Intelligence & Robotics

Research Article

Open Access

Time-optimal trajectory planning for a six-degree-of-freedom manipulator: a method integrating RRT and chaotic PSO

Zuoxun Wang, Chuanzhe Pang, Jinxue Sui, Guojian Zhao, Wangyao Wu, Liteng Xu

College of Information and Electronics Engineering, Shandong Technology and Business University, Yantai 264005, Shandong, China.

Correspondence to: Prof. Zuoxun Wang, College of Information and Electronics Engineering, Shandong Technology and Business University, Yantai 264005, Shandong, China; E-mail: 202214078@sdtbu.edu.cn

How to cite this article: Wang Z, Pang C, Sui J, Zhao G, Wu W, Xu L. Time-optimal trajectory planning for a six-degree-of-freedom manipulator: a method integrating RRT and chaotic PSO. *Intell Robot* 2024;4(4):479-502. http://dx.doi.org/10.20517/ir.2024.28

Received: 20 Oct 2024 **First Decision:** 12 Nov 2024 **Revised:** 6 Dec 2024 **Accepted:** 11 Dec 2024 **Published:** 28 Dec 2024

Academic Editor: Simon Yang **Copy Editor:** Pei-Yun Wang **Production Editor:** Pei-Yun Wang

Abstract

This study proposes an innovative algorithm based on a hybrid optimization strategy, integrating the rapidly-exploring random tree (RRT) and an improved particle swarm optimization (PSO) method to address the time-optimal trajectory planning problem for six-degree-of-freedom robotic arms. The proposed approach emphasizes obstacle avoidance and motion smoothness. RRT is utilized within a dynamic three-dimensional environment to rapidly generate an initial collision-free path. Subsequently, improved PSO enhances global search performance by introducing a multi-source chaotic mapping-based population initialization strategy, dynamically adjusted inertia weights, and non-linear learning factors. These enhancements effectively mitigate the limitations of traditional PSO methods, which are prone to premature convergence in complex optimization problems. Furthermore, the proposed 3-5-3 polynomial interpolation method significantly smooths the trajectory, reducing fluctuations in velocity and acceleration, and thereby improving the precision and energy efficiency of trajectory planning. Experimental results demonstrate that the proposed algorithm outperforms existing methods, such as the improved RRT and improved non-dominated sorting genetic algorithm, across multiple metrics. Notable achievements include a reduction of total motion time by approximately 21%, improved stability in robotic arm motion, and enhanced adaptability to dynamic environments. Particularly, the method achieves superior trajectory diversity and uniformity through joint optimization of multiple chaotic sources, overcoming the inherent limitations of single chaotic mappings. This research expands the application scenarios of RRT and PSO and provides a novel solution for intelligent control and real-time planning of high-degree-of-freedom robotic arms.

Keywords: Chaotic map, rapid random tree, particle swarm optimization algorithm, 3-5-3 polynomial

1. INTRODUCTION

Industrial robots have been used extensively in automated manufacturing across various sectors, including automotive and aerospace, as the industry continues to progress. This advancement has led to demands within the manufacturing industries for improved performance, adaptation to product variations, enhanced safety, and cost reduction[1–3]. However, standard control methods alone cannot fully meet these requirements. Trajectory planning[4] is fundamental to the motion control of industrial robots, directly influencing their efficiency, motion smoothness, and energy consumption. The widespread use of multi-degree-of-freedom robotic arms has significantly enhanced product accuracy and productivity. In many production workshops, robotic arms need to follow predetermined trajectories to optimize production efficiency by minimizing movement time. Consequently, reducing the movement time of robotic arms has emerged as a critical research focus.

At present, many researchers are integrating intelligent optimization algorithms to identify the optimal motion trajectory for robotic arms, aiming to reduce movement time through trajectory optimization. Ji *et al*. proposed the application of a multi-strategy improved sparrow algorithm for time-optimal trajectory planning of robotic arms, demonstrating the feasibility of their approach[5]. Wu *et al*. introduced a time-optimal trajectory planning method based on an improved butterfly algorithm, proving its effectiveness in reducing running time[6]. Zuo *et al.* presented an optimal trajectory planning for robotic arms based on an improved adaptive multi-objective particle swarm algorithm, validating the reliability of the reduced running time[7].

Zhang *et al*. proposed an obstacle avoidance algorithm for space hyper-redundant manipulators using combination of rapidly-exploring random tree (RRT) and shape control method, demonstrating the superiority of the RRT algorithm in target finding[8]. Rybus[9] introduced point-to-point motion planning of a free-floating space manipulator using the RRT method. The author proves the feasibility of collision-free point-to-point motion of this method in free floating space. Mizuno *et al*. proposed enhanced path smoothing based on conjugate gradient descent for firefighting robots in petrochemical complexes[10]. This involved modifying the hybrid state A* algorithm and implementing these modifications and smoothing using conjugate gradient descent to generate smooth paths that meet the robots' requirements. Hsu *et al*. developed a real-time optimal energy-saving walking pattern generator based on the gradient descent method and linear quadratic control[11]. By minimizing the cost function, the proposed center of gravity (COG) height trajectory optimization method can efficiently generate energy-saving trajectories under joint limit and joint velocity limit constraints.

As the degrees of freedom (DOF) or the number of joints in a robotic system increase, the control space expands exponentially. For robots with more joints, state variables such as joint angles, velocities, and accelerations significantly increase, leading to a substantial rise in the dimensionality of the search space. High-DOF robots typically involve more motion constraints, including inter-joint relative motion and workspace limitations, necessitating more sophisticated motion planning and optimization strategies. To address these challenges, the following approaches can be employed:

For high-DOF robotic arms, path planning and time optimization can be divided into multiple hierarchical levels. A coarse global path planning phase can be conducted initially, followed by a refined local optimization phase to improve the path and minimize motion time. Particle swarm optimization (PSO) with multi-granularity strategies can be employed to optimize the motion of robotic joints at various granular levels. This allows for more efficient handling of the increased dimensionality of the search space.

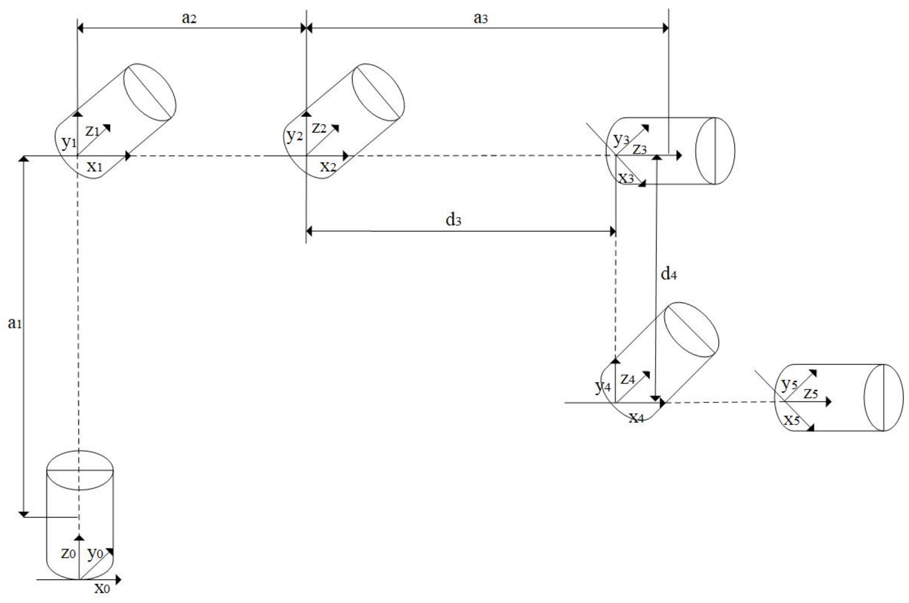

Figure 1. Coordinate system and model of manipulator connecting rod.

With the increase in DOF, joint constraints (e.g., maximum joint angles and velocity limits) and collision detection become more complex. These constraints can be incorporated into the fitness function of the PSO algorithm, ensuring feasible solutions through penalty methods or constraint-handling techniques. Given the intricate constraints that may arise in high-DOF systems, employing constraint optimization algorithms, such as modified PSO, can effectively eliminate solutions that fail to satisfy physical constraints.

In this paper, we enhance the obstacle avoidance capability of the robotic arm while optimizing the movement time, enabling it to complete its movement with only a starting point and an endpoint. Additionally, based on the 5th-degree polynomial[12], the 3-5-3 polynomial is employed to introduce nonlinearly decreasing adaptive inertia weights and asynchronous learning factors by improving the PSO algorithm. This approach helps to avoid the common issue of falling into local optima and ensures a balance between global search capabilities and local accuracy. Comparative tests on functions and changes in joint positions, velocities, and accelerations before and after optimization are conducted to demonstrate the innovation, superiority, and stability of the proposed algorithm.

2.ENVIRONMENT MODELING

2.1.Robot parameters

This study aims to optimize the trajectory planning problem of a six-degree-of-freedom robotic arm in computer numerical control (CNC) machining, enhancing machining precision and enabling the robotic arm to navigate around obstacles for effective gripping. Utilizing the robotics linkage coordinate system establishment method, the Denavit-Hartenberg (D-H) parameter method is selected to construct the mathematical model of the articulated six-degree-of-freedom robotic arm. The parameters are shown in Table 1, and the D-H coordinate system for each joint is illustrated in Figure 1.

In Table 1, a_i represents the length of the connecting rod, α_i indicates the torsion angle of the connecting rod, d_i denotes the connecting rod offset, and θ_i stands for the joint angle.

Table 1. Robot D-H parameters

Joint	a_i (mm)	α_i (°)	d_i (mm)	θ_i (°)	Joint range (°)
1	0	-pi/2	0	-pi/2	(-pi/2, pi/2)
2	100	0	0	3*pi/4	(-pi/2, pi)
3	10	-pi/2	20	-pi/4	(-pi/2, pi/2)
4	0	pi/2	100	0	0
5	0	-pi/2	0	0	0
6	0	0	0	0	0

D-H: Denavit-Hartenberg.

