applied sciences

MDPI

Article

Time-Optimal Trajectory Planning for Woodworking Manipulators Using an Improved PSO Algorithm

Sihan Chen, Changqing Zhang * and Jiaping Yi

Research Institute of Wood Industry, Chinese Academy of Forestry, Beijing 100091, China; han_99330@163.com (S.C.)

* Correspondence: zhangchangqing69@sina.com; Tel.: + 86-10-62889346

Abstract: Woodworking manipulators are applied in wood processing to promote automatic levels in the wood industry. However, traditional trajectory planning results in low operational stability and inefficiency. Therefore, we propose a method combining 3-5-3 piecewise polynomial (composed of cubic and quintic polynomials) interpolation and an improved particle swarm optimization (PSO) algorithm to study trajectory planning and time optimization of woodworking manipulators. In trajectory planning, we conducted the kinematics analysis to determine the position information of joints at path points in joint space and used 3-5-3 piecewise polynomial interpolation to fit a point-to-point trajectory and ensure the stability. For trajectory time optimization, we propose an improved PSO that adapts multiple strategies and incorporates a golden sine optimization algorithm (Gold-SA). Therefore, the proposed improved PSO can be called GoldS-PSO. Using benchmark functions, we compared GoldS-PSO to four other types of PSO algorithms and Gold-SA to verify its effectiveness. Then, using GoldS-PSO to optimize the running time of each joint, our results showed that GoldS-PSO was superior to basic PSO and Gold-SA. The shortest running time obtained by using GoldS-PSO was 47.35% shorter than before optimization, 8.99% shorter than the basic PSO, and 6.23% shorter than the Gold-SA, which improved the running efficiency. Under optimal time for GoldS-PSO, our simulation results showed that the displacement and velocity of each joint were continuous and smooth, and the acceleration was stable without sudden changes, proving the method's feasibility and superiority. This study can serve as the basis for the motion control system of woodworking manipulators and provide reference for agricultural and forestry engineering optimization problems.

Keywords: woodworking manipulator; time-optimal trajectory planning; 3-5-3 piecewise polynomial; PSO algorithm

Citation: Chen, S.; Zhang, C.; Yi, J. Time-Optimal Trajectory Planning for Woodworking Manipulators Using an Improved PSO Algorithm. *Appl. Sci.* **2023**, *13*, 10482. https://doi.org/10.3390/app131810482

Academic Editor: Dimitris Mourtzis

Received: 25 August 2023
Revised: 12 September 2023
Accepted: 18 September 2023
Published: 20 September 2023

1. Introduction

Wood processing, as a follow-up industry to forest harvesting and transportation, is a key step in enhancing the synthetic utilization rate of wood in the forestry sector [1,2]. Compared to other industrial fields, the wood processing industry has a relatively late start-up and slow development, and with the increasing demand for wood products, advanced processing machinery, such as multi-axis CNC, industrial robots, and manipulators, are urgently needed to improve mechanization and automation levels in wood processing [3–6]. Manipulators are widely used in related industries such as mechanical manufacturing, electronic and electrical, food manufacturing, and aerospace manufacturing [7–9]. They are designed to replace workers for tasks requiring high repeatability, high execution difficulty, and low safety. With the expansion of its application scope, it will likely bring unprecedented efficiency and quality to other production and manufacturing industries. Therefore, applying manipulators to wood processing improves processing efficiency, safety, product quality, reduces labor costs and create more value. Woodworking manipulators are convenient, which can realize the free movement and rotation of four-head high-speed electric spindles in the workspace to drill and mill wooden structures. However,

their unstable operation and long running times pose challenges to the wood processing process, leading to high accident rates and poor efficiency [10,11]. Appropriate trajectory interpolation algorithms can improve manipulator's operational stability, and the use of intelligent optimization algorithms for time-optimal trajectory planning can improve the manipulator's efficiency. Thus, trajectory planning and time optimization are crucial to improving the security of operation and shortening the running time of woodworking manipulators. These improvements will enhance the machining efficiency and quality of wood processing [12].

Trajectory planning is an important prerequisite step in manipulator's control system, which includes kinematic modeling of the manipulator, forward and inverse kinematic analysis, trajectory interpolation algorithms, trajectory optimization, and simulation. It significantly impacts the energy consumption, service life, and production efficiency of the manipulator, which determines whether the manipulator's terminals can accurately and fast pass through specified path points and obtain a smooth trajectory [13,14]. According to different methods of describing motion states, there are two typical types of trajectory planning, trajectory planning in joint space and trajectory planning in the Cartesian space. When trajectory planning in joint space, the joint angle is used to represent the position and pose of the manipulator end-effector, which conveniently calculates inverse kinematics and avoids singularity problems compared to the Cartesian space [15,16]. Joint angels or positions can be obtained through inverse kinematics. In addition, introducing a trajectory planning algorithm is necessary to generate a trajectory between path points.

B-spline curves and polynomial function interpolation are commonly used in trajectory planning algorithms [17,18]. B-splines have excellent smoothing performance [19], but cannot pass through all the path points due to their inherent properties, whether it is approximate fitting or interpolation fitting. When using polynomial functions in trajectory planning, continuity of velocity and acceleration can only be ensured when the polynomials' order is at least five [18]. As the polynomials' order increases, the trajectory becomes more accurate and smoother; however, it may cause Runge's phenomenon [20]. Piecewise polynomial interpolation has proven to be a feasible method for guaranteeing trajectory smoothness while simplifying calculation [21,22]. The 3-5-3 piecewise polynomial [23] combines the advantages of cubic and quintic polynomials, enabling the stable operation of the manipulator and without the need for complex calculations in trajectory planning. It is suitable for the trajectory planning of the woodworking manipulator.

