously moves toward it. The process of prey searching by whales is described in Equation (31).

$$YP(n+1) = YP_{rand}(n) - G \cdot F_1 \tag{31}$$

with

$$F_1 = |E \cdot YP_{rand}(n) - YP(n)| \tag{32}$$

where $YP_{rand}(n)$ denotes a randomly selected whale from the current population; F_1 represents the search length. E stands for a random parameter with a value range of [0, 2].

The expression of $|G|$ is given in Equation (33).

$$\begin{cases} G = 2g \cdot u - g \\ g = 2 - \frac{2k_n}{K_{\max}} \end{cases} \tag{33}$$

where u represents a random parameter with a value range of [0, 1]; g means the control parameter.

The whale population captures prey using a bubble-net feeding strategy, which includes encircling contraction and spiral updating. During the shrinking encirclement phase, the whale that currently attains the best solution is regarded as the target, and the remaining whales continuously move toward its position. Its expression is given in Equation (34).

$$YP(n+1) = YP_{\text{best}}(n) - G \cdot F_2 \tag{34}$$

with

$$F_2 = |E \cdot YP_{\text{best}}(n) - YP(n)| \tag{35}$$

where $YP_{best}(n)$ denotes the individual located at the best position in the current population; F_2 represents the encircling length.

During the spiral updating phase, the other whales approach the prey by moving along a spiral path toward the best individual, thereby exploring potential optimal solutions between themselves and the optimal whale. Its expression is given in Equation (36).

$$YP(n+1) = F_3 \cdot e^{hq} \cdot \cos(2\pi h) + YP_{\text{best}}(n) \tag{36}$$

with

$$F_3 = |YP_{\text{best}}(n) - YP(n)| \tag{37}$$

where F_3 represents the distance between the current whale and the whale at the optimal position; q is a constant coefficient; h stands for a random parameter with a value range of [0, 1].

P denotes the probability parameter, set to 0.5. $P \geq 0.5$. The particle population is updated according to Equation (39); $P < 0.5$. When $|G| < 1$, the particle swarm is updated using Equation (24); when $|G| \geq 1$, the update follows Equation (38).

$$V_m(n+1) = \tau V_m(n) + (Pb_{best}(n) - L_m(n))\lambda_1 r_1 + (Pb_{rand}(n) - L_m(n))\lambda_2 r_2 \tag{38}$$

$$V_m(n+1) = F_P \cdot e^{hq} \cdot \cos(2\pi l) + \tau V_m(n) \tag{39}$$

with

$$F_P = |Gb_{best}(n) - L_m(n)| \tag{40}$$

$$l = (g-1) \cdot u + 1 \tag{41}$$

where $Pb_{rand}(n)$ denotes a randomly selected particle from the current population. F_P means the distance between the current particle and the particle at the global best position.

3.5. The CAW-PSO Procedure

Based on the aforementioned improvement strategies, the CAW-PSO trajectory planning flowchart is presented in Figure 4.

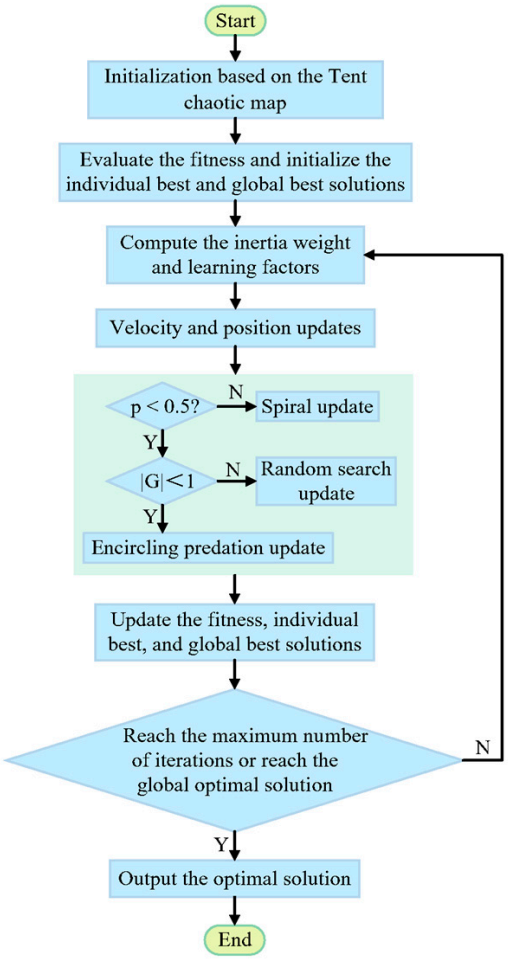


Figure 4. The CAW-PSO algorithm flowchart.

Step 1: Initialize the particle positions using tent chaotic mapping, and initialize the particle velocities using a random function.

Step 2: Calculate the fitness values of all particles and determine the global best solution and the individual best solution.

Step 3: Calculate the inertia weight according to Equation (28); compute the learning factors based on Equations (29) and (30).

Step 4: Update velocity and position according to the standard PSO, applying boundary constraints.

Step 5: Update by integrating the WOA optimization mechanism.

Step 6: Calculate the new fitness values and update the personal best and global best solutions. If the current solution corresponds to a shorter total time, update the optimal time vector set and the global best fitness value.

Step 7: If the termination criteria are not met, repeat Steps 3–6; otherwise, terminate the algorithm and output the global best particle position.