2.2. Construction of polynomial function for trajectory planning

To ensure the stability of the entire operational process, the position, angular velocity, and angular acceleration of each joint must vary continuously throughout the entire duration of operation. While high-order polynomial interpolation[13] can be employed, it often results in increased computational complexity and induces vibrations at the boundaries of the interpolation regions. To mitigate these challenges, the 3-5-3 polynomial interpolation method has been adopted in this study for the trajectory planning of the manipulator. Four path points are selected within the manipulator's workspace, forming four key points and three trajectories in the joint space. Subsequently, 3-5-3 polynomial interpolation planning is applied in sequence. The 3-5-3 polynomial is a high-order polynomial whose coefficients are determined based on boundary conditions (e.g., initial and final positions, velocities, and accelerations) and optimization objectives (e.g., minimizing motion time or maximizing smoothness). The 3-5-3 spline polynomial can be expressed as:

$$\begin{cases} h_{j1}(t) = a_{j13}t^3 + a_{j12}t^2 + a_{j11}t + a_{j10} \\ h_{j2}(t) = a_{j25}t^5 + a_{j24}t^4 + a_{j23}t^3 + a_{j22}t^2 + a_{j21}t + a_{j20} \\ h_{j3}(t) = a_{j33}t^3 + a_{j32}t^2 + a_{j31}t + aj_{30} \end{cases} \tag{1}$$

where the subscript j of the $h_{ji}(t)$ function term denotes the robotic arm joint number, i corresponds to the trajectory path segment, t represents the polynomial interpolation time of the $j-th$ joint, and a denotes the coefficients to be determined. The subscripts correspond to the joints, trajectories, and polynomial coefficients serial numbers.

By controlling velocity and acceleration, the 3-5-3 polynomial enables the robotic arm to accelerate rapidly in regions requiring higher speeds while decelerating gradually as it approaches the endpoint. This avoids unnecessary prolonged acceleration and stoppage, thereby optimizing the overall motion time. Sudden changes in velocity or acceleration during the robotic arm's movement along the path may lead to system oscillations or instability. The higher-order terms of the 3-5-3 polynomial, particularly the quintic term, allow it to effectively describe more complex and smoother motion trajectories. Through precise control of velocity and acceleration, the 3-5-3 polynomial minimizes abrupt changes or oscillations in the trajectory to the greatest extent possible.

According to the constraints of the hybrid polynomial, including the known initial point θ_{j0} of the $j-th$ joint angle, the path points θ_{j1} and θ_{j2}, and the endpoint θ_{j3}, the positions, velocities, and accelerations between the path points, along with the velocities and accelerations at the initial and final positions, are zero. With these 14 boundary conditions and over-constraints, we can solve for the unknown coefficients in the polynomials, denoted by a. Based on these constraints and boundary conditions, we can derive the matrix A. From the expression of the matrix, it can be seen that the constraints are only dependent on time t. The specific derivation can be given as follows:

$$A = \begin{bmatrix} M_{11} & 0 & M_{13} \\ 0 & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \\ M_{41} & M_{42} & M_{43} \end{bmatrix} \tag{2}$$

$$b = \begin{bmatrix} 0 & X_{j3} & 0 & 0 & X_{J0} & 0 & 0 & X_{j2} & X_{j1} \end{bmatrix}^T \tag{3}$$

$$c = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4}$$

$$a = A^{-1} b = \begin{bmatrix} A_1 & A_2 & A_3 \end{bmatrix} \tag{5}$$

where

$$M_{11} = \begin{bmatrix} t_1^3 & t_1^2 & t_1 & 1 \\ 3t_1^2 & 2t_1 & 10 & \\ 6t_1 & 2 & 0 & 0 \end{bmatrix} \tag{6}$$

$$M_{13} = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

$$M_{22} = \begin{bmatrix} t_2^5 & t_2^4 & t_2^3 & t_2^2 & t_2 & 1 \\ 5t_2^4 & 4t_2^3 & 3t_2^2 & 2t_2 & 1 & 0 \\ 20t_2^3 & 12t_2^2 & 6t_2 & 2 & 0 & 0 \end{bmatrix} \tag{8}$$

$$M_{23} = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & -2 & 0 & 0 \end{bmatrix} \tag{9}$$

$$M_{33} = \begin{bmatrix} t_3^3 & t_3^2 & t_3 & 1 \\ 3t_3^3 & 2t_3 & 1 & 0 \\ 6t_3 & 2 & 0 & 0 \end{bmatrix} \tag{10}$$

$$M_{41} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{11}$$

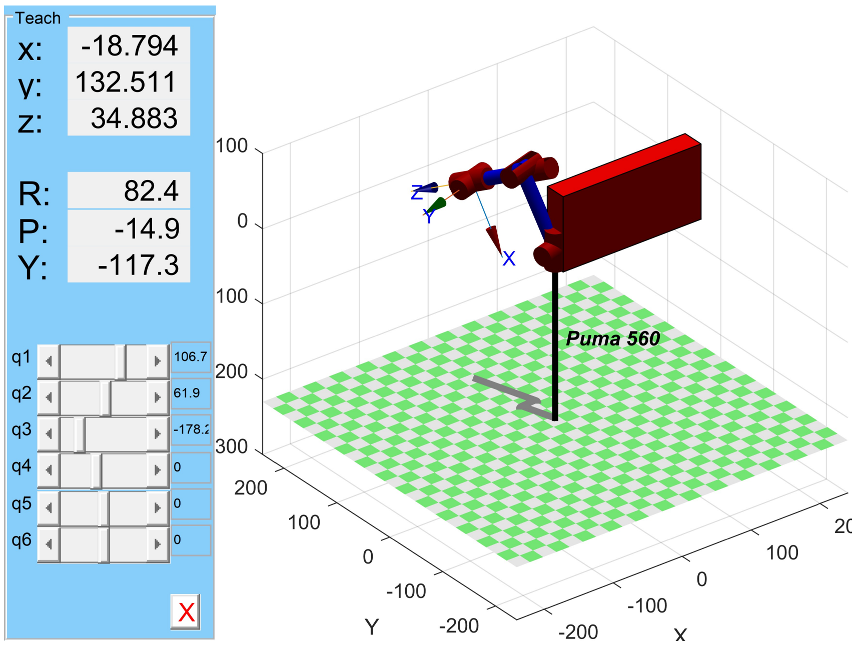

Figure 2. Simulation model of robotic arm initialization.

$$M_{43} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{12}$$

$$A_1 = \begin{bmatrix} a_{j13} & a_{j12} & a_{j11} & a_{j10} \end{bmatrix} \tag{13}$$

$$A_2 = \begin{bmatrix} a_{j25} & a_{j24} & a_{j23} & a_{j22} & a_{j21} & a_{j20} \end{bmatrix} \tag{14}$$

$$A_3 = \begin{bmatrix} a_{j33} & a_{j32} & a_{j31} & a_{j30} \end{bmatrix} \tag{15}$$

2.3. Mechanical arm simulation and construction

The six-degree-of-freedom robotic arm simulation model is constructed by calling the PUMA560 model in the Robotics Toolbox in MATLAB software using the established D-H parameters, as shown in Figure 2.

3. OPTIMIZATION METHODS

3.1. Rapid search randomized tree algorithm

The RRT algorithm[14], proposed by Lavalle in 1998, is a probabilistically complete method for fast path planning based on random sampling. RRT is an incremental sampling search technique that does not require parameter tuning in practice, enhancing its usability. One key advantage of the RRT algorithm is that node expansion does not require preprocessing, allowing it to quickly search for feasible paths based on current environmental conditions.

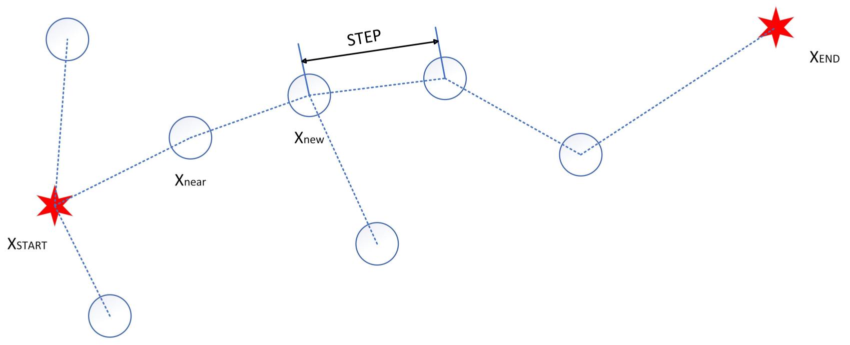

Figure 3. Schematic diagram of RRT algorithm. RRT: Rapidly-exploring random tree.

The fundamental concept of the RRT algorithm involves starting with an initial point as the root node and progressively expanding outward to generate child nodes. Once the random tree reaches a point where there are no obstacles between it and the endpoint, the algorithm connects to the endpoint to complete the path search. The sampling and searching process of the random tree is illustrated in Figure 3.

The advantage of the traditional RRT algorithm is that it does not require complex processing of the environment model for path planning. However, its disadvantage lies in the fact that the path is generated through random sampling, resulting in a tortuous path that is usually not optimal and can be time-consuming.

We designed an experiment to demonstrate the powerful obstacle-avoidance capabilities of RRT: A 3D space was configured with multiple rectangular obstacles of varying sizes. The start and end points were placed diagonally opposite in the 3D space, requiring the path to navigate around the obstacles multiple times. The 3D space had dimensions of 1,000 * 1,000 * 1,000, with the start and end points located at (0, 0, 0) and (1,000, 1,000, 1,000), respectively.

Using random sampling, RRT generated an initial path that avoided the obstacles, consisting of several key path points. Blue line segments represented the tree expansion paths generated by RRT through random sampling and connection, which gradually extended until reaching the target point. The green line segment indicated the final obstacle-avoiding path, representing the optimal route segment between the start and end points. The red regions corresponded to obstacle zones distributed within the 3D space, which the path successfully navigated around. The results are shown in Figure 4.

The RRT tree incrementally expands by randomly sampling points, generating numerous blue line segments that cover most of the feasible search space. Through random sampling and collision detection, RRT ensures that the path completely avoids all obstacles. In a complex 3D environment, the algorithm can quickly identify a path from the start point to the endpoint. The initial blue path successfully avoids all obstacles, while the green path represents the shortest route, further demonstrating the effectiveness of the RRT algorithm. However, the overall path is relatively long and exhibits noticeable jagged characteristics, indicating the need for further smoothing to optimize the trajectory.