Efficiency is a primary factor in industrial production. Therefore, time optimization for trajectory planning has great practical significance, as it improves the operating efficiency of the woodworking manipulators. Recently, many intelligent optimization algorithms were proven effective at solving time-optimal problems of trajectory planning; for instance, the genetic algorithm (GA) [24], whale optimization algorithm (WOA) [25], sparrow search algorithm (SSA) [26], PSO [27], and ant colony optimization algorithm (ACO), etc. [28]. Due to its few parameter settings, easy implementation, and good applicability, PSO has been a popular research topic. However, the basic PSO has a slow convergence rate and is prone to fall into the local extremum, causing low precision. Accordingly, several studies have reformed the PSO algorithm and been successfully applied to engineering problems, among which integrating other algorithms is considered effective. Zhao et al. [29] proposed a hybrid improved whale optimization and PSO algorithm method that enhances the convergence rate, which was applied to the optimum time-jerk path planning of serial manipulators. Kamel et al. [30] applied a GA-PSO algorithm to a team of wheeled mobile robots' position control, which is superior to the GA or PSO algorithm alone. Song et al. [31] improved the PSO-ACO algorithm to achieve comprehensive and global optimization of energy dispatching; the results showed the effectiveness and higher security of this algorithm.

Although the aforementioned PSO algorithms are effective optimization solutions in their application field, they require complex parameter debugging and need many iteration numbers for convergence. Meanwhile, many previous works have only adopted

the single strategy, which is limited in improving algorithm performance. Therefore, to simplify the algorithm structure and improve algorithm performance, we propose a novel PSO algorithm that adapts multiple strategies and integrates Gold-SA [32]. It has a fast convergence rate and better optimization ability, which can be well applied in 3-5-3 piecewise polynomial trajectory planning of the woodworking manipulators to reduce operating time and improve the wood processing efficiency. The following studies were conducted: (1) we established the kinematics model and kinematics analysis of the manipulator by a modified D-H parameter method [33]; (2) based on inverse kinematics, we fit the trajectory between the path points by 3-5-3 piecewise polynomial interpolation; (3) we developed an improved PSO algorithm and optimized the running time of each joint; and (4) we conducted simulation experiments and analyses.

2. Materials and Methods

2.1. Mechanism of Woodworking Manipulator

2.1.1. Physical Model of Woodworking Manipulator

Figure 1 shows the woodworking manipulator's structure. The woodworking manipulator has a T-shaped layout consisting of four motion axes components (A, B, X, and Z), and is equipped with a four-head high-speed electric spindles at the A axis. From the perspective of an industrial robot, it consists of two rectilinear motion joints and two revolute joints. The rectilinear motion joints and the revolute joints can, respectively, achieve the translational motion of the woodworking manipulator along the X, Z axis and the rotation around the A, B axis. All joints are electrically driven by servo motors, drivers, and reducers. When processing the wooden structure, the upper computer sends instructions, and the motion controller responds to the input by controlling the joint angle or displacement, achieving control of the manipulator's velocity and position. Consequently, the end of the woodworking manipulator can run smoothly and quickly according to the predetermined trajectory, and finally reach the designated working position for processing.

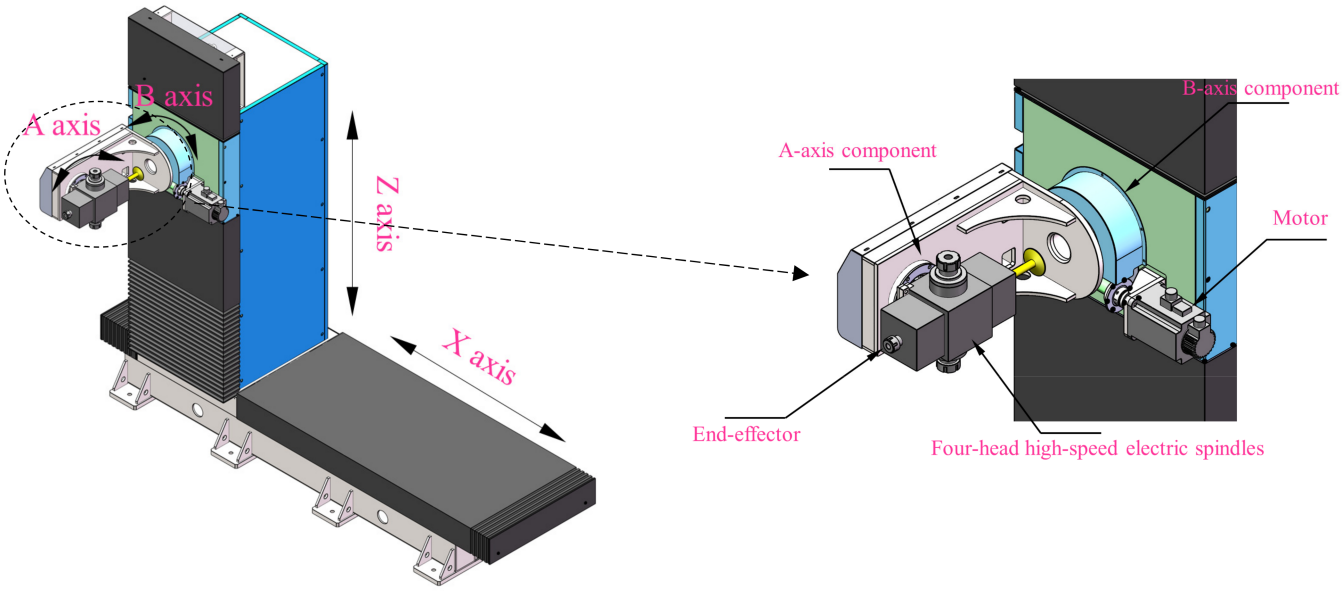

Figure 1. Woodworking manipulator.

2.1.2. Kinematic Model of Woodworking Manipulator

The relative motion components in the woodworking manipulator are treated as rigid connecting rods represented by straight lines. The motion joints are represented as moving and rotating pairs. Thus, the woodworking manipulator can be simplified. The modified D-H parameter method is commonly used for modeling different kinds of manipulators' connecting rod and joint. It is defined by four parameters: connecting rod torsion angle,

connecting rod length, connecting rod offset, and joint angle, which can achieve coordinate system transformations through homogeneous transformation matrices and describe the pose and position of the end-effector [33]. Compared with the D-H parameter method, the modeling results are the same, but the position of the coordinate system is different. In our study, the modified D-H parameter method is more convenient. Then, the simplified manipulator's connecting rod and joint model, established by the modified D-H method, is shown in Figure 2, and the parameters are shown in Table 1.

