Optimal Trajectory Planning for Robotic Arm Based on Improved Dynamic Multi-Population Particle Swarm Optimization Algorithm

Rong Wu[1], Yong Yang[2], Xiaotong Yao[3], Nannan Lu[4]
Electronics and Information Engineering, Gansu Province Microelectronics Industry Research Institute[1]
Gansu Province Integrated Circuit Industry Research Institute[1]
Lanzhou Jiaotong University, 730070, Lanzhou City, Gansu Province, China[2, 3, 4]

***Abstract*—In response to the problem of easy falling into local optima and low execution efficiency of the basic particle swarm optimization algorithm for 6-degree-of-freedom robots under kinematic constraints, a trajectory planning method based on an improved dynamic multi-population particle swarm optimization algorithm is proposed. According to the average fitness value, the population is divided into three subpopulations. The subpopulation with fitness values higher than the average is classified as the inferior group, while the subpopulation with fitness values lower than the average is classified as the superior group. An equal number of populations are selected from both to form a mixed group. The inferior group is updated using Gaussian mutation and mixed particles, while the superior group is updated using Levy flight and greedy strategies. The mixed group is updated using improved learning factors and inertia weights. Simulation results demonstrate that the improved dynamic multi-population particle swarm optimization algorithm enhances work efficiency and convergence speed, validating the feasibility and effectiveness of the algorithm.**

Keywords—Particle swarm optimization; Gaussian mutation; mixed particles; levy flight; greedy strategy

I. Introduction

Currently, robotic arms have seen extensive development and application, particularly in the automation of manufacturing processes [1]. With the continuous expansion of applications and increasing demands in the field of robotic arms, more and more researchers have begun to focus on trajectory planning [2]-[3]. Currently, there are mainly two types of trajectory planning: one focuses on optimizing time [4]-[5], aiming to improve the efficiency of robots by optimizing time; the other focuses on optimizing energy [6], aiming to reduce energy consumption of robots by optimizing energy.

Over the years, due to advancements in robotics technology and its increasing ubiquity across various domains, research in robotics has garnered significant attention. In the process of robot motion, trajectory planning has become essential. In recent years, an increasing number of researchers have been introducing intelligent algorithms [7] [8] [9] [10] [11] [12] [13] to identify the optimal motion trajectory for robotic arms. In response to the problem of time-optimal trajectory planning for robotic arms, numerous intelligent optimization algorithms have emerged. However, there is still no single outstanding algorithm, as each of these algorithms has its own advantages and disadvantages. Therefore, further research is still needed to find better solutions.

In order to better address the time optimization problem of robotic arm trajectory planning, this paper adopts an Improved Dynamic Multi-Population Particle Swarm Optimization Algorithm (IDM-PSO) [14], which divides the population into three subpopulations and utilizes strategies such as Levy flight and greedy strategy, Gaussian mutation and mixed particles, and improved learning factors and inertia weights to enhance its global exploration and local exploitation capabilities. By using MATLAB software for simulation, comparing with Particle Swarm Optimization (PSO) algorithm and Artificial Fish Swarm Optimization (AFSA) algorithm, validate the effectiveness and necessity of the algorithms.

II. Establishing a Mathematical Model for Optimizing the Trajectory Time of a Robotic ARM

A. 3-5-3 Segment Polynomial Interpolation Establishment

When using polynomial interpolation for trajectory planning, low-order polynomial interpolation has low computational complexity but does not guarantee continuous acceleration, while high-order polynomial interpolation, although ensuring continuous trajectory, has high computational complexity and may exhibit Runge's phenomenon [15]. In order to ensure that the trajectory planning interpolation of the robotic arm has continuous velocity and acceleration in joint space without discontinuities, a 3-5-3 segment polynomial interpolation is used, with the interpolation function shown in Eq. (1):

$$\begin{cases} P_{i1}(t) = a_{i13}t_1^3 + a_{i12}t_1^2 + a_{i11}t_1 + a_{i10} \\ P_{i2}(t) = a_{i25}t_2^5 + a_{i24}t_2^4 + a_{i23}t_2^3 + a_{i22}t_2^2 + a_{i21}t_2 + a_{i20} \\ P_{i3}(t) = a_{i33}t_3^3 + a_{i32}t_3^2 + a_{i31}t_3 + a_{i30} \end{cases} \tag{1}$$

In the equation, P_{ij} represents the angular displacement of the ith joint in the jth segment of the trajectory planning; t_1, t_2, t_3 represent the motion times of the ith joint for the first, second, and third segments of the robotic arm, respectively.

During the motion of the robotic arm, each joint passes through the initial point X_0, intermediate points X_1, X_2, and the end point X_3. At X_0 and X_3, the velocity and acceleration are set to 0, and at the three overlapping points of the polynomials, the velocity and acceleration are all equal [16]. From the above conditions, the relationships can be expressed as follows:

$$a = A^{-1}b = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \tag{2}$$

In the equation:

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

$$\begin{cases} a_1 = \begin{bmatrix} a_{i13} & a_{i12} & a_{i11} & a_{i10} \end{bmatrix} \\ a_2 = \begin{bmatrix} a_{i25} & a_{i24} & a_{i23} & a_{i22} & a_{i21} & a_{i20} \end{bmatrix} \\ a_3 = \begin{bmatrix} a_{i33} & a_{i32} & a_{i31} & a_{i30} \end{bmatrix} \end{cases} \tag{4}$$