3.2. Rapid search randomized tree algorithm

PSO algorithm[15] is an evolutionary computation technique known for its simplicity and ease of implementation, requiring minimal parameter adjustments and offering fast convergence. PSO has been widely applied in function optimization, neural network training, fuzzy system control, and other applications traditionally tackled by genetic algorithms. It boasts advantages such as rapid convergence, fewer parameters, and straightforward implementation (particularly in high-dimensional optimization problems, where it converges to the optimal solution faster than genetic algorithms). However, PSO also has the drawback of potentially falling

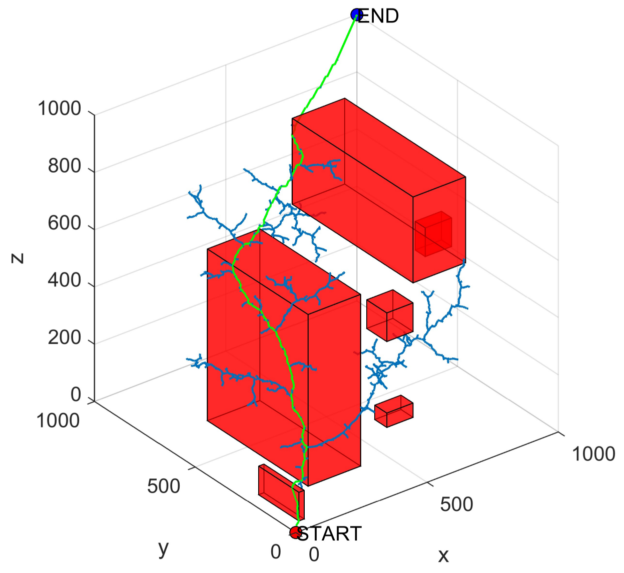

Figure 4. RRT algorithm in three-dimensional space. RRT: Rapidly-exploring random tree.

into local optima, making it highly dependent on effective initialization.

The PSO algorithm originates from studying bird flocking behavior. The basic principle is to find the optimal solution through information sharing and collaborative screening among individuals in the population. Particles have two main attributes: velocity and position. Velocity is a vector that includes both magnitude and direction, while position indicates the coordinates where the particle is currently located. The optimization problem is represented by the positions of the particles in the search space. Each particle has its position and velocity, which determine its current position, distance, and flight direction. Through continuous evaluation of fitness, particles iteratively search for the optimal position.

The particle velocity and position update are calculated using:

$$v_{id}^{k+1} = wv_{id}^{k} + c_1 r_1 \left(p_{id,pbest}^{k} - x_{id}^{k}\right) + c_2 r_2 \left(p_{d,gbest}^{k} - x_{id}^{k}\right) \tag{16}$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \tag{17}$$

where k represents the number of iterations, i indicates the particle serial number, and d means the particle dimension. w stands for the inertia weight, c_1 is the individual learning factor, c_2 points to the population learning factor, and r_1, r_2 are random numbers in the interval [0, 1] to increase the randomness of the search. x represents the position vector, and v corresponds to the velocity vector. $p_{id,pbest}^{k}$ indicates the historical optimal position of particle i in the $d-th$ dimension in the $k-th$ iteration, i.e., the optimal solution obtained by the search of the $i-th$ particle after the $k-th$ iteration. $p_{d,gbest}^{k}$ represents the historical optimal position of the population in the $d-th$ dimension in the $k-th$ iteration, i.e., the optimal solution for the whole population of particles after the $k-th$ iteration.

3.3. Improved PSO algorithm

3.3.1 Initialization of chaotic sequence

The choice of initialization strategy determines the distribution of the initial population in the solution space. In the traditional particle swarm algorithm, the initial population is generated randomly. While this method is simple and easy to implement, it does not ensure a uniform distribution of individuals in the search space, which can negatively affect the algorithm's search speed and optimization performance. Therefore, chaotic sequences are introduced during the initialization phase to ensure a uniform distribution of the initial solutions.

Key parameters in chaotic mappings play a crucial role in determining the system's dynamic behavior, and their selection directly influences the distribution of the initial population. Specifically:

Control Parameter (r): For the proposed composite chaotic system, the value of r governs the chaotic behavior of the system. Lower $r(r < 0.5)$ values generally lead to periodic behavior in the chaotic mapping. When a smaller r is chosen, the generated sequences cluster within a limited search space region, thereby diminishing the algorithm's exploratory potential. In contrast, higher $r(r > 0.8)$ values promote chaotic behavior, facilitating the generation of more widely dispersed initial solutions. This dispersion enhances the diversity of the initial population, thereby improving the algorithm's global search capability.

Initial Value ($x0$): To achieve a well-distributed initial population, an appropriate initial value ($x0$) must be carefully selected. It is imperative to ensure that $x0$ is uniformly distributed across the entire search space to maximize the coverage and effectiveness of the algorithm.

Chaotic systems exhibit ergodicity, meaning they can cover nearly all states within a certain range. By introducing chaotic sequences during the population initialization phase of the improved PSO [sine-tent-cosine particle swarm optimization (SPSO)] algorithm, more uniformly distributed initial particles can be generated. This uniformity and diversity improve the algorithm's ability to cover the search space, addressing the problem of unexplored regions. In high-dimensional optimization problems, chaotic sequences help prevent the clustering or insufficient coverage of local regions often caused by traditional random methods. A more evenly distributed population lays a solid foundation for subsequent global searches. The pseudo-randomness and sensitivity to initial conditions inherent in chaotic systems enhance the ability of the particles to "jump" during the global search phase. In multi-modal problems, chaotic sequences assist particles in escaping local optima, thereby strengthening global search capabilities.

The fitness function value of the random numbers generated using chaotic mapping shows significant improvement compared to those generated by conventional uniformly distributed random number generators. Replacing the conventional method with chaotic mapping yields better results, especially when the search space contains many local optima, making it easier to find the global optimal solution. Chaotic mapping enhances the randomness and diversity of particles.

Chaos mapping is used to generate chaotic sequences, which are sequences of randomness produced by simple deterministic systems. These sequences describe complex chaotic phenomena that are sensitive to initial conditions, non-periodic, and internally stochastic, generated by deterministic nonlinear systems. In optimization, chaotic mapping can serve as an alternative to pseudo-random number generators to produce chaotic numbers between 0 and 1. Experimental results have demonstrated that using chaotic sequences for population initialization, selection, crossover, and mutation influences the algorithm process and often achieves better results than pseudo-random numbers.

Chaotic systems[16] can be classified into low- and high-dimensional categories. High-dimensional systems are more complex, with numerous control parameters and greater computational demands. In contrast, low-dimensional systems are simpler, require fewer control parameters and are easy to implement. Nevertheless, certain limitations are inherent to low-dimensional systems, such as constrained chaotic behavior, discontinuous chaotic intervals, and non-uniform data distribution within the generated chaotic sequences.

To develop chaotic systems with better performance, Sine, Tent, and Cosine chaotic mappings are combined to form a new composite chaotic system. This system effectively overcomes the shortcomings of low-dimensional chaos and is less complex and easier to implement than high-dimensional chaos, which is calculated by :

$$x(i+1) = \begin{cases} \cos\left(\pi\left(r\sin\left(\pi x(i)\right) + 2(1-r)x(i) - 0.5\right)\right), & \text{if } x(i) < 0.5 \\ \cos\left(\pi\left(r\sin\left(\pi x(i)\right) + 2(1-r)(1-x(i)) - 0.5\right)\right), & \text{else} \end{cases} \tag{18}$$

where $r = 0.7$, $x(i)$ and $x(i+1)$ represent the state variables before and after the chaotic mapping, respectively.

To verify whether the chaotic mapping used in this study generates sequences with sufficient uniformity, we evaluated its ability to initialize particle positions reasonably within the search space.

The most commonly used chaotic initialization methods include logistic, circle, and singer mappings. However, the hybrid combination of Sine, Tent, and Cosine chaotic mappings enhances randomness, which introduces diversity during the population initialization phase. This helps the algorithm explore a broader solution space and reduces the likelihood of being trapped in local optima.

Based on the characteristics of the hybrid chaotic mapping, chaotic sequences are generated in the range of [0, 1]. These sequences are then scaled to match the upper and lower bounds of the search space, completing the population initialization.

The logistic chaotic mapping is expressed as follows:

$$x_{i+1} = ax_i\left(1 - x_i\right) \tag{19}$$

Where a is the control parameter, taking values in the range (0, 4]. The chaotic behavior becomes more pronounced as a increases, reaching a state of full chaos when $a = 4$.

The Circle chaotic mapping is expressed as follows:

$$x_{i+1} = \mod\left(x_i + b - \left(\frac{a}{2\pi}\right)\sin\left(2\pi x_i\right), 1\right) \tag{20}$$

Where a and b are control parameters, with commonly used values of $a = 0.5$ and $b = 0.2$. The mod function represents the modulus operation, ensuring that the output remains within the interval [0, 1].

The Singer chaotic mapping is expressed as follows:

$$x_{i+1} = a\left(7.86xi - 23.31x_i^2 + 28.75x_i^3 - 13.3028275x_i^4\right) \tag{21}$$

Where a is the control parameter, taking values in the range [0.9, 1.08]. This range ensures the mapping exhibits chaotic behavior suitable for complex optimization tasks.

To verify that the hybrid chaotic mapping initialization method produces results with better randomness and diversity, as well as more uniform distribution compared to logistic chaotic mapping, circle chaotic mapping, singer chaotic mapping, and random initialization, a comparison was conducted using frequency distribution histograms and scatter plots.

The settings for the experiment were: $lb = 0, ub = 1$, and $NP = 1{,}000$. The interval [0, 1] was divided into ten equal-length subintervals, and the population distribution frequency for each interval was calculated. The results are shown in Figure 5.

The frequency histogram demonstrates that the sequences generated by the hybrid chaotic mapping are more uniformly distributed within the interval [0, 1] compared to other mappings, such as Logistic, Circle, and Singer. The frequency distribution closely aligns with the theoretical values. The scatter plot further validates that the points generated by the hybrid chaotic mapping exhibit no significant clustering and uniformly cover the entire search space, effectively enhancing population diversity during the initialization phase. In contrast, Logistic mapping shows data clustering in certain intervals, which increases the likelihood of searching only local regions. Circle and Singer mappings produce sequences with uneven distribution near the boundaries, potentially limiting global exploration capabilities. Hybrid chaotic mapping combines the advantages of multiple mappings, achieving the best uniformity in distribution and avoiding the limitations of individual chaotic mappings.