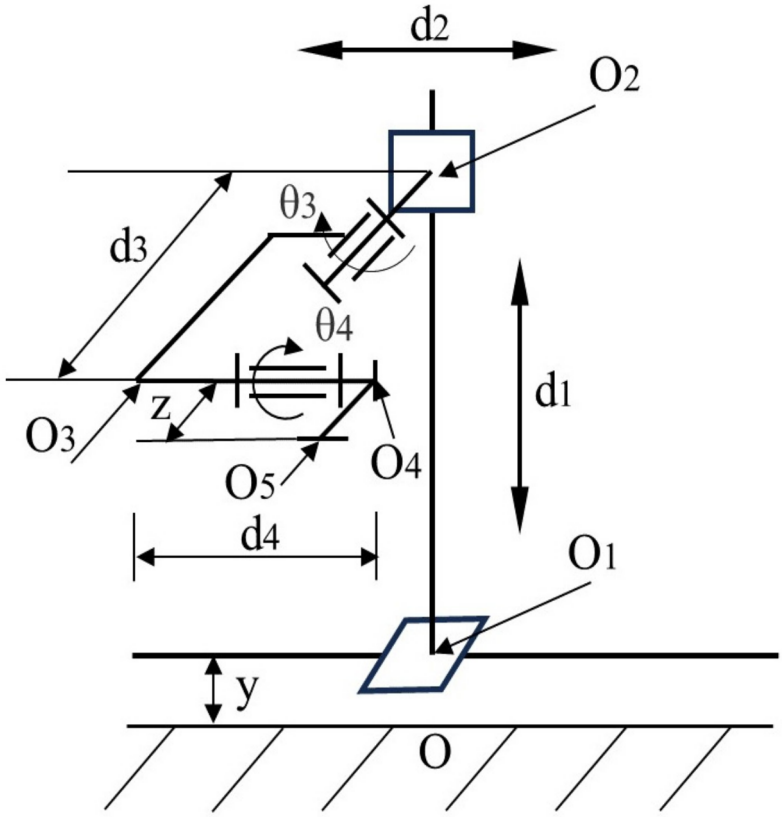

Figure 2. Simplified woodworking manipulator.

Table 1. Modified D-H parameters of woodworking manipulator.

Joint i	θ_i (rad)	d_i (mm)	α_{i-1} (rad)	a_{i-1} (mm)	Joint Range
1	$-\pi/2$	d_1	0	0	0–1000 mm
2	$\pi/2$	d_2	$-\pi/2$	0	0–1500 mm
3	θ_3	d_3	$-\pi/2$	0	−180°–180°
4	θ_4	d_4	$\pi/2$	0	−180°–180°

where θ_i is the joint angle, θ_3 and θ_4 are variable; d_i is connecting rod offset; d_1 and d_2 are variable; d_3 = 672 mm; d_4 = 286.5 mm; α_{i-1} is connecting rod torsion angle; and a_{i-1} is connecting rod length; O represents the position of the base coordinate system; O_1-O_4 represent the position of joint 1–joint 4 coordinate system; and O_5 represents the position of the tool coordinate system.

The modified D-H parameter method was used for the kinematic model, obtaining a homogeneous transformation matrix ${}^{i-1}A_i$ [33] between adjacent connecting rod coordinate systems.

$$ {}^{i-1}A_i = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & a_{i-1} \\ S\theta_i C\alpha_{i-1} & C\theta_i C\alpha_{i-1} & -S\alpha_{i-1} & -S\alpha_{i-1} d_{i-1} \\ S\theta_i S\alpha_{i-1} & C\theta_i S\alpha_{i-1} & C\alpha_{i-1} & C\alpha_{i-1} d_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (1) $$

where $C\theta$ is cosine function, and $S\theta$ is sine function.

Then, calculate the manipulator's forward kinematics T-matrix through the variable values of each joint in Table 1, which shown in the following equation:

$$\begin{aligned} T &= {}_0^0T\,{}_1^1A\,{}_2^2A\,{}_3^3A\,{}_4^4T_{tool} \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -S\theta_3C\theta_4 & S\theta_3S\theta_4 & C\theta_3 & d_2 + d_4C\theta_3 + yS\theta_3S\theta_4 \\ S\theta_3 & C\theta_4 & 0 & d_3 + yC\theta_4 \\ -C\theta_3C\theta_4 & C\theta_3S\theta_4 & -S\theta_3 & d_1 + z - d_4S\theta_3 + \mathrm{yC}\theta_3S\theta_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \end{aligned} \tag{2}$$

where T_0 is the base coordinate transformation matrix; T_{tool} is tool coordinate system transformation matrix and y, z are the parameters in them. r_{ij} represents the pose information of the end-effector, and p_x, p_y and p_z represent the position information of the end-effector.

After the kinematics equation was obtained from the forward kinematics analysis, we assumed that the value of the T-matrix was known and used the algebraic method to reverse solve the values of θ_3, θ_4, d_1, and d_2, in the Equation (3) to obtain the inverse kinematics.

$$\begin{cases} \theta_3 = \arctan2(-r_{33}, r_{13}) \\ \theta_4 = \arctan2(-r_{11}, r_{12}) \\ d_1 = p_z - z - \mathrm{yC}\theta_3S\theta_4 + d_4S\theta_3 \\ d_2 = p_x - d_4C\theta_3 - yS\theta_3S\theta_4 \end{cases}, \tag{3}$$

2.2. 3-5-3 Piecewise Polynomial Interpolation

By inverse kinematics, the angle of each joint when the manipulator's end-effector passes through the path point can be obtained. Planning the trajectory between each path point is necessary to determine the joint trajectory interpolation function and calculate the joint angle, velocity, and acceleration motion parameters between the path points. Polynomial functions are commonly used as trajectory interpolation functions. However, when only a cubic polynomial is used, the angular acceleration is discontinuous, and when only using a quintic polynomial to pass through multiple path points, it leads to complex calculations. Therefore, we used the 3-5-3 piecewise polynomial interpolation [23] to conduct trajectory planning, which reduces computational complexity, effectively ensuring the manipulator's stability while passing through multiple path points. The 3-5-3 piecewise polynomial interpolation is divided into three segments, with the first and third segments being cubic polynomial and the second segment being quintic polynomial. Taking joint 1 as an example, the joint trajectory interpolation functions of each segment are shown in the following equation:

$$\theta_1(t) = a_{10} + a_{11}t + a_{12}t^2 + a_{13}t^3, \tag{4}$$

$$\theta_2(t) = a_{20} + a_{21}t + a_{22}t^2 + a_{23}t^3 + a_{24}t^4 + a_{25}t^5, \tag{5}$$