B. Establishing the Objective Function for Time Optimization

In order to reduce the operation time of the robotic arm and improve its efficiency, the objective function is established with time as the optimization target. The objective function is as follows:

$$\mathrm{f} = t_{i1} + t_{i2} + t_{i3} \tag{5}$$

In the equation: $t_{\mathrm{i}1}$, $t_{\mathrm{i}2}$, $t_{\mathrm{i}3}$ represent the motion times of the ith joint in the three segments of trajectory planning. Additionally, in order to reduce uncertainties such as collisions, damage, and loss of control during the robotic arm's motion, velocity and acceleration constraints need to be imposed on the robotic arm. The constraints are as follows:

$$\begin{cases} \left| v_{ij} \right| \le v_{\max} \\ \left| a_{ij} \right| \le a_{\max} \end{cases} \tag{6}$$

In the equation: "v_{ij}" and "a_{ij}" respectively represent the velocity and acceleration of the ith joint of the robotic arm as they vary over time during the jth trajectory segment. "v_{max}" and "a_{max}" respectively represent the maximum allowable velocity and maximum acceleration of the ith joint during its jth trajectory segment.

III. Improved Dynamic Multi-Population Particle Swarm Optimization Algorithm

A. Particle Swarm Optimization Algorithm

The Particle Swarm Optimization (PSO) algorithm is a population-based search algorithm [17]. Suppose in a D-dimensional search space, a population consists of N particles, where the i-th particle represents a D-dimensional vector $x_{i=}(x_{i1}, x_{i2}, \cdots, x_{iD})^T$, representing the position of the i-th particle in the D-dimensional search space. The individual best solution generated by a particle from the start to the end of the iteration is denoted as $P_i = (P_{i1}, P_{i2}, \cdots, P_{iD})^T$, and P_g represents the global best solution of the entire population. The velocities and positions of the particles are randomly initialized. The iterative update formulas for the velocity and position of particle i in the d -th dimension (1≤d≤D) are as follows:

$$v_{id}^{k+1} = \omega \cdot v_{id}^{k} + c_1 r_1 (P_{id} - x_{id}^{k}) + c_2 r_2 (P_g - x_{id}^{k}) \tag{7}$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \tag{8}$$

In the equations: k represents the current iteration number; ω denotes the inertia weight; r_1 and r_2 are random numbers uniformly distributed in the range [0, 1]; c_1 and c_2 are the learning factors; v_{id}^{k+1} and x_{id}^{k+1} represent the updated velocity and position of the particle after the k-th iteration.

B. Improved Dynamic Multi-Population Particle Swarm Optimization Algorithm

To address the shortcomings of Particle Swarm Optimization such as difficulty in escaping local optima, low execution efficiency, and the imbalance between global and local search capabilities, this paper adopts an Improved Dynamic Multi-Population Particle Swarm Optimization Algorithm (IDM-PSO) [18]. Through the dynamic multi-population strategy, the PSO algorithm is improved by dividing the population into inferior, superior, and mixed groups based on their fitness values. The inferior group consists of particles with fitness values higher than the average, while the superior group consists of particles with fitness values lower than the average. An equal number of populations are selected from both to form a mixed group, ensuring balanced population sizes for all three groups.

1) The updating mechanism for the superior group: The main reason for the slow convergence speed of Particle Swarm Optimization in the later stages of optimization is its difficulty in escaping from the current local extremum, resulting in a decrease in accuracy. To enhance the ability of the superior group to escape from local optima, Levy flight is introduced. Levy flight has the characteristics of short-distance tracking and long-distance jumping. This type of flight enhances particle activity and jumping ability, expands the particle search range, helps to enhance particle diversity, avoids the algorithm falling into local optima, and can improve the convergence accuracy and speed of the algorithm. The formula for Levy flight is as follows:

$$Levy(x) = \frac{0.01\mu}{|\nu|^{\frac{1}{\lambda}}} \quad (9)$$

In the equation: λ∈[1, 3], In this paper, we set λ=1.5; μ and ν follow a normal distribution. The formula is as follows:

$$\begin{cases} \mu \sim N(0, \sigma_\mu^2) \\ \nu \sim N(0, \sigma_\nu^2) \end{cases} \quad (10)$$

$$\begin{cases} \sigma_\mu^2 = \left(\dfrac{\Gamma(1+\lambda) \cdot \sin\left(\dfrac{\pi\lambda}{2}\right)}{\lambda \cdot \Gamma\left(\dfrac{1+\lambda}{2}\right) \cdot 2^{\frac{\lambda-1}{2}}} \right)^{\frac{1}{\lambda}} \\ \sigma_\nu^2 = 1 \end{cases} \quad (11)$$

In Eq. (11), Γ is the standard gamma function, expressed as follows:

$$\Gamma = \int_0^{+\infty} e^{-t} \cdot t^{x-1} dt \quad (12)$$

After using Levy flight, the particle velocity update formula as in Eq. (7), combined with the position update formula according to Eq. (8), yields:

$$x^l = x_{id}^{k+1} + \partial \cdot Levy(D) \quad (13)$$

In the equation: α is the step size control factor, where α=0.01; x^l represents the updated position after using the Levy flight strategy; D is the dimensionality of the particle, where D=3.

Although Levy flight can help particles escape local optima, it does not guarantee that the updated particle position is better than the original position. Therefore, to avoid meaningless position updates, this paper introduces the evaluation strategy of greedy algorithm to decide whether to update the optimal particle position. That is, the position update is only performed if the updated position is better than the original position; otherwise, the original position is retained. The process is shown in Eq. (14):

$$x^{new} = \begin{cases} x^l, f\left(x^l\right) < f\left(x_{id}^{k+1}\right) \\ x_{id}^{k+1}, f\left(x^l\right) > f\left(x_{id}^{k+1}\right) \end{cases} \quad (14)$$

In Eq. (14): x^{new} represents the particle position updated using the greedy strategy; f(x^l) is the fitness value of the particle position updated using Levy flight, and f$\left(x_{id}^{k+1}\right)$ is the fitness value of the particle after the *k*-th iteration.