4. Results and Discussion

4.1. Performance Analysis of the CAW-PSO

The CAW-PSO algorithm is configured with the following parameters: inertia weight $\tau_{max} = 0.9$, $\tau_{min} = 0.4$, learning factor $\lambda_1 = 2$, $\lambda_2 = 2$, probability parameter $P_{WOA} = 0.5$, spiral shape constant $B_{WOA} = 1$, control parameter $A_{WOA} = 2$, chaotic mapping parameters $\beta = 0.5$, population size 50; maximum number of iterations 1000. Each algorithm is independently executed 30 times on the test functions. The detailed parameter settings for the PSO, WOA, and WOA-PSO (WOA-PSO hybrid algorithm) are listed in Table 2.

Table 2. Algorithm parameter settings.

Algorithm	Parameter Setting
CAW-PSO	$\tau_{max} = 0.9$, $\tau_{min} = 0.4$, $\lambda_1 = 2$, $\lambda_2 = 2$, $P_{WOA} = 0.5$, $B_{WOA} = 1$, $A_{WOA} = 2$, $\beta = 0.5$
PSO	$\tau = 0.7$, $\lambda_1 = 2$, $\lambda_2 = 2$
WOA	$A_{WOA} = 2$, $B_{WOA} = 1$, $P_{WOA} = 0.5$
WOA-PSO	$\tau = 0.7$, $\lambda_1 = 2$, $\lambda_2 = 2$, $A_{WOA} = 2$, $B_{WOA} = 1$, $P_{WOA} = 0.5$

The population size, maximum number of iterations, and number of independent runs are kept consistent across the three algorithms.

Seven test functions were selected from the CEC2017 benchmark suite; details are provided in Table 3. The test dimension is 10, with a search space of $[-100, 100]^D$.

Table 3. Function test table.

Name	Function
F5	$f_5(X) = \sum_{i=1}^{D}\left(x_i^2 - 10\cos(2\pi X_i) + 10\right)$
F8	$f_8(X) = \sum_{i=1}^{D}\left(Z_i^2 - 10\cos(2\pi z_i) + 10\right) + f_{13}^*$ $\hat{x} = M_1 \frac{5.12(x-o)}{100}$ $y_i = \begin{cases} \hat{x}_i & if\ \lvert\hat{x}_i\rvert \le 0.5 \\ \text{round}(2\hat{x}_i)/2 & if\ \lvert\hat{x}_i\rvert > 0.5 \end{cases}$ for i = 1.2 . . . . D $z = M_1\Lambda^{10}M_2T_{asy}^{0.2}(T_{osz}(y))$
F14	$f_{14}(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k\max}\left[a^k\cos(2\pi b^k(x_i + 0.5))\right]\right)$ $-D\sum_{k=0}^{k\max}\left[a^k\cos(2\pi b^k \cdot 0.5)\right]$ $a = 0.5, b = 3, k\max = 20$
F17	$f_{17}(X) = \left\lvert\sum_{i=1}^{D} x_i^2 - D\right\rvert^{1/4} + \left(0.5\sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i\right)/D + 0.5$
F26	$F_6(X) = f_{20}\left(M(\frac{0.5(X-\theta_6)}{100})\right) + {F_6}^*$
F27	$F_7(X) = f_7(M(\frac{600(X-\theta_7)}{100})) + {F_7}^*$
F30	$F_{10}(X) = f_{10}(\frac{10000(X-\theta_{10})}{100}) + {F_{10}}^*$

The algorithms are evaluated based on the best value, mean, and standard deviation. The statistical data of the test results are summarized in Table 4, and the convergence curves of the algorithms on the test functions are compared in Figure 5.

Table 4. Test results.

Name	Evaluation	PSO	WOA	WOA-PSO	CAW-PSO
CEC2017-F5	Best	5.0696×10^{2}	5.2002×10^{2}	5.1408×10^{2}	5.0498×10^{2}
	Average	5.1817×10^{2}	5.5192×10^{2}	5.3379×10^{2}	5.1698×10^{2}
	STD	8.3631×10^{0}	1.8854×10^{1}	1.1525×10^{1}	7.6321×10^{0}
CEC2017-F8	Best	8.0597×10^{2}	8.1315×10^{2}	8.1005×10^{2}	8.0400×10^{2}
	Average	8.1430×10^{2}	8.3946×10^{2}	8.2463×10^{2}	8.1345×10^{2}
	STD	7.1416×10^{0}	1.3217×10^{1}	9.2271×10^{0}	5.1017×10^{0}
CEC2017-F14	Best	1.4314×10^{3}	1.4728×10^{3}	1.4679×10^{3}	1.4264×10^{3}
	Average	1.4767×10^{3}	1.8384×10^{3}	1.5211×10^{3}	1.4569×10^{3}
	STD	6.2504×10^{1}	6.6097×10^{2}	3.3681×10^{1}	2.3777×10^{1}
CEC2017-F17	Best	1.7080×10^{3}	1.7278×10^{3}	1.7255×10^{3}	1.7083×10^{3}
	Average	1.7519×10^{3}	1.8040×10^{3}	1.7870×10^{3}	1.7434×10^{3}
	STD	3.3134×10^{1}	5.4522×10^{1}	5.0668×10^{1}	1.6538×10^{1}
CEC2017-F26	Best	2.6000×10^{3}	2.6093×10^{3}	2.6091×10^{3}	2.6000×10^{3}
	Average	2.9623×10^{3}	3.3725×10^{3}	2.9840×10^{3}	2.8503×10^{3}
	STD	2.7395×10^{2}	5.0544×10^{2}	2.8330×10^{2}	1.0393×10^{2}
CEC2017-F27	Best	3.0946×10^{3}	3.0931×10^{3}	3.0923×10^{3}	3.0911×10^{3}
	Average	3.1138×10^{3}	3.1398×10^{3}	3.1235×10^{3}	3.0998×10^{3}
	STD	2.8241×10^{1}	4.6336×10^{1}	3.5436×10^{1}	1.0002×10^{1}
CEC2017-F30	Best	7.2095×10^{3}	4.2897×10^{3}	4.9734×10^{3}	4.8946×10^{3}
	Average	2.5227×10^{5}	8.5997×10^{5}	4.7330×10^{5}	8.7958×10^{4}
	STD	4.0853×10^{5}	8.6233×10^{5}	5.2130×10^{5}	2.1363×10^{5}