$$\theta_3(t) = a_{30} + a_{31}t + a_{32}t^2 + a_{33}t^3, \tag{6}$$

where a_{ij} is the jth coefficient of the ith function; t is interpolation time of each segment; and $\theta_i(t)$ represents the ith angular displacement at time t. Thus, the angular velocity and the angular acceleration are:

$$\dot{\theta}_1(t) = a_{11} + 2a_{12}t + 3a_{13}t^2, \tag{7}$$

$$\dot{\theta}_2(t) = a_{21} + 2a_{22}t + 3a_{23}t^2 + 4a_{24}t^3 + 5a_{25}t^4, \tag{8}$$

$$\dot{\theta}_3(t) = a_{31} + 2a_{32}t + 3a_{33}t^2, \tag{9}$$

$$\ddot{\theta}_1(t) = 2a_{12} + 6a_{13}t, \tag{10}$$

$$\ddot{\theta}_2(t) = 2a_{22} + 6a_{23}t + 12a_{24}t^2 + 20a_{25}t^3, \tag{11}$$

$$\ddot{\theta}_3(t) = 2a_{32} + 6a_{33}t, \tag{12}$$

To ensure the continuous and smooth trajectory of each segment during operation, the joint angular displacement, velocity, and acceleration must satisfy the constraints as per the following equation:

$$\begin{cases} \theta_1(0) = \theta_0 \\ \theta_2(0) = \theta_1(1) \\ \theta_3(0) = \theta_2(1) \\ \theta_3(1) = \theta_f \end{cases}, \tag{13}$$

$$\begin{cases} \dot{\theta}_1(0) = \dot{\theta}_1 = 0 \\ \dot{\theta}_2(0) = \dot{\theta}_1(1) \\ \dot{\theta}_3(0) = \dot{\theta}_2(1) \\ \dot{\theta}_3(1) = \dot{\theta}_f = 0 \end{cases}, \tag{14}$$

$$\begin{cases} \ddot{\theta}_1(0) = \ddot{\theta}_1 = 0 \\ \ddot{\theta}_2(0) = \ddot{\theta}_1(1) \\ \ddot{\theta}_3(0) = \ddot{\theta}_2(1) \\ \ddot{\theta}_3(1) = \ddot{\theta}_f = 0 \end{cases}, \tag{15}$$

where $\theta_i(0)$, $\dot{\theta}_i(0)$, and $\ddot{\theta}_i(0)$ are the joint angular displacement, velocity, and acceleration at the beginning of the ith segment, respectively; and $\theta_i(1)$, $\dot{\theta}_i(1)$ and $\ddot{\theta}_i(1)$ are the angular displacement, velocity, and acceleration at the end of the ith segment. When the manipulator starts and stops, the angular velocity and acceleration must be 0 to ensure safety. Then, the polynomial coefficient matrix a can be solved by Equations (16)–(18), where t_i of A matrix is ith segment interoperation time and X_i is the ith path point interpolation position of joint 1.

$$A = \begin{bmatrix} 1 & t_1 & t_1^2 & t_1^3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2t_1 & 3t_1^2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 6t_1 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & t_2 & t_2^2 & t_2^3 & t_2^4 & t_2^5 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2t_2 & 3t_2^2 & 4t_2^3 & 5t_2^4 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 6t_2 & 12t_2^2 & 20t_2^3 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & t_3 & t_3^2 & t_3^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2t_3 & 3t_3^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 6t_3 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \tag{16}$$

$$b = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & X_3 & 0 & 0 & X_0 & 0 & 0 & X_1 & X_2 \end{bmatrix}^{\mathrm{T}}, \tag{17}$$

$$a = A^{-1}\, b = \begin{bmatrix} a_{10} & a_{11} & a_{12} & a_{13} & a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}^{\mathrm{T}}, \tag{18}$$

Figure 3 shows the joint 1 trajectory planning results when the interpolation time of each section of 3-5-3 polynomial interpolation is 5 s. The results show that the joint displacement and velocity curves are smooth, and there are no abrupt changes in the acceleration curve. This indicates that the 3-5-3 piecewise polynomial interpolation can ensure the smooth operation of the woodworking manipulator.

According to the solving process of coefficient matrix a, 3-5-3 piecewise polynomial interpolation must predetermine the t_i of each polynomial, which greatly impacts the overall trajectory planning. Thus, selecting a suitable t_i not only ensures the stability of each joint during operation, but also improves the efficiency of the manipulator.

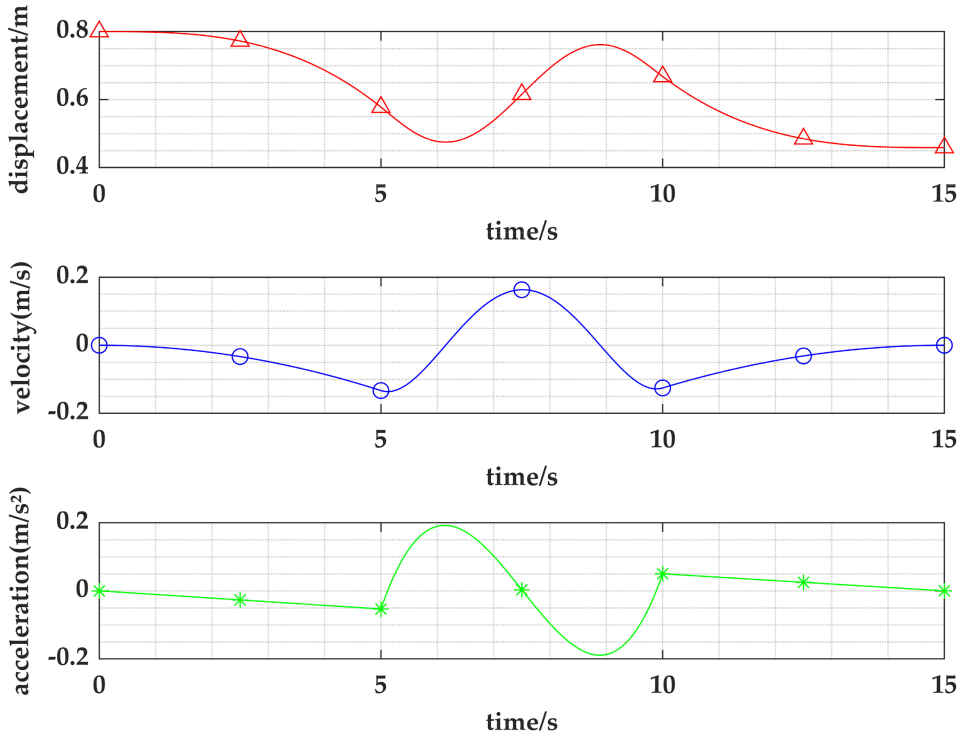

Figure 3. Joint 1 trajectory planning results by 3-5-3 polynomial interpolation.