2) The updating mechanism for the inferior group: The particles in the inferior group have limited valuable information and are far from the optimal solution of the problem. They can be optimized by means of Gaussian mutation [19] to search for the optimal solution across the entire space. Mutation can explore various possible solution regions in the entire search space and also prevent premature convergence and increase the diversity of subpopulations. The mutation criteria are as follows:

$$r > \frac{1}{2}\left(1 + \arctan\left(\frac{k}{k_{\max}}\right) \cdot \frac{4}{\pi}\right) \quad (15)$$

$$x^g = x_{id}^{k+1} \cdot (1 + N(0,1)) \quad (16)$$

where, k represents the current iteration number; k_{max} is the maximum iteration number; r and N(0, 1) are random numbers between [0,1]; and x^g denotes the updated position after mutation. When Eq. (15) holds true, Eq. (16) is executed to perform Gaussian mutation on the particles. The probability of Eq. (15) holding true gradually decreases after multiple iterations, and consequently, the probability of particle mutation decreases as well. The mutation operation can cause particles to mutate with a relatively high probability in the initial stages, thereby expanding the search range of particles in the solution space and ensuring particle diversity.

In addition, the concept of mixed particles is introduced to modify the traditional velocity update formula. The mixed particle, denoted as P_{mix}, is composed of dimensions randomly selected from each particle's current historical best position, with no repetition of dimensions from the same particle. The generation of mixed particles [20] is illustrated in Fig. 1.

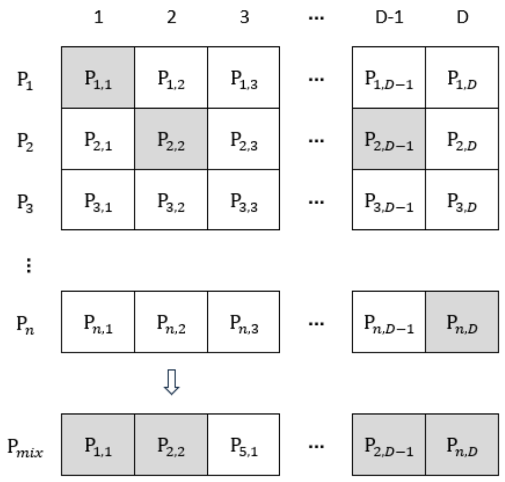

Fig. 1. The process of generating mixed particles.

The velocity update method for the inferior group obtained from this is as follows:

$$v_{id}^{k+1} = \omega \cdot v_{id}^{k+1} + c_1 r_1 \left(P_{id} - x_{id}^k\right) + c_2 r_2 \left(P_g - x_{id}^k\right) + c_3 r_3 \left(P_{mix} - x_{id}^k\right) \quad (17)$$

In the equation: c_3 is the mixed learning factor, $c_3 = 1.5$; r_3 is a random number between 0 and 1. The mixed particle serves as a traction factor guiding the velocity update of the particles. It effectively addresses the issue of falling into local optima. Additionally, its own excellence also encourages the particles to evolve towards more optimal directions.

3) The mixed group updating mechanism: The mixed group lies between the superior group and the inferior group. Due to significant differences among individuals in the early stages of the algorithm, it focuses more on their cognitive aspects. Therefore, improved learning factors and inertia weights are introduced [21].

$$c_1 = 2\cos\left(\frac{\pi k}{2k_{\max}}\right), c_2 = 2\sin\left(\frac{\pi k}{2k_{\max}}\right) \tag{18}$$

$$\omega = \omega_{\min} + \left(\omega_{\max} - \omega_{\min}\right)\cdot\cos\left(\frac{\pi k}{k_{\max}}\right) \tag{19}$$

From Eq. (18) and Eq. (19), it can be observed that the velocity and position update formulas for the mixed group are:

$$v_{id}^{k+1} = \omega\cdot v_{id}^{k+1} + c_1 r_1\left(P_{id} - x_{id}^{k}\right) + c_2 r_2\left(P_g - x_{id}^{k}\right) \tag{20}$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \tag{21}$$

C. *The process of the Improved Dynamic Multi-Population Particle Swarm Optimization Algorithm*

Based on the aforementioned method for improving the particle swarm algorithm, the steps of the Improved Dynamic Multi-Population Particle Swarm Optimization Algorithm in optimizing the time-optimal trajectory planning of robotic arms are as follows:

Step 1: Initialize the parameters of the Improved Dynamic Multi-Population Particle Swarm Optimization Algorithm, including the population size N, maximum iteration count k_{max}, population dimension D, upper bound ub and lower bound lb of the search space, and initialize the population positions.

Step 2: Calculate the fitness values of the population according to Eq. (5) and divide them into three subpopulations based on their fitness values.

Step 3: Calculate the fitness values of the three subpopulations using Eq. (5). Utilize the runtime of the three trajectory segments from each subpopulation into Eq. (1) and (2). Assess whether the constraints are satisfied using Eq. (6). If satisfied, update the fitness values of the three subpopulations using the respective methods; otherwise, assign a large value to eliminate them in the next iteration.

Step 4: Merge the subpopulations. Output the optimal solution after the iteration ends.

Based on the above steps, the flowchart of the Improved Dynamic Multi-Population Particle Swarm Optimization Algorithm is shown in Fig. 2.