By introducing hybrid chaotic mapping as the initialization method, population diversity is significantly improved, enhancing the algorithm's global search capability and reducing the risk of being trapped in local optima.

3.3.2 Improved inertia weights and learning factors

The values of inertia weights[17] www and learning factors ccc in the PSO algorithm directly affect its convergence performance. Addressing the shortcomings of the basic particle swarm algorithm - specifically its slow convergence speed for complex functions and low optimization accuracy - larger weights[18] enhance the algorithm's global search capability, while smaller weights improve local search ability.

To improve these aspects, this paper introduces adaptive inertia weights in the global search stage, which decrease nonlinearly with the number of evolutionary generations. This approach enhances global searching ability in the early stages and ensures particle diversity in the latter stages by obtaining smaller random values, thus improving the solution's accuracy. The proposed inertia weight function is expressed as follows:

$$\begin{cases} w = \dfrac{1}{\left[(t+1)^{(\alpha+\beta)}\right]^3} \cdot \left(\dfrac{\alpha \cdot t^{\alpha} + \beta \cdot t^{\beta}}{T_{\max}}\right) & t \leqslant 0.6T_{\max} \\ w = (w_{\max} - w_{\min}) + 0.1 \cdot rand & t > 0.6T_{\max} \end{cases} \tag{22}$$

where $\alpha = 3$, $\beta = 5$, w_{max} is the maximum inertia weight, w_{min} is the minimum inertia weight, t is the current number of iterations of the particle, T_{max} is the maximum number of iterations, and $rand$ is a random number between (0, 1).

The learning factors c_1 and c_2 of the traditional PSO algorithm are fixed values, which can affect its opti-

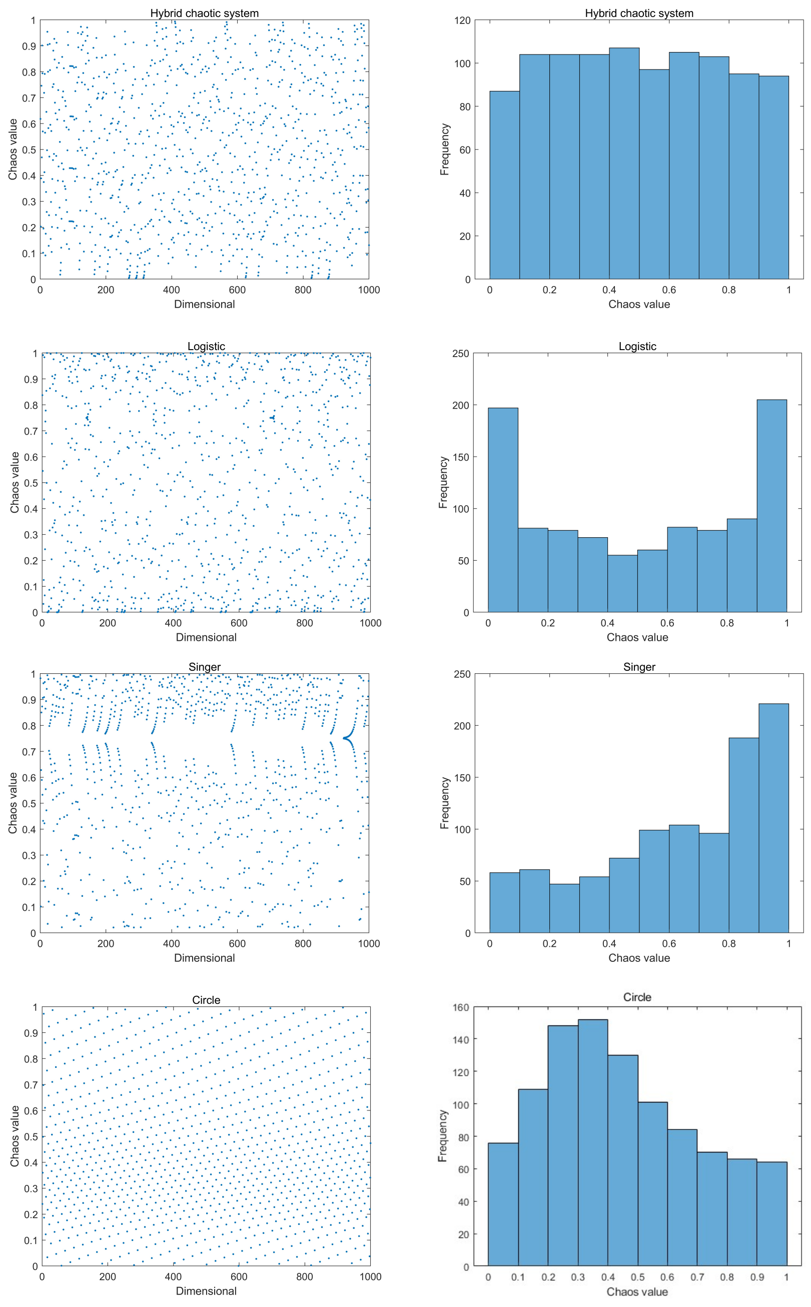

Figure 5. Comparison of various chaotic mappings.

mization speed and solving accuracy. The factor c_1 represents the particle's self-learning and summarization ability, reflecting the individual's self-perception, while c_2 represents the particle's ability to learn from better-performing particles, reflecting the particle's social perception.

Since a larger c_1 improves global search capability and a larger c_2 enhances local search ability, it is advantageous to set c_1 large and c_2 small in the pre-search stage, and c_1 small and c_2 large in the post-search stage. Therefore, c_1 is set to decrease linearly with the number of iterations, while c_2 increases linearly. This introduces asynchronous learning factors, defined by:

$$\begin{cases} c_1 = c_{1,start} + \left(c_{1,end} - c_{1,start}\right) \dfrac{t}{T_{\max}} \\ c_2 = c_{2,start} + \left(c_{2,end} - c_{2,start}\right) \dfrac{t}{T_{\max}} \end{cases} \tag{23}$$

Where $c_{1,start} = 2.5, c_{1,end} = 0.5, c_{2,start} = 0.5, c_{2,end} = 2.5$, and t and T_{max} represent the number of current iterations and total iterations, respectively.

3.4. Time-optimal trajectory planning

In the time-optimal trajectory planning of a robotic arm, choosing the appropriate time interval is key for success. Under the premise of satisfying speed constraints, the RRT algorithm is used to obtain the trajectory of the robotic arm. Subsequently, the SPSO algorithm is employed to determine the optimal interpolation time. This process solves the unknown coefficients of the polynomials, enabling the robotic arm to achieve obstacle avoidance and a time-optimized trajectory. By optimizing each joint of the robotic arm, the maximum value of the interpolation time interval is selected to ensure that each joint reaches the target angle.

Four joint angle values are selected: one at the end of the RRT algorithm, one intermediate value between this point and the start point, and the start and end points. This paper aims to enhance the obstacle avoidance capability, efficiency, and smooth operation of the robotic arm, ensuring the shortest possible time for all joint movements. The optimization objective is to satisfy all constraints while minimizing the joint movement time, as determined by:

$$f(t) = \min \sum_{i=1}^{n} \left(t_{i1} + t_{i2} + t_{i3}\right) \tag{24}$$

$$\begin{cases} \max\left(\left|V_{i1}\right|\right) \leqslant V_{\max} \\ \max\left(\left|V_{i2}\right|\right) \leqslant V_{\max} \\ \max\left(\left|V_{i3}\right|\right) \leqslant V_{\max} \end{cases} \tag{25}$$

Where $t_{i,j}$ represents the j th interpolation time under the i th joint, $V_{i,j}$ indicates the actual value of the velocity of the i th joint, and V_{max} is the maximum value of the velocity of the joint. The flowchart of the optimal planning of the robotic arm time incorporating the RRT and SPSO algorithms is shown in Figure 6.

3.5. Algorithm performance test

3.5.1 Compare objects and test functions

To evaluate the performance of the proposed SPSO algorithm, it is compared with the elementary PSO, swallow tern algorithm (STOA)[19], and Sine Cosine algorithm (SCA)[20]. All algorithms were implemented in MATLAB 2023a and executed on a Windows 11 system with 16 GB of memory. For each algorithm, the population size was set to 30, and the number of iterations was 1,000. The minimum (min), mean, maximum (max), standard deviation (std), optimal value, and running time of each algorithm were recorded to assess their ability to find the optimal value.

In this paper, six benchmark test functions are selected to evaluate the performance of the proposed SPSO algorithm. Each algorithm is run independently 30 times on these functions, as listed in Table 2. The experiments tracked the minimum (min), mean, maximum (max), standard deviation (std), optimal value (value), and running time (time) of each algorithm. These metrics are used to evaluate the optimization capabilities of the algorithms. The results are summarized in Table 3.