2.3. The Traditional PSO Algorithm

PSO is an intelligent optimization algorithm originating from research into the behavior of birds foraging in forests. In PSO, bird swarms are abstracted into particle swarms with each bird representing a particle. The particles iteratively solving the extremum of the fitness function in the search space simulate the process of birds sharing information with each other to search for the largest food in the forest, as shown in Figure 4.

The process of particle iterative solution is achieved by continuously updating the particles' position and velocity information. Figure 5 shows the process of updating particle position and velocity. In the D dimension search space, we can assume that the particle number is N, $X_i = [X_{i1}, X_{i2}, \ldots, X_{iD}]$, which represents the position of the ith particle, $V_i = [V_{i1}, V_{i2}, \ldots, V_{iD}]$, which represents the velocity of the ith particle. $P_i = [P_{i1}, P_{i2}, \ldots, P_{iD}]$ represents the current best position of ith particle and can be updated by the following equation:

$$P_i(iter+1) = \begin{cases} P_i(iter), \ fitness(X_i(iter+1) > X_i(iter)) \\ X_i(iter+1), \ fitness(X_i(iter+1) < X_i(iter)) \end{cases}, \tag{19}$$

where *iter* represents iterations; *fitness* () is fitness function; $X_i(iter)$ and $P_i(iter)$ are the position and current best position of the ith particle when the iteration number is *iter*, respectively.

$P_{gbest} = [P_{gbest1}, P_{gbest2}, \ldots P_{gbestD}]$ is the current global best position. Then, the formula for updating the position and velocity of particles is as follows:

$$V_{ij}^{iter+1} = \omega V_{ij}^{iter} + c_1 r_1 \left(P_{ij} - X_{ij}^{iter} \right) + c_2 r_2 \left(P_{gbestj} - X_{ij}^{iter} \right), \tag{20}$$

$$X_{ij}^{iter+1} = X_{ij}^{iter} + V_{ij}^{iter+1}, \tag{21}$$

where ω is inertia weight; c_1 and c_2 are learning factors; r_1 and r_2 are the random numbers in the range (0,1); V_{ij}^{iter} and X_{ij}^{iter} are the velocity and position of the jth dimensional component of the ith particle when the number of iterations is *iter*, respectively. It is called basic PSO [34] when inertia weight $\omega = 1$.

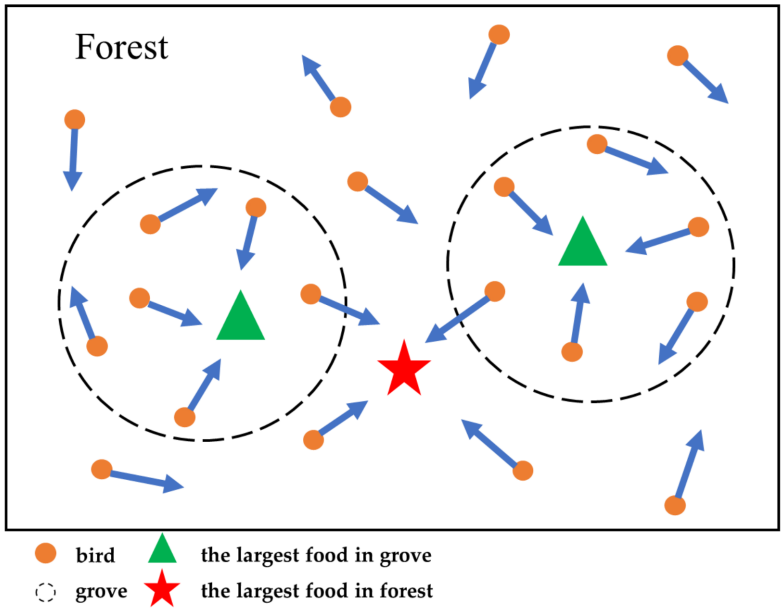

Figure 4. The process of birds foraging.

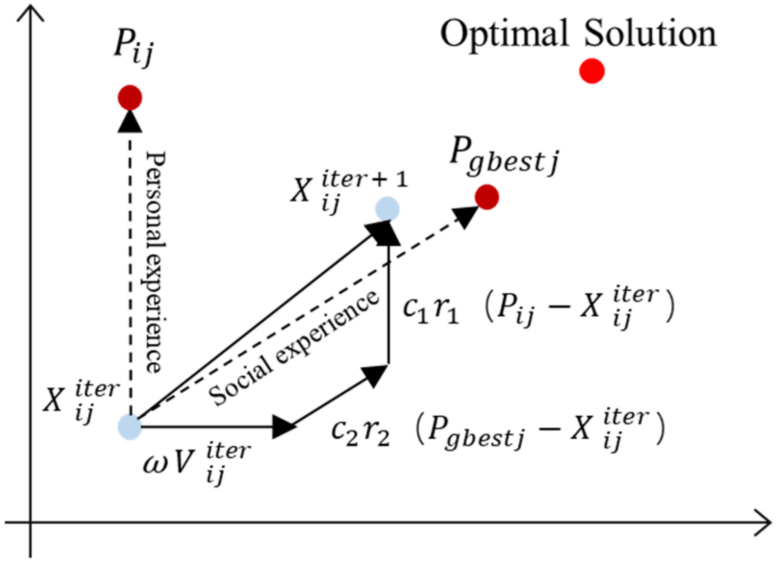

Figure 5. The updating of particle position and velocity.