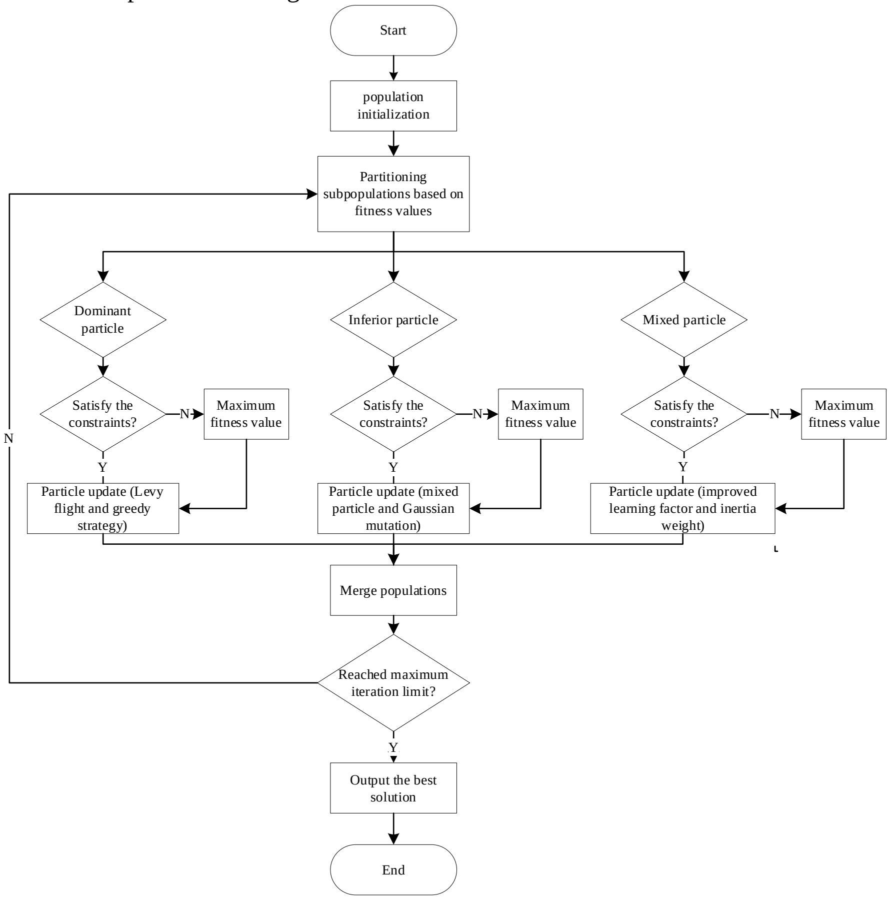
Start
population initialization
Partitioning subpopulations based on fitness values
Dominant particle
Inferior particle
Mixed particle
Satisfy the constraints?
N
Maximum fitness value
Y
Particle update (Levy flight and greedy strategy)
Particle update (mixed particle and Gaussian mutation)
Particle update (improved learning factor and inertia weight)
Merge populations
Reached maximum iteration limit?
Output the best solution
End

Fig. 2. Improved dynamic multi-population particle swarm optimization algorithm flowchart.

IV. Experimental Simulation and Analysis

A. Experimental Design

The experiment employs the PUMA560 robotic arm model and conducts simulations for time-optimal trajectory planning of the robotic arm using MATLAB. The D-H parameters of the PUMA560 robotic arm are listed in Table I. The robotic arm model constructed based on the D-H parameters table is illustrated in Fig. 3.

TABLE I. D-H Modeling Parameters of the PUMA560 Robotic Arm

Link	$\theta_i/(°)$	$\partial_{i-1}/(°)$	a_{i-1}/mm	d_i/mm
1	θ_1	0	0.00	0.00
2	θ_2	-90	0.00	149.09
3	θ_3	0	431.80	0.00
4	θ_4	-90	20.32	433.07
5	θ_5	90	0.00	0.00
6	θ_6	-90	0.00	0.00

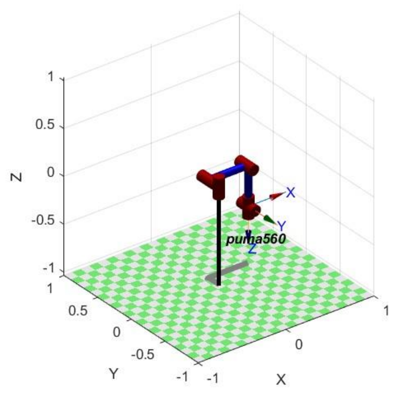

Fig. 3. Model of the PUMA560 robotic Arm.

Interpolation using a 3-5-3 polynomial requires specified preset times to calculate the polynomial coefficients. In this study, the preset interpolation time for each segment is set to four seconds, totaling 12 seconds for all three segments. The path points for each joint of the robotic arm are provided in Table II.