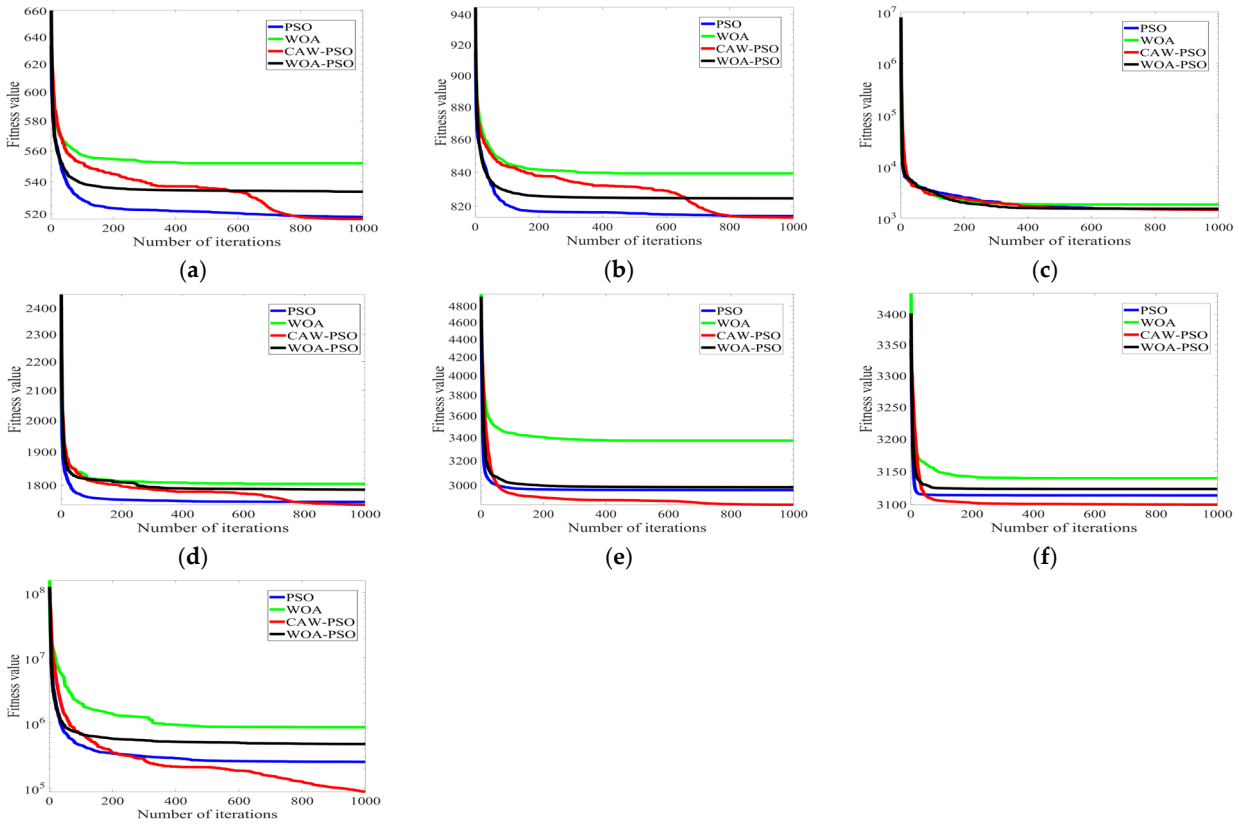


Figure 5. Comparison of convergence curves of the algorithms on the test functions: (**a**) F5; (**b**) F8; (**c**) F14; (**d**) F17; (**e**) F26; (**f**) F27; (**g**) F30.

Based on the test results, compared with the PSO, WOA, and WOA-PSO, the CAW-PSO achieves the best values closer to the true solution and exhibits higher convergence accuracy. Furthermore, it exhibits smaller mean values and standard deviations, indicating greater consistency across multiple independent runs and improved algorithmic stability. In terms of optimization performance and result stability, results indicate that the CAW-PSO is superior to PSO, WOA, and WOA-PSO.

In Figure 5, PSO, WOA, and WOA-PSO rapidly converge by getting trapped in local optima. The CAW-PSO algorithm exhibits multiple decreases during the iteration process, gradually approaching the optimal solution and ultimately converging to the global optimum. This indicates that the CAW-PSO algorithm can effectively avoid local optima.

4.2. Simulation Results and Analysis

In this study, the SIASUN GCR16-2000 robotic arm model introduced in Section 2 is employed for simulation analysis, comparing the optimization results of the CAW-PSO, PSO, WOA, and WOA-PSO.