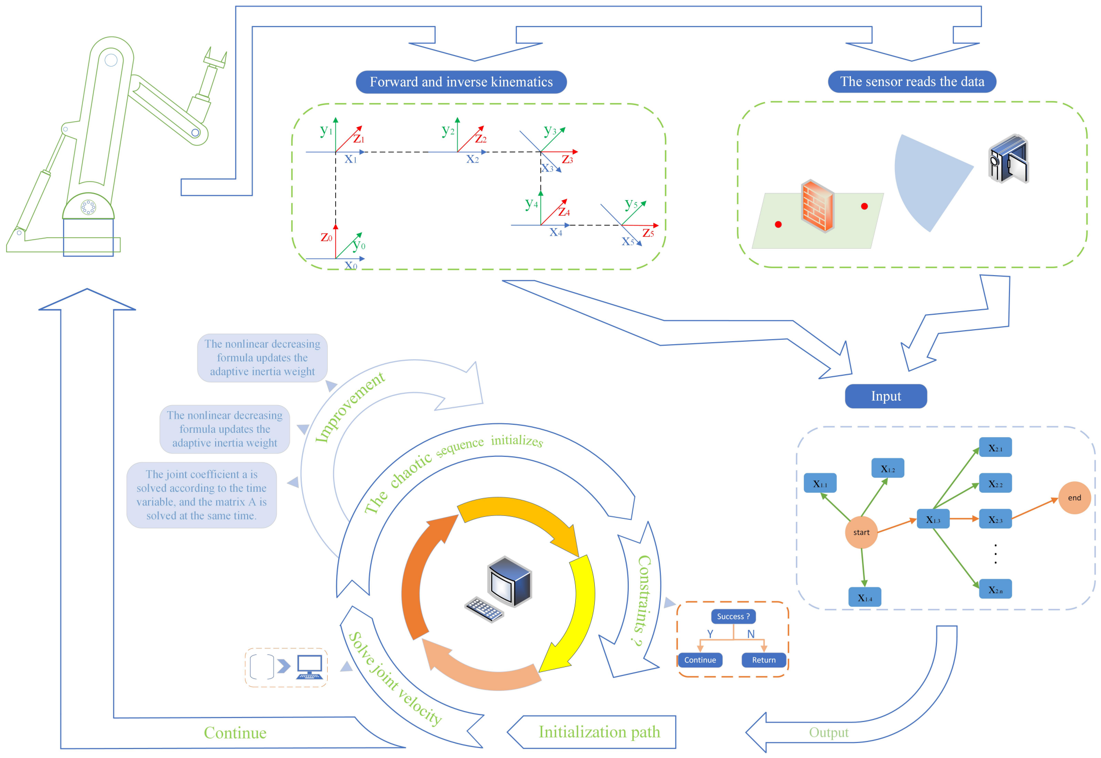
Forward and inverse kinematics
The sensor reads the data
Input
start
end
The nonlinear decreasing formula updates the adaptive inertia weight
The nonlinear decreasing formula updates the adaptive inertia weight
The joint coefficient a is solved according to the time variable, and the matrix A is solved at the same time.
Improvement
The chaotic sequence initializes
Constrains ?
Success ?
Y
N
Continue
Return
Solve joint velocity
Initialization path
Output
Continue

Figure 6. Flow chart of fusion RRT algorithm and SPSO algorithm. RRT: Rapidly-exploring random tree; SPSO: sine-tent-cosine particle swarm optimization.

Table 2. Benchmark function test

Type	Function	Range
F1	$\sum_{i=1}^{n-1}\left[100(x_{i+1}-x_i^2)^2+(x_i-1)^2\right]$	$[-30,30]^n$
F2	$\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	$[-100,100]^n$
F3	$-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	$[-32,32]^n$
F4	$\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$[-600,600]^n$
F5	$\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_i)^6}\right]^{-1}$	$[-65,65]^n$
F6	$\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	$[-5,5]^n$

As can be seen from Table 3, functions F1-F3 are multidimensional test functions with multiple local optimal solutions. These functions simulate the complexity of many real-world problems and test the algorithm's balance between global exploration and local exploitation. Specifically, they assess how well the algorithm avoids falling into local optima and finds global optima during the search process. Functions F4-F6 are hybrid functions, designed to test the algorithm's ability to handle complex hybrid optimization problems. These functions may contain locally optimal solutions of varying shapes, sizes, and distributions, thereby increasing the challenge for the algorithm. The design of these hybrid functions ensures a balance between global and local optima, comprehensively assessing the algorithm's global and local search capabilities.

From Table 3, it is evident that among the multidimensional test functions, F1 is characterized by multiple peaks, making it difficult to find the optimal solution. This causes the SPSO algorithm to converge with less than ideal accuracy during certain iterations, although it still ranks second relative to other algorithms. F1 is a multi-modal function with multiple local optima, simulating the complexity of real-world optimization problems. In the case of the F1 function, SPSO demonstrates the best performance, with the smallest value and optimal solution closest to the global optimum, while also requiring less runtime. In contrast, PSO per-

Table 3. Test results of different algorithms

Function	Indicators	STOA	SCA	PSO	SPSO
	min	2.6964e+01	2.8587e+01	7.6570e+04	**2.3475e+01**
	mean	2.8042e+01	8.6948e+02	1.9469e+05	**1.3225e+02**
F1	max	**2.8831e+01**	6.0236e+03	2.7335e+05	4.1647e+02
	std	**6.4460e-01**	1.7838e+03	4.8347e+04	1.0090e+02
	value	**2.8025e+01**	2.8657e+01	1.3243e+05	2.9172e+01
	time	1.0046e-01	1.0682e-01	5.2805e-02	**5.0901e-02**
	min	1.2693e+00	3.9519e+00	1.1089e+02	**5.4567e-05**
	mean	2.4177e+00	4.5292e+00	1.4866e+02	**1.3594e-02**
F2	max	3.2625e+00	5.8863e+00	1.8546e+02	**7.7506e-02**
	std	4.8232e-01	4.8477e-01	2.0573e+01	**2.1178e-02**
	value	2.5012e+00	4.4582e+00	1.7588e+02	**5.8475e-03**
	time	8.2215e-02	8.7004e-02	3.8469e-02	**3.8137e-02**
	min	1.9955e+01	**1.2653e-03**	8.2444e+00	2.1259e+00
	mean	1.9959e+01	1.5570e+01	8.9821e+00	**4.0529e+00**
F3	max	1.9961e+01	2.0250e+01	9.5201e+00	**7.6328e+00**
	std	**1.6446e-03**	8.4171e+00	3.1175e-01	1.3053e+00
	value	1.9958e+01	2.0194e+01	9.1735e+00	**4.5723e+00**
	time	9.8262e-02	1.1170e-01	5.9892e-02	**4.9187e-02**
	min	**0**	2.5287e-04	1.0250e+00	1.3754e-02
	mean	1.4992e-02	2.3515e-01	1.0368e+00	**1.0752e-02**
F4	max	**6.2563e-02**	8.6754e-01	1.0442e+00	3.1040e-01
	std	1.8722e-02	2.6758e-01	**4.5081e-03**	8.8390e-02
	value	**1.0335e-01**	1.6372e-01	1.0366e+00	5.0236e-00
	time	1.0979e-01	1.1771e-01	6.3239e-02	**5.5914e-02**
	min	9.9800e-01	9.9800e-01	9.9800e-01	**9.9800e-01**
	mean	1.8831e+00	1.5933e+00	4.4893e+00	**2.9710e+00**
F5	max	1.0763e+01	9.9821e+00	1.1719e+01	**7.8740e+00**
	std	2.2412e+00	**9.3277e-01**	3.1922e+00	2.4106e+00
	value	2.9821e+00	9.9806e-01	5.9288e+00	**9.9800e-01**
	time	4.2101e-01	4.2298e-01	4.0591e-01	**3.9334e-01**
	min	4.0428e-04	3.8921e-04	9.3239e-04	**3.0749e-04**
	mean	1.1839e-03	9.9477e-04	1.2723e-03	**5.0467e-04**
F6	max	**1.2302e-03**	1.5386e-03	1.9476e-03	1.3368e-03
	std	**1.8350e-04**	3.6076e-04	2.9019e-04	3.5338e-04
	value	1.2403e-03	7.1507e-04	1.1581e-03	**3.0749e-04**
	time	2.3222e-02	2.8757e-02	1.2793e-02	**1.0753e-02**

Bold indicates that the optimized value is different from others. STOA: Swallow tern algorithm; SCA: Sine Cosine algorithm; PSO: particle swarm optimization; SPSO: sine-tent-cosine particle swarm optimization.

forms poorly, yielding significantly higher minimum values. Such problems challenge the algorithm's ability to balance global exploration and local exploitation, particularly in avoiding entrapment in local optima. As shown in Table 3 and Figure 7, when optimizing the F1 function, SPSO achieves the optimal fitness value of 23.475 in only 400 iterations, while STOA and SCA require approximately 800 iterations, and PSO takes 1,000 iterations. The optimal fitness value achieved by SPSO is significantly better than those of the other algorithms, indicating its superior global search capability. The F1 function is characterized by a multi-peak distribution, which often leads traditional algorithms to become trapped in local optima. However, SPSO effectively avoids this issue through the use of chaotic sequences and dynamic inertia weights.

In the F2 test function, the SPSO algorithm finds the optimal value faster than other algorithms, with the smallest standard deviation and highest stability. F2 is a nonlinear optimization function characterized by complex nonlinear relationships and constraints. It evaluates the algorithm's capability to optimize high-dimensional nonlinear problems, focusing on efficiency and stability in large-scale complex scenarios. For the F2 function, SPSO not only finds the smallest minimum value but also exhibits the lowest standard deviation, indicating the highest stability. Additionally, SPSO achieves the shortest runtime, reflecting its high efficiency. For the F2 function, SPSO finds the optimal solution (fitness value of 0.000054567) in only 300 iterations, outperforming other algorithms in both the number of iterations and the quality of the optimal solution. The F2 function requires high search efficiency, and SPSO achieves rapid convergence by dynamically adjusting the weights of global and local search through adaptive learning factors.

F3 is a function with multiple local optima, designed to test an algorithm's performance in multi-local optima scenarios. This function requires optimization algorithms to avoid local optima and successfully locate the global optimum. On the F3 function, SPSO achieves a balanced performance, finding a smaller optimal value with a lower standard deviation and shorter runtime. This highlights its high stability and efficiency in multi-local optima problems. When optimizing the F3 function, SPSO reaches a fitness value of 2.1259 after 400 iterations, whereas PSO achieves a suboptimal fitness value only after 600 iterations. SPSO also outperforms SCA and STOA in this scenario.

F4 is a hybrid function that combines a global optimum with multiple local optima. Its objective is to evaluate how well an algorithm balances local search and global search in complex optimization spaces. SPSO performs exceptionally well on the F4 function, achieving a minimum value close to the global optimum with the lowest standard deviation, indicating its strong capability in solving hybrid optimization problems. For the F4 function, SPSO identifies the optimal solution (fitness value of 0.013754) in approximately 500 iterations. Although the optimal values obtained by STOA and SCA are similar, SPSO requires significantly fewer iterations. For the F4 hybrid function, the global search capability of SPSO proves particularly advantageous, as its dynamic parameter adjustment effectively prevents premature convergence to local optima.

F5 is a relatively simple optimization problem, typically with a single global optimum. This function tests the algorithm's performance in simpler optimization scenarios. SPSO demonstrates exceptional stability on the F5 function, achieving a minimum value close to the global optimum with shorter runtime and higher stability. For the F5 function, SPSO, SCA, and STOA all eventually converge to a fitness value of 0.998. However, SPSO reaches this value in fewer than 100 iterations, while SCA requires approximately 150 iterations, and STOA takes about 700 iterations.