2.4. The Improvement Strategy of PSO

From the structure of PSO, improvements can be made in terms of population initialization, inertia weights, and learning factors, as well as particle position and velocity update formulas. Hu et al. [35] analyzed the impact of PSO velocity on algorithm performance and confirmed that it was not an essential part and the presence of velocity may cause the iterative process in the wrong evolutionary direction. So, they proposed a simpler particle swarm optimization (SPSO), and the position update formula is shown in Equation (22):

$$X_{ij}^{iter+1} = \omega X_{ij}^{iter} + c_1 r_1 (P_{ij} - X_{ij}^{iter}) + c_2 r_2 (P_{gbestj} - X_{ij}^{iter}), \quad (22)$$

Compared to PSO, SPSO removes the velocity term, simplifies the control process of particles, and avoids the velocity parameters' impact on the algorithm's convergence rate and accuracy.

We adopted the idea of SPSO in this paper. Firstly, the initialization of the population was based on Circle chaotic mapping. Introducing tanh function to PSO, we proposed an S-curve type inertia weight nonlinear decreasing method, which increased the algorithm's ability to develop a large search space in the early iteration and precision search around the optimal solution in the later iteration. It

also integrated Gold-SA to change the particle position update method. The convergence rate and accuracy of the improved algorithm significantly increased, and overall performance was better.

2.4.1. Initial Population by Circle Chaotic Mapping

The rand function is commonly used for population initialization to generate particle position and velocity information. Although this method has high randomness, it is difficult to ensure the uniformity of distribution in the whole search space. Consequently, population diversity reduces, leading to a slow convergence rate and affecting the convergence accuracy of the algorithm. Chaotic mapping has good ergodicity and randomness [36], providing a more uniform initial population distribution than conventional random number generators by generating chaotic sequences, and increasing the population diversity. Therefore, we used Circle chaotic mapping to initialize the population, and the generated chaotic sequence was as follows [37]:

$$k(i+1) = \mathrm{mod}(k(i) + 0.2 - (0.5/2\pi)\sin(2\pi k(i)), 1), \tag{23}$$

where mod (a, b) is the remainder of a over b, and k (i) is the ith chaotic sequences number. The generation of a Circle chaotic sequence does not rely on initial values, and k (1) can be a random number between [0,1]. Figure 6 shows the distribution diagram and histogram of the 0–1 sequence values generated by the rand function and Circle chaotic map when the population size is 100. Obviously, when comparing histograms (c) and (d), there is less randomness in the Circle map, and the distribution of data points between 0.2 and 0.6 is denser. However, in comparing distributions (a) and (b), while some randomness is lost, the data points generated by the Circle map are more evenly distributed in space than the rand function. Furthermore, there are no data "clustering" or "overlapping" phenomena, which can generate high-quality initial populations in the search space, accelerate the convergence rate and enhance the ability of particles to escape from local extremum.

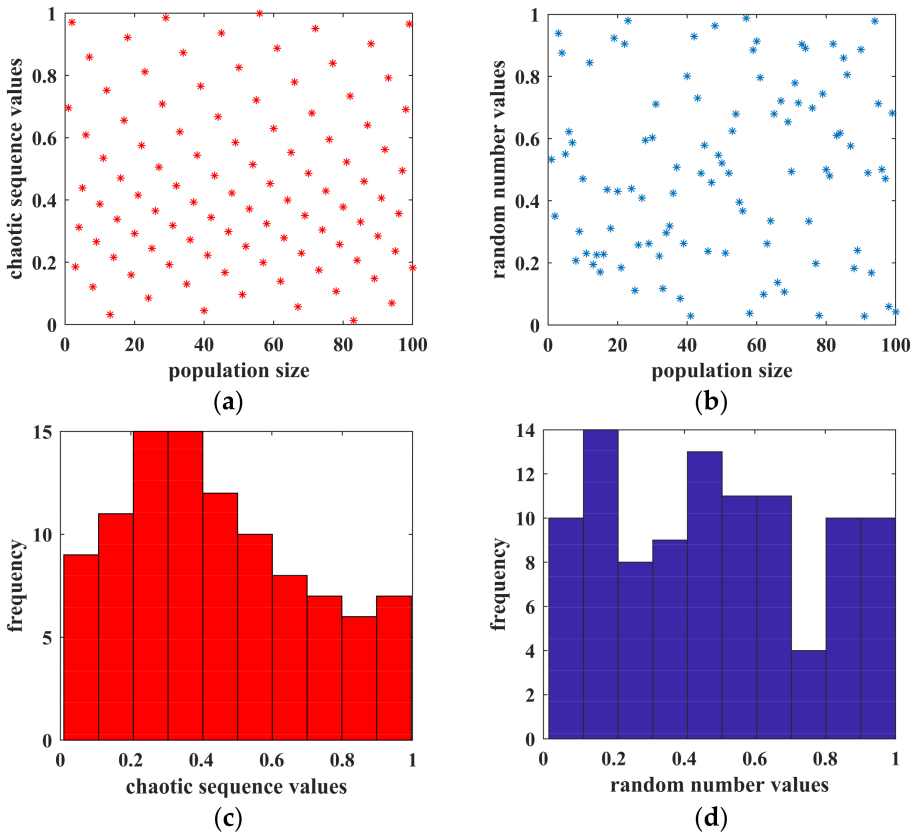

Figure 6. Comparison between two kinds of random number generators: (**a**) distribution diagram of Circle mapping chaotic sequence values; (**b**) distribution diagram of random number values of rand function; (**c**) histogram of Circle mapping chaotic sequence values; (**d**) histogram of random number values of rand function.

2.4.2. S-curve Type Inertia Weight Nonlinear Decreasing Method

ω is a vital parameter in PSO. In the early iteration, it generally has a larger value with a strong global optimization ability to accelerate the population's evolution toward the dominant population [38]. When the particles approach the optimal solution in the later iteration, it can seek local optima by using smaller values ω and performing fine searches to improve convergence accuracy. Thus, we propose an S-curve type inertia weight nonlinear decreasing method based on the tanh function [39]. The inertia weight updating formula is as follows, and Figure 7 shows the inertia weight curve:

$$\omega = \omega_{\min} + (\omega_{\max} - \omega_{\min})(1 + \tanh(2 - 4iter/iter_{\max}))/2, \quad (24)$$

where $\omega_{\min}$ is the minimum inertia weight value; $\omega_{\max}$ is the max inertia weight value; tanh() is hyperbolic tangent function; and $iter_{\max}$ is the max iterations.