Four path points of the robotic arm end-effector are defined as follows: starting point Q_1 (−466.6, −136.8, 402.5); end point Q_4 (63.2, 1173.5, 1851.9); waypoints Q_2 (−93.9, 815.1, 818.3) and Q_3 (−17.3, 686.1, 1813.5), unit: millimeter. Q_1 and Q_4 determine the starting and ending points of the manipulator motion, respectively, while Q_2 and Q_3 are located at the turning points where the trajectory changes. The spatial distribution of the path points is shown in Figure 6. Based on the inverse kinematics solution of the robot, the joint angles corresponding to the four path points are obtained, as detailed in Table 5.

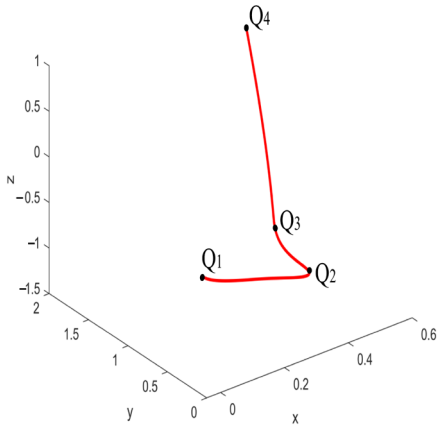


Figure 6. Spatial distribution of path points.

Table 5. Inverse kinematics solutions for path points.

Joint/p	Q_1/(rad)	Q_2/(rad)	Q_3/(rad)	Q_4/(rad)
1	3.056	1.478	1.321	1.364
2	0.293	−0.041	−0.201	0.426
3	2.783	2.229	1.186	0.448
4	−3.068	−2.181	−0.977	−1.163
5	−1.431	−1.515	−1.355	−1.409
6	−0.121	−0.119	−0.12	1.545

The population size is 50, with a maximum of 200 iterations. The maximum velocity and acceleration are 4 m/s and 4 m/s^2, respectively. The total time before optimization is 12 s. Using the MATLAB simulation environment, the optimization performance of the CAW-PSO, PSO, WOA, and WOA-PSO is comparatively analyzed for the time-optimal trajectory planning problem.

The optimization results for each joint are presented in Table 6. Comparison of optimization performance for Joint 1 shows that the CAW-PSO algorithm improves by 13.19% over PSO, by 1.11% over WOA and by 3.14% over WOA-PSO. The CAW-PSO algorithm improves by 4.88% over PSO, by 1.11% over WOA and by 11.41% over WOA-PSO for Joint 2. Comparison of optimization performance for Joint 3 shows that the CAW-PSO algorithm improves by 11.65% over PSO, by 4.02% over WOA and by 2.32% over WOA-PSO. The CAW-PSO algorithm improves by 17.98% over PSO, by 1.98% over WOA and by 12.41% over WOA-PSO for Joint 4. Comparison of optimization performance for Joint 5 shows that the CAW-PSO algorithm improves by 11.03% over PSO, by 6.63% over WOA and by 16.55% over WOA-PSO. The CAW-PSO algorithm improves by 7.45% over PSO, by 4.24% over WOA and by 2.35% over WOA-PSO for Joint 6.

Table 6. Optimization performance of each joint.

Joint/p	PSO	WOA	WOA-PSO	CAW-PSO
1	2.1506	1.8897	1.9369	1.8670
2	1.9829	1.8573	1.9941	1.7666
3	2.6084	2.4011	2.3593	2.3045
4	2.7228	2.2784	2.5496	2.2333
5	0.8185	0.7799	0.8726	0.7282
6	1.7747	1.7153	1.6821	1.6425

The convergence curve comparison for each joint is shown in Figure 7, and the overall convergence curve comparison of all joints is presented in Figure 8. In Figure 7, curves of different colors represent the convergence processes of each joint under different algorithms.