TABLE II. Joint Space Path Points of the Robotic Arm

Joint	X0(rad)	X1(rad)	X2(rad)	X3(rad)
1	0.1024	0.4164	0.1374	-0.2236
2	-0.3157	0.2236	0.9763	0.3492
3	0.2384	-0.1752	0.7600	0.3893
4	0.1232	0.4535	-0.2478	0.4457
5	0.2453	-0.2223	0.3479	-0.0045
6	0.3253	-0.0886	0.2976	-0.1672

B. Experimental Simulation

In order to validate the correctness and effectiveness of the Improved Dynamic Multi-Population Particle Swarm Optimization Algorithm (IDM-PSO), comparative experiments were conducted with the Basic Particle Swarm Optimization Algorithm (PSO) and the Artificial Fish Swarm Algorithm (AFSA). During the iterative optimization process, for PSO, the number of particles N = 30, the learning factors $c_1 = c_2 = 1.5$, the inertia weight $\omega = 0.9$, and the maximum number of iterations $k_{max} = 90$; for IDM-PSO, the number of particles N = 30, in the inferior and superior groups $c_1 = c_2 = 1.5$, in the mixed group, c_1 and c_2 vary according to Eq. (18), the inertia weights $\omega_{max} = 0.9$ and $\omega_{min} = 0.4$, and ω varies according to Eq. (19), with the maximum number of iterations $k_{max} = 90$; for AFSA, the number of particles N = 30, and the maximum number of iterations $k_{max} = 90$. To ensure the stability of the robotic arm's actual operation and the accuracy of trajectory planning, the maximum angular velocity for each joint was set to 3.5 rad/s, and the maximum acceleration was set to 6.5 rad/s^2. The experiment will optimize the trajectory planning time of the six joints of the robotic arm using these three different intelligent algorithms. The comparison of adaptation curves for each joint is depicted in the simulated results as shown in Fig. 4 to Fig. 9.

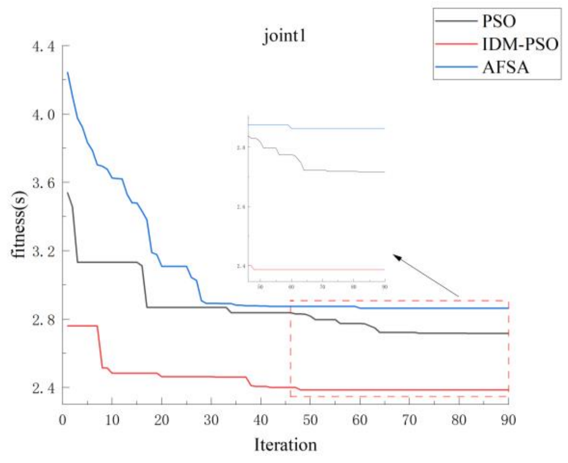

Fig. 4. Comparison chart of convergence for joint 1.

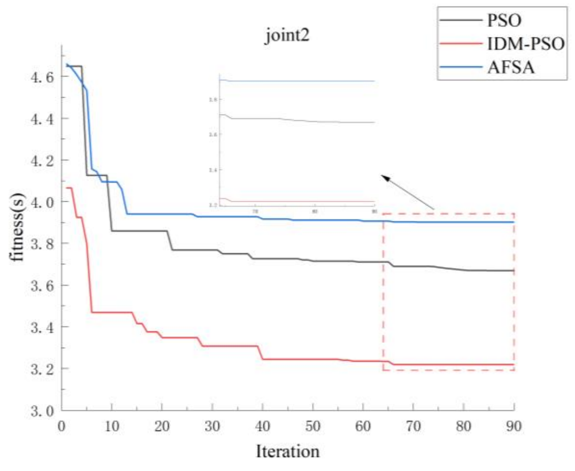

Fig. 5. Comparison chart of convergence for joint 2.

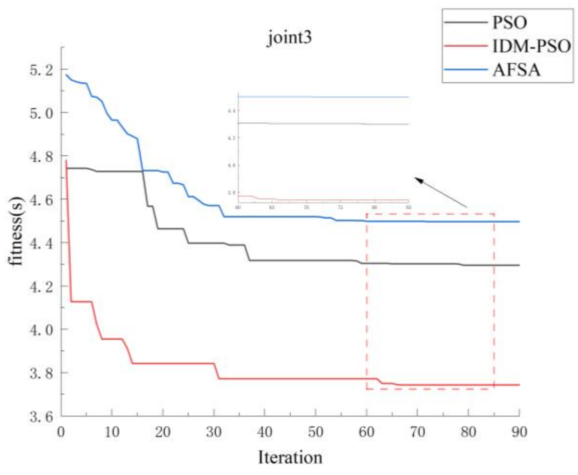

Fig. 6. Comparison chart of convergence for joint 3.

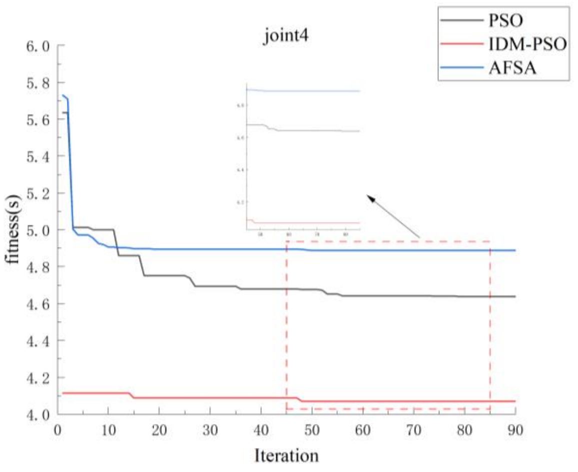

Fig. 7. Comparison chart of convergence for joint 4.

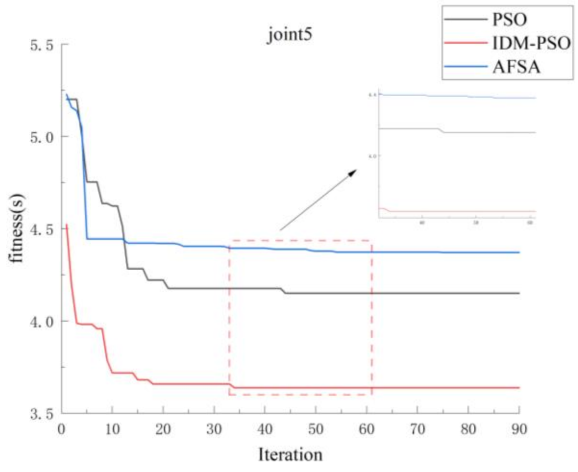

Fig. 8. Comparison chart of convergence for joint 5.