F6 is a near-optimal solution problem, containing solutions close to the global optimum. It evaluates how quickly an algorithm can converge to near-optimal solutions and handle minor errors. SPSO outperforms other algorithms on the F6 function, achieving the smallest minimum value with shorter runtime, indicating its strong capability in solving near-optimal solution problems. For the F6 chaotic characteristic function, SPSO finds the optimal solution in approximately 300 iterations, achieving a fitness value of 0.000307, which outperforms the other algorithms. This function places high demands on randomness and distribution, and SPSO's chaotic sequence initialization strategy significantly enhances the randomness and coverage of the search process.

In testing hybrid functions, SPSO ranks in the top two overall when solving complex functions. In summary, whether facing multi-peak or hybrid functions, SPSO ranks first overall, demonstrating the shortest time, greater stability, and smallest standard deviation.

In this paper, the optimization curve of the test function is plotted based on the number of iterations and the adaptation size, as shown in Figure 7. From the convergence curves of functions F1 to F6, it is evident that SPSO exhibits fast convergence speed and strong optimization capabilities. Specifically, for the F1 function, SPSO reaches the optimal value in approximately 400 iterations, whereas STOA reaches the optimal value in about 800 iterations. For the F5 function, all the algorithms eventually stabilize, but the SPSO algorithm converges faster and achieves better results compared to the other algorithms.

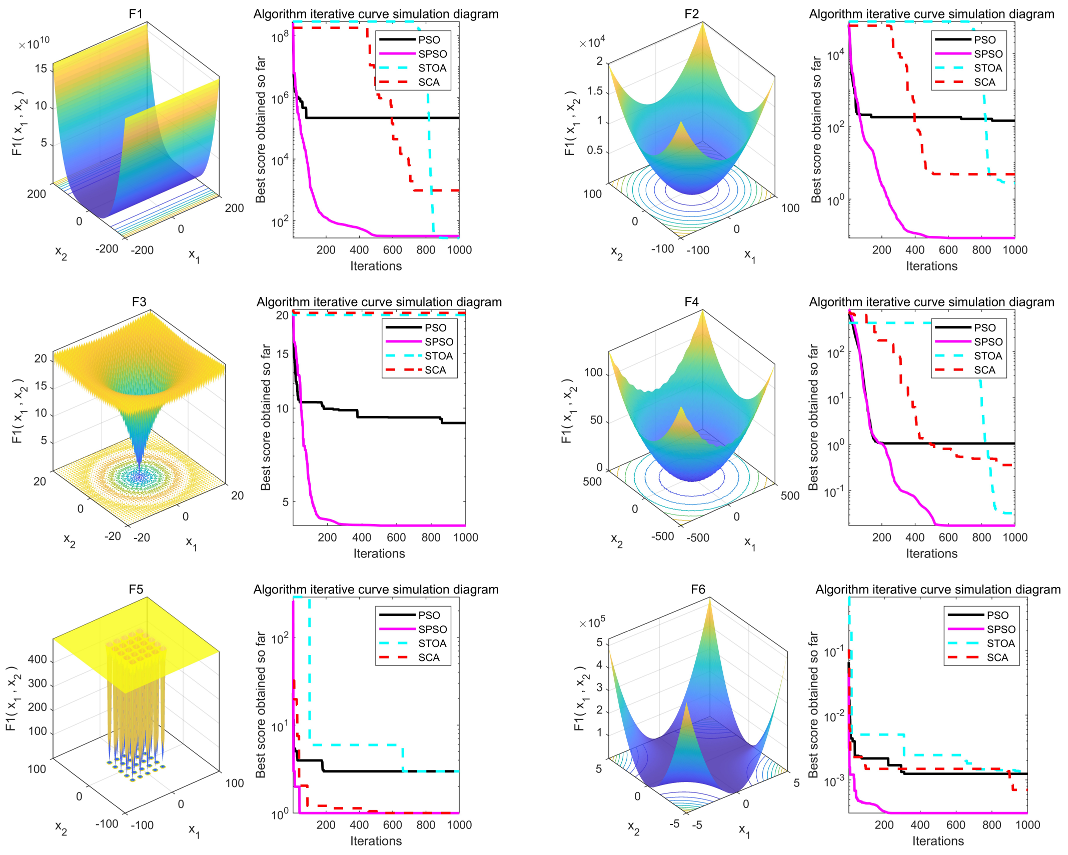

Figure 7. Comparison of various chaotic mappings.

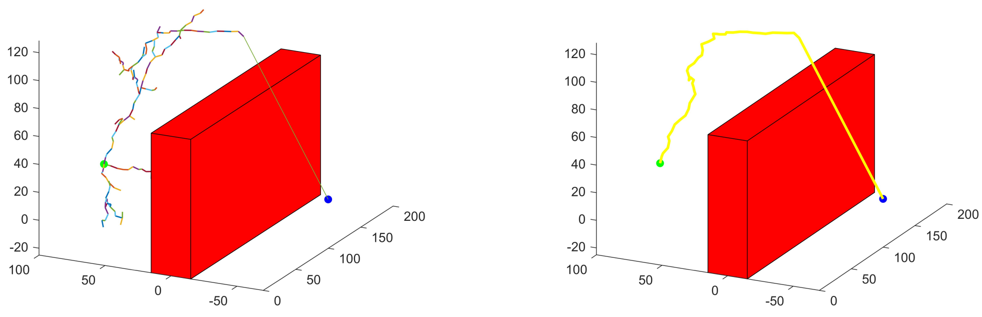

Figure 8. Comparison of various chaotic mappings.

4. SIMULATION EXPERIMENT AND ANALYSIS

To verify the obstacle avoidance capability of the RRT-SPSO algorithm and its time-saving potential, the PUMA560 robotic arm is used as the research object in this study. Simulation experiments are conducted in MATLAB. Initially, the start and target points are determined in the Cartesian space. The RRT algorithm is employed to avoid obstacles in this space, resulting in a path that reaches the target from the initial point, as illustrated in Figure 8.

Table 4. Joint position information corresponding to each path point

Waypoint	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Path point 1	3.7851	2.2724	0.1196	3.1416	2.3920	-0.6435
Path point 2	3.4906	2.0046	-0.2843	3.1416	1.7203	-0.3490
Path point 3	2.8927	1.9561	-0.6257	3.1416	1.3303	0.2489
Path point 4	2.3663	2.1315	0.3768	3.1416	2.5083	0.7753

Table 5. Joint position information corresponding to each path point

Initialization	Joint	1	2	3	4	5	6
1	Time 1 (s)	0.29569	0.26779	0.51552	0.1	0.75865	0.29774
1	Time 2 (s)	0.23916	0.1	0.4297	0.1	0.41572	0.21682
1	Time 3 (s)	0.51522	0.16749	1.177	0.1	1.2283	0.5132
3	Total time t (s)	1.0501	0.53528	2.1222	0.3	2.4027	1.0278

In the resulting path, trajectory planning for the robotic arm is performed in Cartesian space to determine the four path nodes through which the end of the robotic arm passes. These four points include the initial point, the target point, the point at which the RRT algorithm ends, and an intermediate node between the initial point and the endpoint of the RRT algorithm. They are then converted to the corresponding angle values for each joint in the joint space using inverse kinematics, as shown in Table 4.

Subsequently, the SPSO algorithm in RRT-SPSO is simulated in MATLAB for each joint to perform the time-optimal solution under the conditions of robotic arm motion. The variation of each time period, the total time variation, and the number of iterations for each joint after optimization are shown in Figure 9.

As illustrated in Figure 9, the SPSO algorithm reaches the optimal value after 20 iterations, and all times are reduced compared to the original values. The optimal movement times of all the joints of the robotic arm at various stages are obtained, as shown in Table 5.

To ensure the smooth operation of the robotic arm joints and complete the corresponding movement tasks, the movement times of each joint should be synchronized. Although the required motion times for each joint differ, selecting the maximum time value for each segment ensures that all six joints complete their motions simultaneously. This synchronization effectively prevents motion-induced vibrations or errors caused by asynchrony between joints. After optimization, the time discrepancies among the joints are significantly reduced, resulting in a more uniform time distribution and noticeably improved synchronization and efficiency of the robotic arm. In particular, joints with longer motion times (e.g., Joint 5) are significantly optimized, enhancing overall efficiency. The total motion time is reduced by nearly 20%, demonstrating the effectiveness of the proposed method in minimizing the robotic arm's motion time and improving operational efficiency. The improved trajectory planning uses 3-5-3 polynomial interpolation to smooth the motion trajectory. The smoothed curves reduce vibrations during the robotic arm's motion, contributing to greater load stability. Before optimization, the motion curves (velocity and acceleration) exhibited abrupt changes, potentially causing arm instability and jitter. After optimization, the motion curves are smoother, with continuous velocity and acceleration transitions, free of abrupt changes. Therefore, the maximum time value of the six joints in each section of the trajectory is selected. As shown in Table 4, the times for the three phases are $t1 = 0.75865$, $t2 = 0.4297$, and $t3 = 1.2283$, with a total time of $t = 2.41665$. Compared to the initial value, the time is reduced by 0.58338 s, approximately 19% shorter, demonstrating that SPSO can effectively improve the working efficiency of the robotic arm. During trajectory planning, the displacement, velocity, and acceleration of each joint can be determined based on the time, as shown in Figure 10.