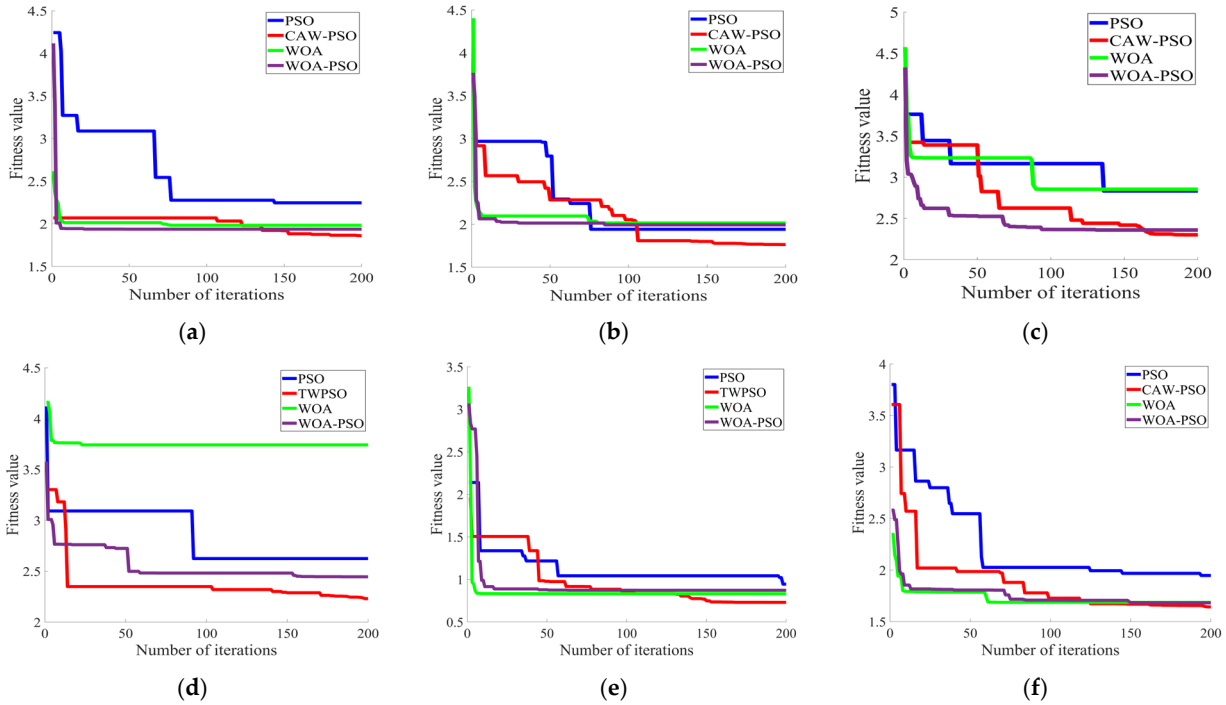


Figure 7. Joint optimization comparison: (**a**) Joint 1; (**b**) Joint 2; (**c**) Joint 3; (**d**) Joint 4; (**e**) Joint 5; (**f**) Joint 6.

In comparison to PSO, WOA, and WOA-PSO, the CAW-PSO shows a smoother gradient descent in its convergence curve, attains a lower terminal value, and achieves better optimization performance.

For the overall optimization of joints, the optimized fitness value of CAW-PSO is 1.757, while those of PSO, WOA, and WOA-PSO are 2.010, 1.820, and 1.818, respectively. The CAW-PSO algorithm improves by 12.59% over PSO, by 3.46% over WOA, and by 3.36% over WOA-PSO. As shown in Figure 8, the optimization performance of CAW-PSO is the best.

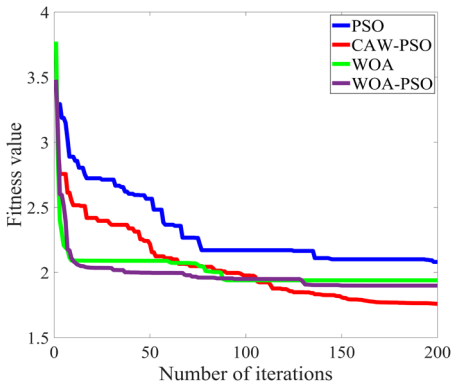


Figure 8. Overall convergence curve comparison.

The above results indicate that the CAW-PSO algorithm is superior to the PSO, WOA, and WOA-PSO algorithms, showing better convergence performance and stability.

Figure 9 presents a comparison of trajectories, where (a), (c), and (e) depict the joint motion curves obtained through 3-5-3 polynomial interpolation, while (b), (d), and (f) represent the joint motion curves optimized by the CAW-PSO algorithm. After optimizing by the CAW-PSO algorithm, the joint motion time decreased from 12 s to 3.661 s, reducing by 69.94%. The start and end positions of the trajectory curves remained unchanged before and after optimization, while the velocities and accelerations of each joint were significantly improved after optimization. The CAW-PSO algorithm greatly reduced the motion time, enhancing the efficiency of the cleaning operation. Simulation results demonstrate the effectiveness of the CAW-PSO algorithm.

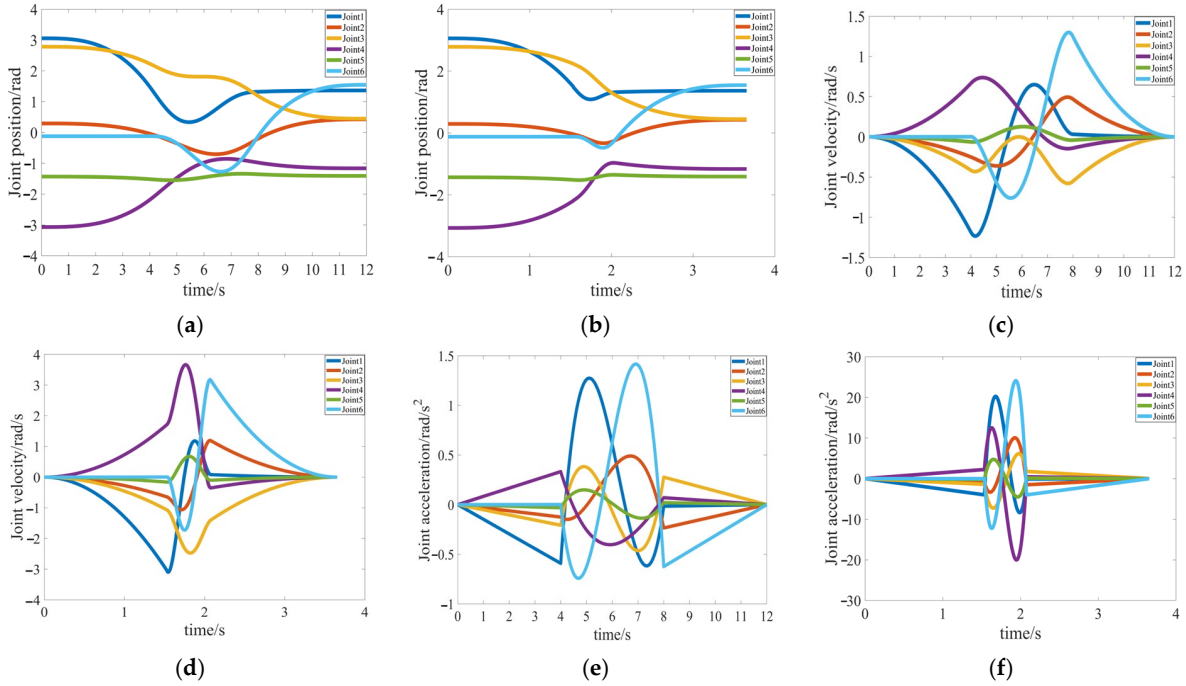


Figure 9. Comparison of trajectory curves: (**a**) Unoptimized position curves; (**b**) Position curves after CAW-PSO optimization; (**c**) Unoptimized velocity curves; (**d**) Velocity curves after CAW-PSO optimization; (**e**) Unoptimized acceleration curves; (**f**) Acceleration curves after CAW-PSO optimization.