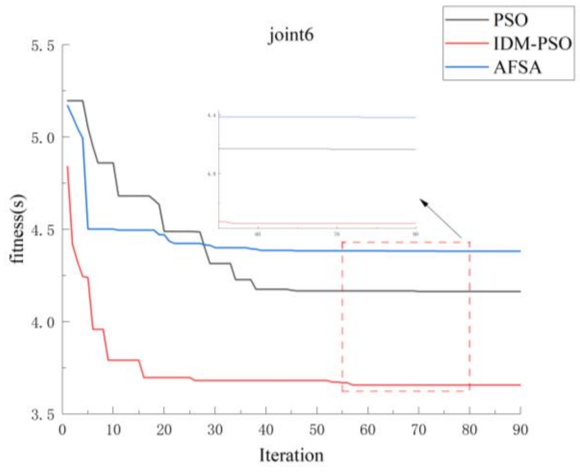

Fig. 9. Comparison chart of convergence for joint 6.

From Fig. (4) to Fig. (9), it can be observed that the Improved Dynamic Multi-Population Particle Swarm Optimization Algorithm exhibits higher convergence accuracy compared to the Basic Particle Swarm Optimization Algorithm and significantly improved efficiency compared to the Artificial Fish Swarm Algorithm. While retaining the advantages of the Basic Particle Swarm Optimization Algorithm, the Improved Dynamic Multi-Population Particle Swarm Optimization Algorithm is more capable of escaping local optima and achieves faster optimization efficiency. The time taken for each joint segment in the 3-5-3 polynomial trajectory planning under the optimization of the three algorithms is shown in Tables III, IV, and V.

TABLE III. MOTION TIME FOR TRAJECTORY PLANNING OF EACH JOINT (PSO)

Joint	$t(s)$	$t_1(s)$	$t_2(s)$	$t_3(s)$
1	2.717	0.895	1.031	0.791
2	3.670	0.889	1.466	1.315
3	4.296	1.021	2.253	1.022
4	4.639	0.851	2.297	1.491
5	4.150	1.183	2.054	0.913
6	4.163	1.041	1.953	1.169

TABLE IV. MOTION TIME FOR TRAJECTORY PLANNING OF EACH JOINT (AFSA)

Joint	$t(s)$	$t_1(s)$	$t_2(s)$	$t_3(s)$
1	2.864	0.939	1.098	0.827
2	3.902	1.000	1.470	1.432
3	4.497	1.143	2.318	1.036
4	4.888	0.907	2.370	1.611
5	4.372	1.258	2.141	0.973
6	4.382	1.109	2.040	1.233

TABLE V. MOTION TIME FOR TRAJECTORY PLANNING OF EACH JOINT (IDM-PSO)

Joint	$t(s)$	$t_1(s)$	$t_2(s)$	$t_3(s)$
1	2.386	0.814	0.861	0.711
2	3.221	0.755	1.296	1.170
3	3.743	0.922	1.985	0.836
4	4.071	0.758	2.038	1.275
5	3.639	1.026	1.816	0.797
6	3.657	0.972	1.606	1.079

In Tables III, IV, and V, the time taken for each joint for each trajectory segment under optimization by the three intelligent algorithms is statistically recorded. In which, "t" represents the total time used for trajectory planning in three segments after polynomial trajectory interpolation for each joint optimized using intelligent algorithms. "t_1, t_2, t_3" represent the time used for trajectory planning in three segments for each joint. To ensure that all joints can complete the motion task while satisfying velocity and acceleration constraints, the maximum time for each segment of the trajectory for all six joints needs to be selected. Therefore, for the Basic Particle Swarm Optimization Algorithm, the time for the three segments of the trajectory is as follows: t_1=1.183s, t_2= 2.297s, t_3=1.491s, with a total time t=4.971s. For the Artificial Fish Swarm Algorithm, the time for the three segments of the trajectory is t_1=1.258s, t_2=2.370s, t_3= 1.611s, with a total time t=5.239s. For the Improved Dynamic Multi-Population Particle Swarm Optimization Algorithm, the time for the three segments of the trajectory is t_1=1.026s, t_2=2.038s, t_3=1.275s, with a total time t=4.339s. The comparison shows that the Improved Dynamic Multi-Population Particle Swarm Optimization Algorithm reduces the trajectory planning time by approximately 12.7% compared to the Basic Particle Swarm Optimization Algorithm and by approximately 17% compared to the Artificial Fish Swarm Algorithm, leading to improved efficiency. Three algorithms (IDM-PSO, PSO, and AFSA) were subjected to six repeated experiments, and the experimental results are shown in Table VI.

The experimental data from Table VI indicates that over six experiments, there is no significant difference in the time taken to complete the trajectory planning of the robotic arm among the three algorithms, validating the accuracy of the algorithms.