The RRT algorithm generates an initial path through random sampling in obstacle-laden environments, en-

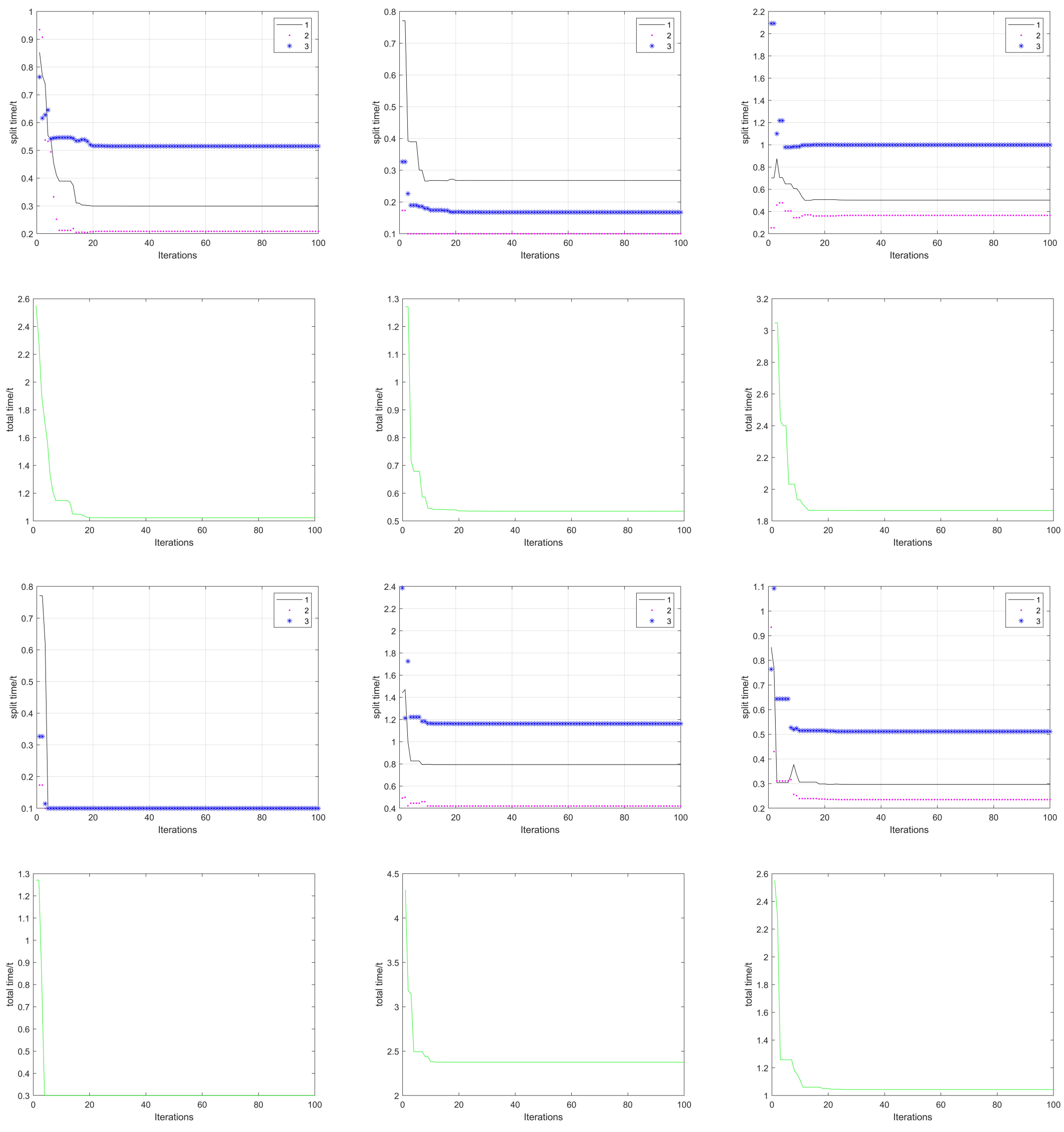

Figure 9. Optimal particle position iteration.

suring path feasibility in complex scenarios. While the initial path avoids obstacles, it is often longer and less smooth, necessitating further optimization. RRT provides a foundational obstacle-avoiding path that reduces the computational burden for subsequent path optimization. This foundation allows the SPSO algorithm to focus on refining a feasible path rather than exploring the entire search space from scratch.

Once the RRT algorithm generates the preliminary path, SPSO is employed to optimize the path duration. By optimizing the interpolation of the path, SPSO reduces unnecessary turns, smooths the path, and minimizes the robotic arm's motion time, thereby enhancing operational efficiency. Specifically, SPSO optimizes the time interpolation of the path, enabling each joint of the robotic arm to complete its assigned task in the shortest possible time. Through joint-specific motion optimization, appropriate time intervals are selected to minimize motion time while ensuring efficient task completion. To prevent the robotic arm from operating too slowly or quickly at certain stages, the time intervals are fine-tuned using SPSO, further reducing time consumption. Time optimization is performed with a focus on synchronizing the motion times of all joints. The slowest joint is selected as the controlling time for the entire robotic arm, avoiding time wastage caused by unsynchronized

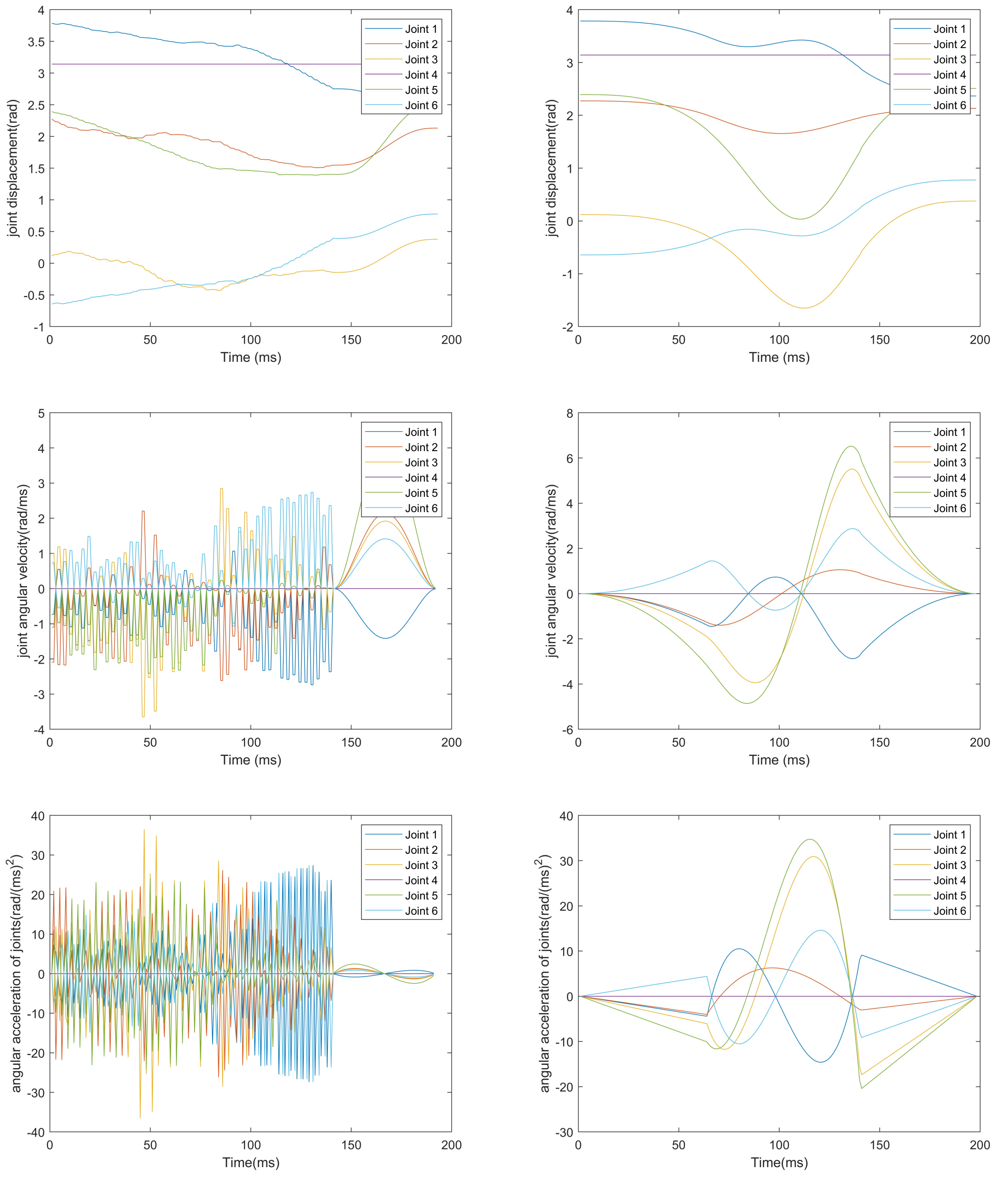

Figure 10. Comparison chart before and after optimization.

joint motions.

A 3-5-3 polynomial interpolation method, comprising cubic, quintic, and cubic polynomials, is employed for trajectory planning. This approach ensures trajectory smoothness, enabling continuous variations in joint angles, velocities, and accelerations. The use of 3-5-3 polynomial interpolation avoids the oscillatory effects often associated with higher-order polynomial interpolation, thereby reducing energy waste and unnecessary time consumption during motion.

As shown in Figure 10, the joint movements before optimization are erratic, causing significant jitter during operation and abrupt speed changes. This can lead to the robotic arm dropping objects during handling.

Table 6. Total time comparison for different algorithms

	Total time t (s)
Initialization	3
Improved RRT	2.0718
RRT-SPSO	2.21675
NSGA-II	2.38035

RRT: Rapidly-exploring random tree; SPSO: sine-tent-cosine particle swarm optimization.

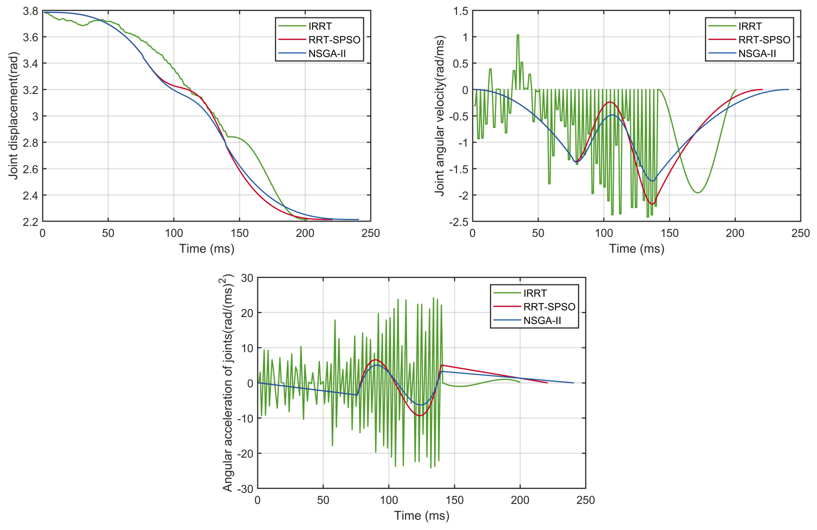

Figure 11. Comparison of various algorithms.