4.3. Experimental Verification on Actual Equipment

Cleaning logic setup: The robotic arm moves from the start-up position to the cleaning start point, performing the cleaning task. After completing the task, it moves to a temporary standby position to prepare for the next cleaning operation. Upon completing all cleaning tasks, the robotic arm returns to the start-up position.

The experimental setup is shown in Figure 10. The experimental setup consists of a host computer, a collaborative robot, luminaires, and an integrated control cabinet. The integrated control cabinet houses the controller, Ethernet cable, relay, and power supply.

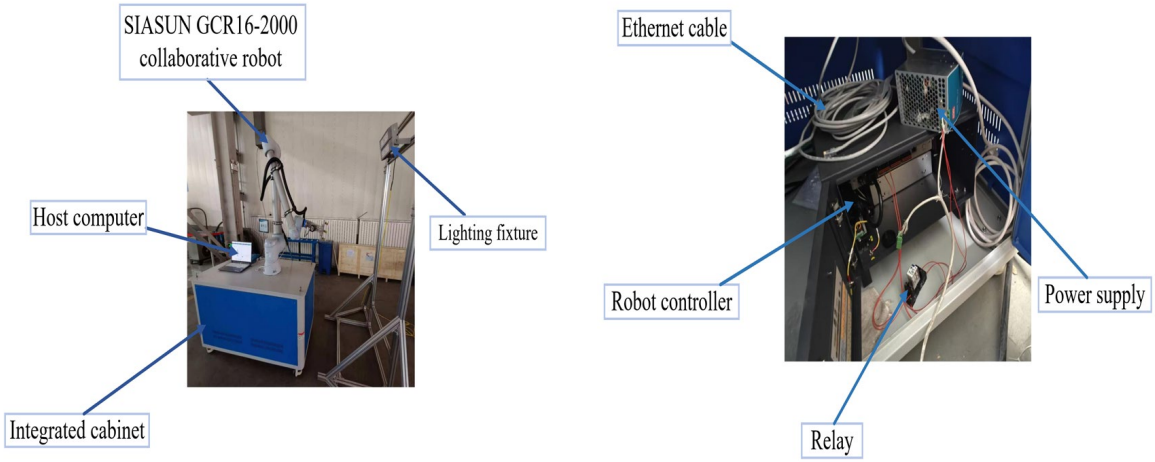


Figure 10. Experimental setup: (**a**) Experimental platform layout; (**b**) Internal layout of the integrated control cabinet.

After optimization by the CAW-PSO algorithm, the optimized trajectory key points are sent to the SIASUN GCR16-2000 collaborative robot controller via the host computer, commanding the robot to move to the cleaning start point. The cleaning operation is simulated, moving to the standby position. The robot executes five times, and the runtime of each run is recorded in Table 7. The operating state of the robotic arm is monitored through the interface shown in Figure 11. The entire motion process is illustrated in Figure 12. Compared with the simulation time, the relative errors of the five experimental runs are 0.22%, 0.33%, 1.64%, 2.24%, and 1.37%, respectively, with a mean relative error of 1.16%. The experimental results show good agreement with the simulation results, indicating that the CAW-PSO algorithm achieves satisfactory optimization performance. In the motion process, no error messages are generated in the monitoring interface shown in Figure 11, indicating that no fault alarms are triggered. The variations in joint angles in Figure 11 do not exceed the hardware limits of the robot, suggesting that no abrupt changes occur in velocity or acceleration. The real-world experiments fully confirm the feasibility of the CAW-PSO algorithm.

Table 7. Robot motion time.

Number of Experiments	1	2	3	4	5
Runtime/s	3.669	3.673	3.721	3.743	3.711

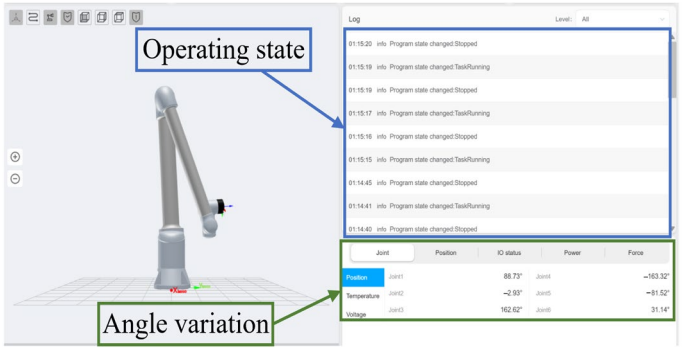


Figure 11. Operational monitoring of the robotic arm.