TABLE VI. THE EXPERIMENTAL RESULTS OF DIFFERENT OPTIMIZATION ALGORITHMS (UNIT: S)

Number of Experiments	IDM-PSO	PSO	AFSA
1	4.339	4.971	5.239
2	4.379	5.019	5.240
3	4.354	5.007	5.249
4	4.357	5.281	5.242
5	4.289	4.915	5.247
6	4.300	5.087	5.226
Average value	4.336	5.047	5.241

The motion characteristics, including displacement, velocity, and acceleration of each joint optimized by the Improved Dynamic Multi-Population Particle Swarm Optimization Algorithm during trajectory planning, as well as the end-effector trajectory of the robotic arm, are illustrated in the following figures.

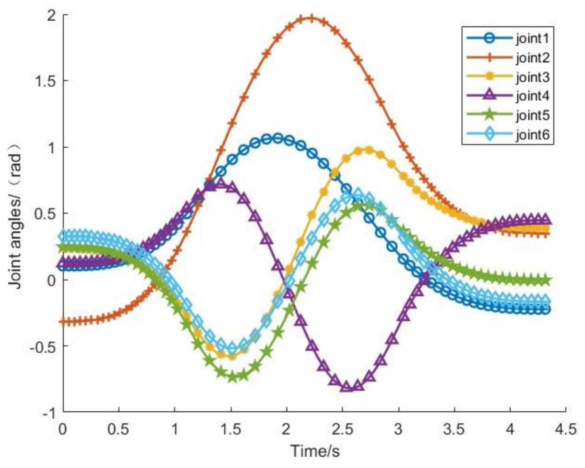

Fig. 10. Displacement curves of each joint.

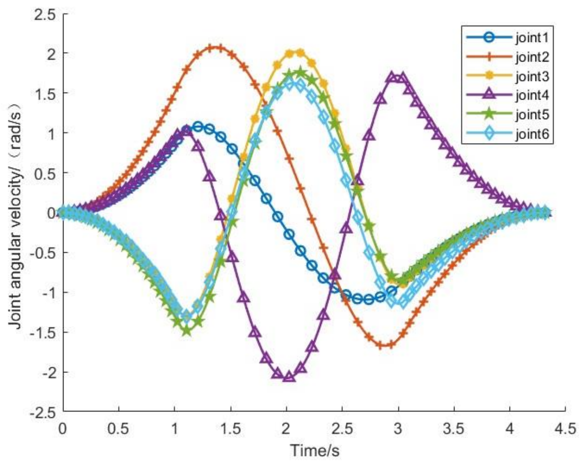

Fig. 11. Velocity curves for each joint.

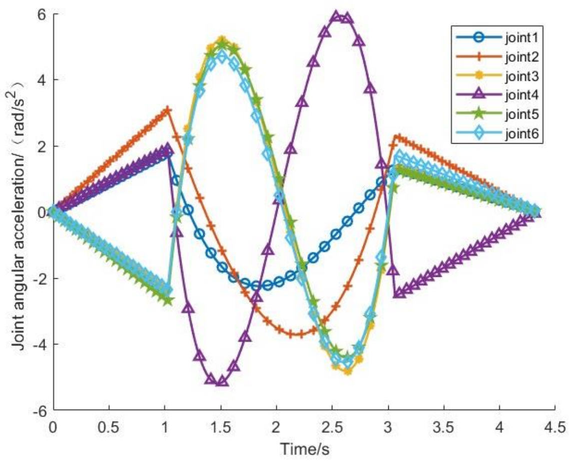

Fig. 12. Acceleration curves for each joint.

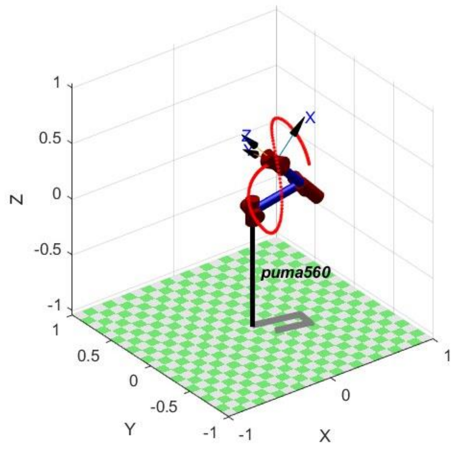

Fig. 13. The end-effector trajectory of the robotic arm.

From the above Fig. 10 to Fig. 13, it can be observed that under the optimization of the improved dynamic multi-population particle swarm optimization algorithm, the displacement, velocity, and acceleration curves of each joint are continuous without abrupt changes, satisfying the constraint conditions. This validates the correctness and effectiveness of the improved dynamic multi-population particle swarm optimization algorithm.

V. Conclusion

This paper focuses on the time-optimal trajectory planning of the PUMA560 robotic arm. Under the constraints of velocity and acceleration of the robotic arm, the trajectory interpolation is conducted using the 3-5-3 polynomial interpolation function. An improved dynamic multi-population particle swarm optimization algorithm is employed for optimization, compared with the basic particle swarm optimization algorithm and the artificial fish swarm optimization algorithm. The improved algorithm achieves higher optimization accuracy and stronger capability to escape local optima. Through simulation experiments, it is observed that the trajectory curves of each joint are continuous without discontinuities. Compared with the basic particle swarm optimization algorithm and the artificial fish swarm optimization algorithm, the proposed approach reduces the time by approximately 12.7% and 17%, respectively, thus improving efficiency. The results demonstrate the correctness and effectiveness of the improved multi-population particle swarm optimization algorithm.

In this paper, time-optimal trajectory planning for the robotic arm has been conducted, with factors such as energy and impact left unconsidered. In future work, further research is required to explore objectives such as energy optimality, impact optimality, and hybrid optimality.

Acknowledgment

This work was supported by the Major Cultivation Project of Innovative Platform in Gansu Province in 2024.