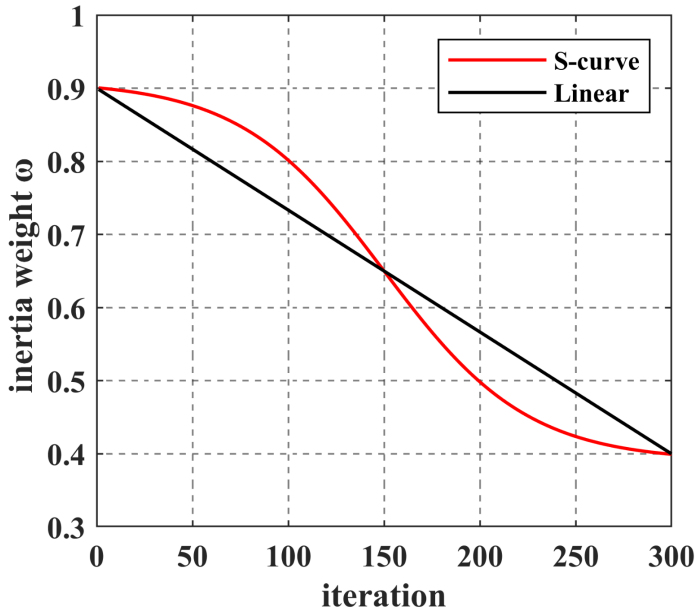

Figure 7. Inertia weight S-curve decreasing and Linear decreasing.

Compared to Linear decreasing, the S-curve decreasing in the early and late iteration of the search is slower change. Maintaining a larger ω value in the early iteration is beneficial for global optimization, whereas maintaining a smaller ω value in the later iteration is beneficial for local optimization.

2.4.3. Position Updating Integrating the Gold-SA

Gold-SA is an optimization algorithm developed by Tanyildizi et al. in 2017 based on the sine function, which has the advantages of good convergence, strong robustness, and less parameter tuning [32]. Gold-SA, which searches the entire solution space, is simulating the process of scanning a unit circle with a radius to obtain a sine function. Therefore, it has good ergodicity, can carry out extensive searches, and has a strong local optimization ability. In addition, Gold-SA continuously reduces the search space near the optimal solution through a golden section operator, improving the algorithm's convergence rate and accuracy. The position information of Gold-SA is updated using the following formula.

$$X_{ij}^{iter+1} = X_{ij}^{iter} |\sin(R_1)| - R_2 \sin(R_1) \left| \left(x_1 P_{gbestj} - x_2 X_{ij}^{iter} \right) \right|, \quad (25)$$

where R_1 is the random numbers in the range $(0, 2\pi)$, and R_2 is the random numbers in the range $(0, \pi)$. x_1 and x_2 are golden section coefficients, which are updated in the following equation:

$$x_1 = a\,(1 - \tau) + b\tau, \quad (26)$$

$$x_2 = a\tau + b\,(1 - \tau), \quad (27)$$

$$\tau = \left(\sqrt{5}-1\right)/2, \tag{28}$$

The initial values of a and b are $-\pi$ and π, respectively [32]. The golden ratio τ is approximately 0.618033. By integrating the Gold-SA position update strategy with the SPSO algorithm, a new position update formula is obtained as the following equation:

$$X_{ij}^{iter+1} = \omega X_{ij}^{iter} + c_1|\sin(R_1)|\left(P_{ij} - X_{ij}^{iter}\right) - c_2 R_2 \sin(R_1)\left|\left(x_1 P_{gbestj} - x_2 X_{ij}^{iter}\right)\right|, \tag{29}$$

Meanwhile, to further improve search efficiency, an acceleration factor λ was introduced to improve the convergence rate [40]. Finally, the GoldS-PSO position update formula is as follows:

$$X_{ij}^{iter+1} = \lambda(\omega X_{ij}^{iter} + c_1|.\sin(R_1)|(P_{ij} - X_{ij}^{iter}) - c_2 R_2 \sin(R_1)|(x_1 P_{gbestj} - x_2 X_{ij}^{iter})|), \tag{30}$$

In conclusion, the implementation steps of the GoldS-PSO are as follows:

Step 1: Set the maximum iterations $iter_{\max}$, the particle number N, upper bounds U_b and lower bounds L_b of particle position and fitness function 's dimension D;

Step 2: Population initialization based on Circle chaotic mapping;

Step 3: Calculate the fitness of each particle and record the current historical best position P_i for each particle, the current global best position P_{gbest} and optimal value of fitness G_{best};

Step 4: Update the inertia weight ω, golden section coefficients x_1 and x_2, and position X_i^{iter};

Step 5: Determine whether the updated particle position is out of bounds and use the boundary value as the particle position if it is;

Step 6: Calculate particle fitness again, update the current best historical position P_i for each particle, the current global best position P_{gbest}, and the current optimal value of fitness G_{best};

Step 7: Repeat Step 4–6 before reaching the maximum iterations and return the optimal solution X_{best} and optimal value G_{best} after reaching it.

2.5. Comparative Testing Experiment

To test the performance of the GoldS-PSO algorithm, we selected eight test functions commonly used by IEEE Congress on Evolutionary Computation (IEEE CEC) and verified its effectiveness experimentally [40]. In Table 2, F_1–F_5 are unimodal benchmark test functions with only one optimal solution, which involves testing the algorithm's convergence rate and accuracy. F_6–F_8 are multimodal benchmark testing functions with many local optima, which mainly test the algorithm's ability to jump out of the local extremum.

Table 2. Benchmark functions.