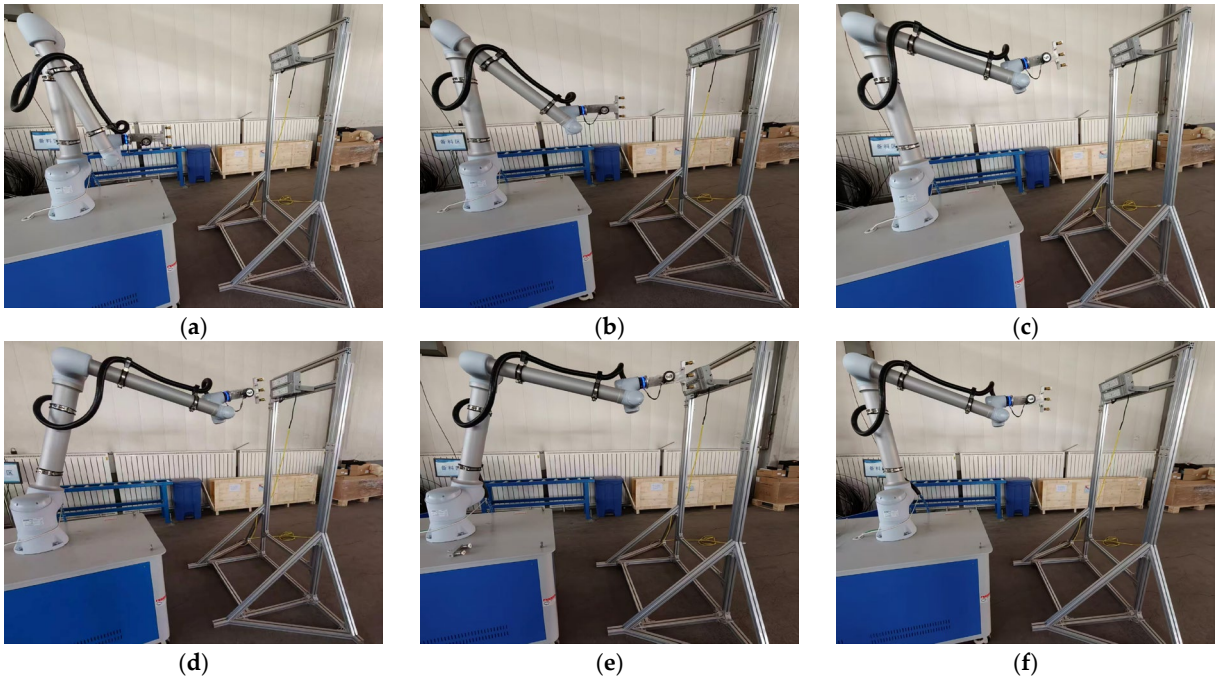


Figure 12. Experimental process: (**a**) Start-up point; (**b**) Path point 1; (**c**) Path point 2; (**d**) Cleaning start point; (**e**) Cleaning end point; (**f**) Temporary standby point.

The proposed CAW-PSO algorithm is suitable for solving the trajectory planning problem of tunnel luminaire cleaning robotic manipulators, but it may not be directly applicable to other domains. The CAW-PSO algorithm can effectively escape local optima, but it requires more computational resources. Due to joint friction in the robotic manipulator, the actual motion time does not reach the simulation result.

5. Conclusions

This paper proposes an improved particle swarm optimization algorithm (CAW-PSO). Trajectory construction is performed by 3-5-3 polynomial interpolation to implement time-optimal trajectory planning for the tunnel lighting cleaning robot arm. The purpose is to reduce the motion time of the robot arm. The improvements include: first, the tent chaotic map is introduced to improve population quality; second, a linearly decreasing inertia weight is designed, balancing global and local search; next, dynamic learning factors are defined, improving the individual learning and global cognition of particles; finally, the optimization mechanism of the WOA is integrated to escape local optima, increasing convergence accuracy.

Seven test functions are selected from the CEC2017 benchmark suite to compare algorithm performance. The test results demonstrate that CAW-PSO excels over PSO, WOA, and WOA-PSO when addressing both simple and complex problems. Simulation comparisons and real-world experiments are conducted. After optimization by the CAW-PSO algorithm, the motion time was reduced by 69.49%, decreasing from 12 s to 3.661 s. In the optimization of robot arm joints, the CAW-PSO algorithm improves by 12.59% over PSO, by 3.46% over WOA, and by 3.36% over WOA-PSO. In the operation of the SIASUN GCR16-2000 physical robot, no errors are reported, the emergency stop is not triggered, and no sudden changes occurred in velocity or acceleration. The mean relative error of the actual runtime is 1.16%, the experimental results are in good agreement with the simulation results. The effectiveness and feasibility of CAW-PSO have been validated in the field of trajectory planning for tunnel lighting cleaning robots.

In future work, more advanced optimization mechanisms will be investigated to reduce computational resource consumption and lower computational costs. In trajectory planning, energy optimization is incorporated to reduce the energy consumption of the robotic manipulator during task execution and improve the overall system efficiency.

Author Contributions: Conceptualization, Z.Y. and T.S.; methodology, Z.Y.; software, T.S. and H.L.; validation, T.S.; formal analysis, T.S.; investigation, T.S.; resources, Z.Y.; data curation, Z.Y.; writing—original draft preparation, T.S.; writing—review and editing, Z.Y.; visualization, T.S.; supervision, H.Z.; project administration, Z.Y.; funding acquisition, Z.L. and H.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the open project special fund of Intelligent Transportation Laboratory in Shanxi Province, grant number 2024-ITLOP-KD-05.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available by request from the corresponding author.