References

[1] Xue-ling Yan, Bo-kai Zhu, and Chao Ma. "The Use of Industrial Robots and Employment in Manufacturing: Evidence from China." Statistical Research, vol. 37, no. 1, pp. 74-87, 2020.

[2] Yong Guo, and Lai Guang. Review of Joint Space Trajectory Planning and Optimization for Industrial Robots. Mechanical Transmission, vol. 44, no. 2, pp.154-165, 2020.

[3] Li Li, Jun-yun Shang, Yan-li Feng, and Ya-wen Huai. A Review of Joint-Type Industrial Robot Trajectory Planning. Computer Engineering and Applications, vol. 54, no. 5, pp.36-50, 2018.

[4] Zhe Zhou, and Yong Ouyang. Time-Optimal Trajectory Planning for Six-Axis Painting Robots. Combined Machine Tools and Automated Manufacturing Technology, no. 6, pp.53-57, 2023.

[5] Jia Xie, Jia-zhen Wu, Yong-guo Li, and Jin-tao Liang. Application of Improved Particle Swarm Optimization Algorithm in Trajectory Planning of Manipulator Arms. Mechanical Science and Technology, vol. 38, no. 1, pp. 368-378, 2024.

[6] Yu-xue Pu, Peng-fei Shu, Qi Jiang, and Wei-zhong Chen. Time-Energy Optimal Trajectory Planning for Industrial Robots. Computer Engineering and Applications, vol. 55, no. 22, pp. 86-90, 2019.

[7] Rong Fu, and He-hua Ju. Time-Optimal Trajectory Planning Algorithm for Manipulator Arms Based on Particle Swarm Optimization. Information and Control, vol. 40, no. 6, pp. 802-808, 2011.

[8] Wei Deng, Qi-wan Zhang, Ping Liu, and Rui Song. Optimal Time Trajectory Planning Based on Dual Population Genetic Chaotic Optimization Algorithm. Computer Integrated Manufacturing Systems, vol. 24, no. 1, pp. 101, 2018.

[9] Ji-chun Wu, Zhai-wu Zhang, Yong-da Yang, Ping Zhang, and Da-peng Fan. Time-Optimal Trajectory Planning for Manipulator Arms Based on Improved Swordfish Algorithm. Computer Integrated Manufacturing Systems, pp. 1-19, 2024.

[10] Qiang Xu, Jian-lei Xu, Yan-hai Hu, Hai-hui Chen, Xing Zhang, and Zhao-hui Xing. "Mechanical Arm Trajectory Optimization Based on Improved Simulated Annealing Genetic Algorithm." Journal of System Simulation, pp. 1-10, 2024.

[11] Guo-yu Zuo, Mi Li, and Bang-gui Zheng. "Optimal Trajectory Planning Method for Mechanical Arm Based on Improved Adaptive Multi-objective Particle Swarm Optimization Algorithm." Experimental Technology and Management, pp.1-14, 2024.

[12] Miao, X., Fu, H. and Song, X.. Research on motion trajectory planning of the robotic arm of a robot. Artificial Life and Robotics, vol. 27, no. 3, pp. 561-567, 2022.

[13] Fan, Pu, and Hai-dong Hu. "Trajectory planning of vibration suppression for hybrid structure flexible manipulator based on differential evolution particle swarm optimization algorithm." Journal of Physics: Conference Series, vol. 2691. no. 1, 2024.

[14] Yu-jia Wang, Shan-kun Nie, and Shan-li Xiao. Particle Swarm Optimization Algorithm Based on Dynamic Multi-Population. Electronic Science and Technology, vol.30, no. 7, pp. 9-12+16, 2017.

[15] Zhang Long, Xian-tao Li, Tao Shuai, Fei-juan Wen, Wen-rong Feng, and Chun-ping Liang. A Review of Research Status on Industrial Robot Trajectory Planning. Mechanical Science and Technology, vol. 40, no. 6, pp. 853-862, 2021.

[16] Shi-qi Li, Ping He, Ke Han, and Zhi-yong Zhang. A Redundant Manipulator Inverse Kinematics Solution and Optimization Method. Journal of Huazhong University of Science and Technology (Natural Science Edition), pp. 1-8, 2024.

[17] Tao Sui, Hao Jiang, Liu-jun Kong, and Qiang Jiang. Research on Manipulator Arm Trajectory Planning Based on Improved Particle Swarm Optimization Algorithm. Journal of Shenyang Ligong University, vol.42, no. 1, pp. 7-12, 2023.

[18] Yun-long Gao, and Peng Yan. Joint Optimization Algorithm Based on Multi-Population Particle Swarm Optimization and Cuckoo Search. Control and Decision, vol. 31, no. 4, pp. 601-608, 2016.

[19] Yang Yang, and Feng-yong Li. Short-Term Load Forecasting Based on Gaussian Mutation Particle Swarm Optimization. Computer Simulation, vol. 40, no. 1, pp. 125-130, 2023.

[20] Ke-xin Tang, Xiao-lei Liang, Wen-feng Zhou, Qian-hui Ma, and Yu Zhang. Dynamic Multi-Population Particle Swarm Optimization Algorithm with Recombination Learning and Hybrid Mutation. Control and Decision, vol. 36, no. 12, pp. 2871-2880, 2021.

[21] Xian-shan Shi, Hong-bin Miao, and Wei Zhang. Time-Optimal Trajectory Planning for Six-DOF Manipulator Arm Based on Improved Particle Swarm Optimization Algorithm. Machine Tool & Hydraulics, vol. 51, no. 1, pp. 20-25, 2023.