However, after optimization using the RRT-SPSO algorithm, the joint movements, velocity, and acceleration curves of the robotic arm become smooth and continuous without sudden changes. This reduces joint jitter and ensures the robotic arm maintains good stability during trajectory planning and movement.

Recent studies from the past few years were selected for comparison to validate the superiority of the proposed algorithm. The first reference focuses on the trajectory and runtime of an improved RRT (IRRT)[21] algorithm in a 3D space, while the second reference employs an improved non-dominated sorting genetic algorithm (NSGA-II)[22] to optimize time-optimal trajectory planning for robotic arms. Both studies involve time-optimal trajectory planning for point-to-point motion in a 3D space, which closely aligns with the scope of this paper. Therefore, these two references were selected for comparison.

The first comparison involves runtime. A unified experimental environment was set up to evaluate the time required to move from one point to another in a 3D space. The experiment aims to optimize the robotic arm's motion time and assess the performance of each algorithm in terms of trajectory smoothness and dynamic characteristics in 3D space. The results for Joint 1 are presented in Table 6.

Below are the displacement, velocity, and acceleration profiles for Joint 1, as shown in Figure 11.

The IRRT demonstrated excellent performance in terms of total runtime (2.0718 s), reducing the time by approximately 31% compared to the initial approach. The runtime of the algorithm proposed in this paper is 2.21675 s, which, although slightly less efficient than the IRRT, outperforms NSGA-II and shows advantages in several performance metrics.

From the experimental graphs, it can be observed that the displacement curves of RRT-SPSO and NSGA-II are more continuous, with fewer path turning points, indicating superior path optimization by these two algorithms. Additionally, the velocity curves of RRT-SPSO and NSGA-II are smoother, demonstrating better stability in maintaining the robotic arm's motion dynamics. In contrast, the velocity curve of the IRRT exhibits more noticeable fluctuations, likely due to its strong local exploitation capabilities but insufficient global smoothness.

Furthermore, an analysis of the acceleration curves shows that RRT-SPSO and NSGA-II achieve the smoothest profiles, with minimal abrupt changes, highlighting their superior trajectory optimization effectiveness.

Overall, the proposed algorithm achieves a good balance between real-time performance and smoothness, addressing issues in the IRRT such as path irregularities and abrupt velocity and acceleration changes. By applying trajectory smoothing techniques (e.g., 3-5-3 polynomial interpolation), RRT-SPSO effectively reduces velocity and acceleration fluctuations.

In addition, compared to the limitation of NSGA-II, which requires manual path input, the proposed algorithm demonstrates greater autonomy by automatically avoiding obstacles, optimizing the path, and achieving time-optimal solutions. This autonomy makes the proposed algorithm more suitable and reliable in dynamic obstacle environments.

5. CONCLUSION

In this study, a six-degree-of-freedom robotic arm is taken as the research object, employing the RRT algorithm for trajectory planning and an SPSO algorithm based on 3-5-3 polynomials for time-optimal trajectory planning. The following conclusions are drawn from the simulation experiments:

Obstacle Avoidance: The RRT algorithm effectively avoids obstacles encountered during the operation of the robotic arm, deriving a feasible path.

Enhanced PSO Algorithm: The introduction of chaotic mapping enhances the randomness and diversity of particles, ensuring a uniform distribution of the initial solution. Additionally, the use of nonlinearly decreasing adaptive inertia weights and asynchronous learning factors further refines the algorithm. These adaptive weights are divided into two parts: the first enhances global searching ability, while the second improves local search accuracy. This approach mitigates the issues of slow convergence and susceptibility to local optima in traditional PSO algorithms.

Optimization Results: The goal of shortening trajectory planning time is achieved, with simulation experiments demonstrating a 19% reduction in time. The use of 3-5-3 polynomials ensures a smooth trajectory, preventing sudden changes in velocity and acceleration during motion.

These findings highlight the effectiveness of the RRT-SPSO algorithm in improving the efficiency and stability of robotic arm trajectory planning, making it a promising approach for industrial applications.

Industrial scenarios in real-world applications are often characterized by uncertainties. Robots may encounter various dynamic factors during task execution, such as changes in the operational environment, adjustments to task parameters, or external disturbances. In some industrial applications, multiple robots may be required to work collaboratively (e.g., cooperative handling, assembly line operations). In such cases, beyond the path planning and time optimization for individual robots, coordination and collaboration among robots must also be addressed.

For instance, multiple robots need to avoid mutual collisions while efficiently completing their tasks. Robust optimization techniques can be employed to ensure that the algorithm remains effective even under environmental changes or uncertainties. This involves incorporating the dynamic changes of the model into consideration or employing optimization algorithms with fault-tolerant mechanisms. Moreover, multi-objective optimization and cooperative optimization strategies can be introduced to address inter-robot collaboration. The optimization problems of multiple robots can be formulated as a large-scale multi-objective optimization problem, which can then be solved using an SPSO algorithm for joint optimization.

DECLARATIONS

Acknowledgments
The authors would like to express their sincere gratitude to all those who contributed to the successful completion of this paper.

Authors' contributions
Conceived and proposed this study: Wang Z
Wrote the manuscript, carried out simulation operation, and collected data: Pang C
Provided key feedback for this study and assisted in the completion of the experiment: Sui J
Provides data support and assists in drawing flowcharts: Zhao G
Checked the results and assisted in the improvement of the manuscript: Wu W
Supervises the overall research direction: Xu L

Availability of data and materials
Some results of supporting the study are presented in the Supplementary Materials. Other raw data that support the findings of this study are available from the corresponding author upon reasonable request.

Financial support and sponsorship
None.

Conflicts of interest
All authors declared that there are no conflicts of interest.

Ethical approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

REFERENCES

1. Arents J, Greitans M. Smart industrial robot control trends, challenges and opportunities within manufacturing. *Appl Sci* 2022;12:937. DOI
2. Lee CC, Qin S, Li Y. Does industrial robot application promote green technology innovation in the manufacturing industry? *Technol Forecast Soc* 2022;183:121893. DOI
3. Li Z, Li S, Luo X. An overview of calibration technology of industrial robots. *IEEE/CAA J Autom Sin* 2021;8:23–36. DOI

4. Kim J, Croft EA. Online near time-optimal trajectory planning for industrial robots. *Robot Cim Int Manuf* 2019;58:158–71. DOI
5. Ji H, Jin H. Application of multi-strategy improved sparrow algorithm in trajectory location of underwater telescopic boom. *Mach Tool Hydraul* 2023;51:50–6. DOI
6. Wu P, Wang Z, Jing H, Zhao P. Optimal time - jerk trajectory planning for delta parallel robot based on improved butterfly optimization algorithm. *Appl Sci* 2022;12:8145. DOI
7. Zuo G, Li M, Zheng B. Optimal trajectory planning for robotic arms based on an improved adaptive multiobjective particle swarm algorithm. *Exp Technol Manag* 2024;41:184-91. DOI
8. Zhang X, Liu J, Li Y. An obstacle avoidance algorithm for space hyper-redundant manipulators using combination of RRT and shape control method. *Robotica* 2022;40:1036–69. DOI
9. Rybus T. Point-to-point motion planning of a free-floating space manipulator using the rapidly-exploring random trees (RRT) method. *Robotica* 2020;38:957–82. DOI
10. Mizuno N, Ohno K, Hamada R, et al. Enhanced path smoothing based on conjugate gradient descent for firefighting robots in petrochemical complexes. *Adv Robot* 2019;33:687–98. DOI
11. Hsu HK, Huang HP, Huang MB. A real-time optimal energy-saving walking pattern generator based on gradient descent method and linear quadratic control. *Adv Robot* 2019;33:487–507. DOI
12. Lu S, Ding B, Li Y. Minimum-jerk trajectory planning pertaining to a translational 3-degree-of-freedom parallel manipulator through piecewise quintic polynomials interpolation. *Adv Mech Eng* 2020;12:1687814020913667. DOI
13. Long Z, Li X, Shuai T, Wen F, Feng W, Liang C. Review of research state of trajectory planning for industrial robots. *Mech Sci Technol Aerosp Eng* 2021;40:853–62. DOI
14. Jiang L, Liu S, Cui Y, Jiang H. Path planning for robotic manipulator in complex multi-obstacle environment based on improved_RRT. *IEEE/ASME T Mech* 2022;27:4774–85. DOI
15. Dadgar M, Jafari S, Hamzeh A. A PSO-based multi-robot cooperation method for target searching in unknown environments. *Neurocomputing* 2016;177:62–74. DOI
16. Qin QX, Liang ZY, Xu Y. Image encryption algorithm based on logistic-tent chaotic mapping and bit plane. *J Dalian Minzu Univ* 2022;24:245–52. Available from: https://xueshu.baidu.com/usercenter/paper/show?paperid=1g2b0080ce780gh0dt5g0md0f1413195&site=xueshu_se&hitarticle=1. [Last accessed on 28 Dec 2024]
17. Nickabadi A, Mehdi Ebadzadeh M, Safabakhsh R. A novel particle swarm optimization algorithm with adaptive inertia weight. *Appl Soft Comput* 2011;11:3658–70. DOI
18. Pluhacek M, Senkerik R, Davendra D, Kominkova Oplatkova Z, Zelinka I. On the behavior and performance of chaos driven PSO algorithm with inertia weight. *Comput Math Appl* 2013;66:122–34. DOI
19. Dhiman G, Kaur A. STOA: a bio-inspired based optimization algorithm for industrial engineering problems. *Eng Appl Artif Intell* 2019;82:148–74. DOI
20. Mirjalili S. SCA: a sine cosine algorithm for solving optimization problems. *Knowl Based Syst* 2016;96:120–33. DOI
21. Li J, Li C, Chen T, Zhang Y. Improved RRT algorithm for AUV target search in unknown 3D environment. *J Mar Sci Eng* 2022;10:826. DOI
22. Hou J, Du J, Chen Z. Time-optimal trajectory planning for the manipulator based on improved non-dominated sorting genetic algorithm II. *Appl Sci* 2023;13:6757. DOI