Conflicts of Interest: Authors Zhibin Yao, Taibo Song, and Hui Li were employed by the company China Shanxi Provincial Intelligent Transportation Laboratory Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

PSO	Particle Swarm Optimization
WOA	Whale Optimization Algorithm
WOA-PSO	Whale Optimization Algorithm–Particle Swarm Optimization hybrid algorithm
CAW-PSO	Chaotic Adaptive Whale–Particle Swarm Optimization

References

1. Vega-Heredia, M.; Muhammad, I.; Ghanta, S.; Ayyalusami, V.; Aisyah, S.; Elara, M.R. Multi-sensor orientation tracking for a façade-cleaning robot. *Sensors* **2020**, *20*, 1483. [CrossRef] [PubMed]
2. Sun, H.; Xu, R.; Luo, J.; Cheng, H. Review of the Application of UAV Edge Computing in Fire Rescue. *Sensors* **2025**, *25*, 3304. [CrossRef] [PubMed]
3. Valero, S.; Martinez, J.C.; Montes, A.M.; Marín, C.; Bolaños, R.; Álvarez, D. Machine vision-assisted design of end effector pose in robotic mixed depalletizing of heterogeneous cargo. *Sensors* **2025**, *25*, 1137. [CrossRef] [PubMed]
4. Ji, Y.; Yu, J. Time-Impact Optimal Trajectory Planning for Wafer-Handling Robotic Arms Based on the Improved Snake Optimization Algorithm. *Sensors* **2025**, *25*, 1872. [CrossRef] [PubMed]

5. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43. [CrossRef]
6. Gong, C.; Zhou, N.; Xia, S.; Huang, S. Quantum particle swarm optimization algorithm based on diversity migration strategy. *Futur. Gener. Comput. Syst.* **2024**, *157*, 445–458. [CrossRef]
7. Luo, T.; Xie, J.; Zhang, B.; Zhang, Y.; Li, C.; Zhou, J. An improved levy chaotic particle swarm optimization algorithm for energy-efficient cluster routing scheme in industrial wireless sensor networks. *Expert Syst. Appl.* **2024**, *241*, 122780. [CrossRef]
8. Wang, L.; Hong, L.; Fu, H.; Cai, Z.; Zhong, Y.; Wang, L. Adaptive distance-based multi-objective particle swarm optimization algorithm with simple position update. *Swarm Evol. Comput.* **2025**, *94*, 101890. [CrossRef]
9. Allaoui, M.; Belhaouari, S.B.; Hedjam, R.; Bouanane, K.; Kherfi, M.L. t-SNE-PSO: Optimizing t-SNE using particle swarm optimization. *Expert Syst. Appl.* **2025**, *269*, 126398. [CrossRef]
10. Jiao, C.; Yu, K.; Zhou, Q. An opposition-based learning adaptive chaotic particle swarm optimization algorithm. *J. Bionic Eng.* **2024**, *21*, 3076–3097. [CrossRef]
11. Nickabadi, A.; Ebadzadeh, M.M.; Safabakhsh, R. A novel particle swarm optimization algorithm with adaptive inertia weight. *Appl. Soft Comput.* **2011**, *11*, 3658–3670. [CrossRef]
12. Alatas, B.; Akin, E.; Ozer, A.B. Chaos embedded particle swarm optimization algorithms. *Chaos Solitons Fractals* **2009**, *40*, 1715–1734. [CrossRef]
13. Pan, K.; Liang, C.-D.; Lu, M. Optimal scheduling of electric vehicle ordered charging and discharging based on improved gravitational search and particle swarm optimization algorithm. *Int. J. Electr. Power Energy Syst.* **2024**, *157*, 109766. [CrossRef]
14. Sun, B.; Li, G.; Wang, S.; Xie, C. Two-dimensional bin-packing problem with conflicts and load balancing: A hybrid chaotic and evolutionary particle swarm optimization algorithm. *Comput. Ind. Eng.* **2025**, *200*, 110851. [CrossRef]
15. Shu, B.; Hu, G.; Cheng, M.; Zhang, C. MSFPSO: Multi-algorithm integrated particle swarm optimization with novel strategies for solving complex engineering design problems. *Comput. Methods Appl. Mech. Eng.* **2025**, *437*, 117791. [CrossRef]
16. Yan, J.; Zheng, K.; Zhang, M.; Wang, J.; Ma, F.; Li, Z.; Zhu, F.; Liu, H. An enhanced Lévy flight quantum particle swarm optimization for stress monitoring in jacket wind turbines. *Ocean Eng.* **2025**, *324*, 120704. [CrossRef]
17. Wang, Y.; Liu, Z.; Han, H. Collaborative knowledge transfer-based multiobjective multitask particle swarm optimization. *Swarm Evol. Comput.* **2025**, *98*, 102115. [CrossRef]
18. Tao, F.; Chen, Z.; Wang, Z.; Zhu, L.; Wang, J. Multi-Strategy Improved Particle Swarm Optimization Algorithm for Path Planning of UAV in 3-D Low Altitude Urban Environment. *IEEE Internet Things J.* **2025**, *12*, 40470–40483. [CrossRef]
19. Xu, Y.; Zhang, M.; Wang, D.; Yang, M.; Liang, C. Hybrid Gaussian quantum particle swarm optimization and adaptive genetic algorithm for flexible job-shop scheduling problem. *Eng. Appl. Artif. Intell.* **2025**, *154*, 110882. [CrossRef]
20. Li, F.; Pan, H.; Ji, Y.; Ouyang, H. Particle swarm optimization based on particle perturbation and elite preservation strategies for dynamic optimization problems. *Expert Syst. Appl.* **2025**, *279*, 127423. [CrossRef]
21. Shi, Y.; Eberhart, R. A modified particle swarm optimizer. *Evol. Comput.* **1998**, *890*, 69–73.