Function	Range	Dimension	Optimal Value
$F_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]$	30	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i^2\rvert + \prod_{i=1}^{n} \lvert x_i^2\rvert$	$[-10, 10]$	30	0
$F_3(x) = max_i\{\lvert x_i^2\rvert,\ 1 \le i \le n\}$	$[-100, 100]$	30	0
$F_4(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5\rvert)^2$	$[-100, 100]$	30	0
$F_5(x) = \sum_{i=1}^{n} ix_i^4 + random\ [0,\ 1)$	$[-1.28, 1.28]$	30	0
$F_6(x) = -20\ exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]$	30	0
$F_7(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]$	30	0

Table 2. *Cont.*

Function	Range	Dimension	Optimal Value
$F_8(x) = 0.1\left\{\sin^2(3\pi x_i) + \sum_{i=1}^{n}(x_i-1)^2\left[1+\sin^2(3\pi x_i+1)\right]+ (x_i-1)^2\left[1+\sin^2(2\pi x_n)\right]\right\}\sum_{i=1}^{n}u(x_i, 5, 100, 4)$ $u = (x_i, a, k, m) = \begin{cases} k(x_i-a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i-a)^m, & x_i < -a. \end{cases}$	$[-50,50]$	30	0

Figure 8 shows the F_1–F_8 diagram when the function dimension is 2. Then, we selected four related algorithms, basic PSO [34], SPSO [35], compression factor particle swarm optimization (CFPSO) [41] and Gold-SA [32] for comparative experiments. In the experiment, to ensure fairness, the basic parameters of different PSO and Gold-SA were set the same, with a population size of N = 30, a dimension of D = 30, and maximum iterations of $iter_{\max}$ = 1000. Other parameters are consistent with the original literature.

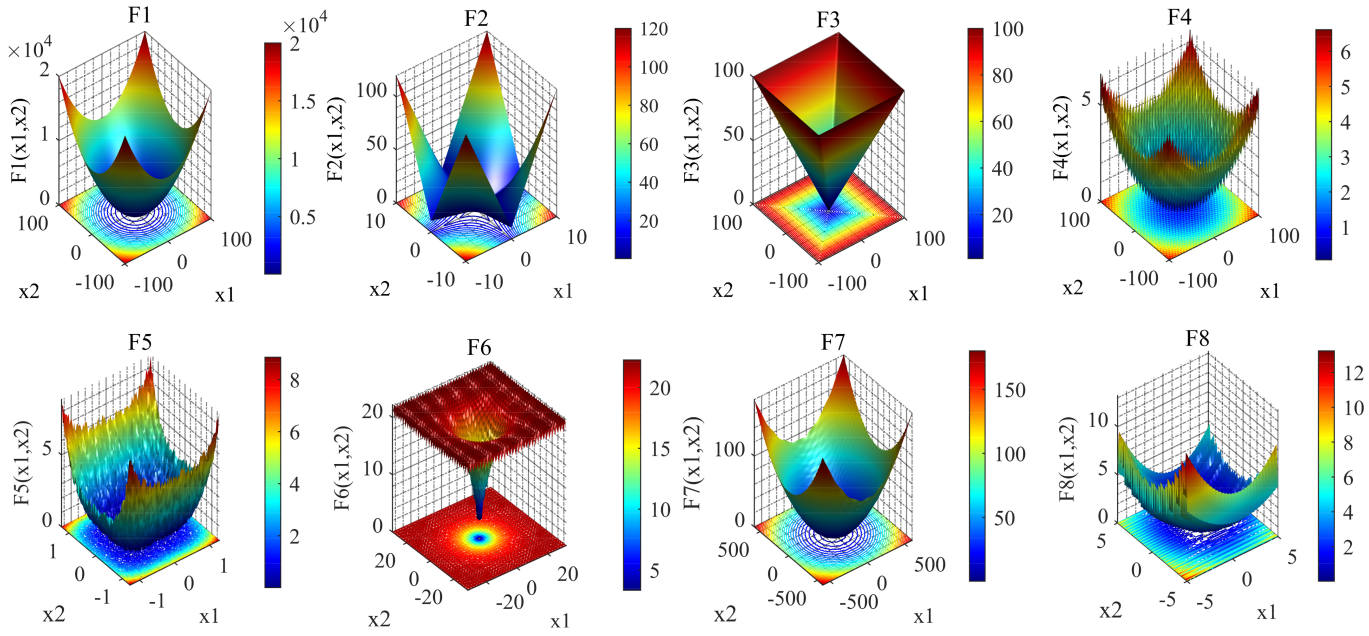

Figure 8. F_1–F_8 benchmark test function three-dimensional diagram.

Each algorithm runs 30 times independently, calculating the mean value (Mean), best value (Best), and standard deviation (Std) of the optimal results as performance indicators for algorithm optimization, reflecting the algorithm's convergence accuracy and stability. As there are no high requirements for real-time in the woodworking manipulator's trajectory planning, we did not choose to compare and analyze the running time of each algorithm as a specific performance indicator. The results of specific performance indicators in the comparative experiment are recorded in Table 3, and the best results are highlighted in bold font.

Table 3 shows that the average and standard deviation of GoldS-PSO optimization results in F_1–F_3 reached the theoretical optimal values. This finding indicates that the improved algorithm had good solving accuracy and stability in such unimodal optimization problems. Gold-SA and SPSO algorithms also achieved good results but did not reach the theoretical optimal values in the F_2 and F_3. However, there is still a significant gap in algorithm accuracy compared to GoldS-PSO. None of the algorithms reached the theoretical optimal value at F_4 and F_5, but GoldS-PSO had higher accuracy. For the test functions F_6 and F_7, GoldS-PSO and Gold-SA achieved better results with standard deviations of 0, indicating better robustness of the algorithm. In the F_8 test results, although the mean and best values of GoldS-PSO are slightly lower than Gold-SA, it had higher convergence accuracy than the PSO, CFPSO, and SPSO, demonstrating that the improved algorithm has a stronger ability to jump out of local optima. In summary, GoldS-PSO has significantly improved convergence

accuracy and stability compared to other PSO algorithms. It is only slightly inferior to Gold-SA in the terms of F_8 results. The overall performance of the algorithm has improved.

Figure 9 shows the curve of the best fitness value for each algorithm during the iterative evolution of the F_1–F_8 test function, which can more intuitively compare algorithm's convergence rate and accuracy. F1–F3 in Figure 9 shows that GoldS-PSO's convergence rate is significantly better than other algorithms. It can fully converge to the theoretical optimal value in the first 600 generations, nearly 300 generations faster than the fastest Gold-SA on average.

Although Gold-SA and GoldS-PSO both completed convergence in the first 100 generations from the F_6 and F_7 in Figure 9, GoldS-PSO still used the least iterations, showing that the efficiency and rapidity of the improved algorithm was significantly enhanced. It is shown from the F_4, F_5 and F_8 in Figure 9 that GoldS-PSO has an excellent ability to jump out of local extremum.

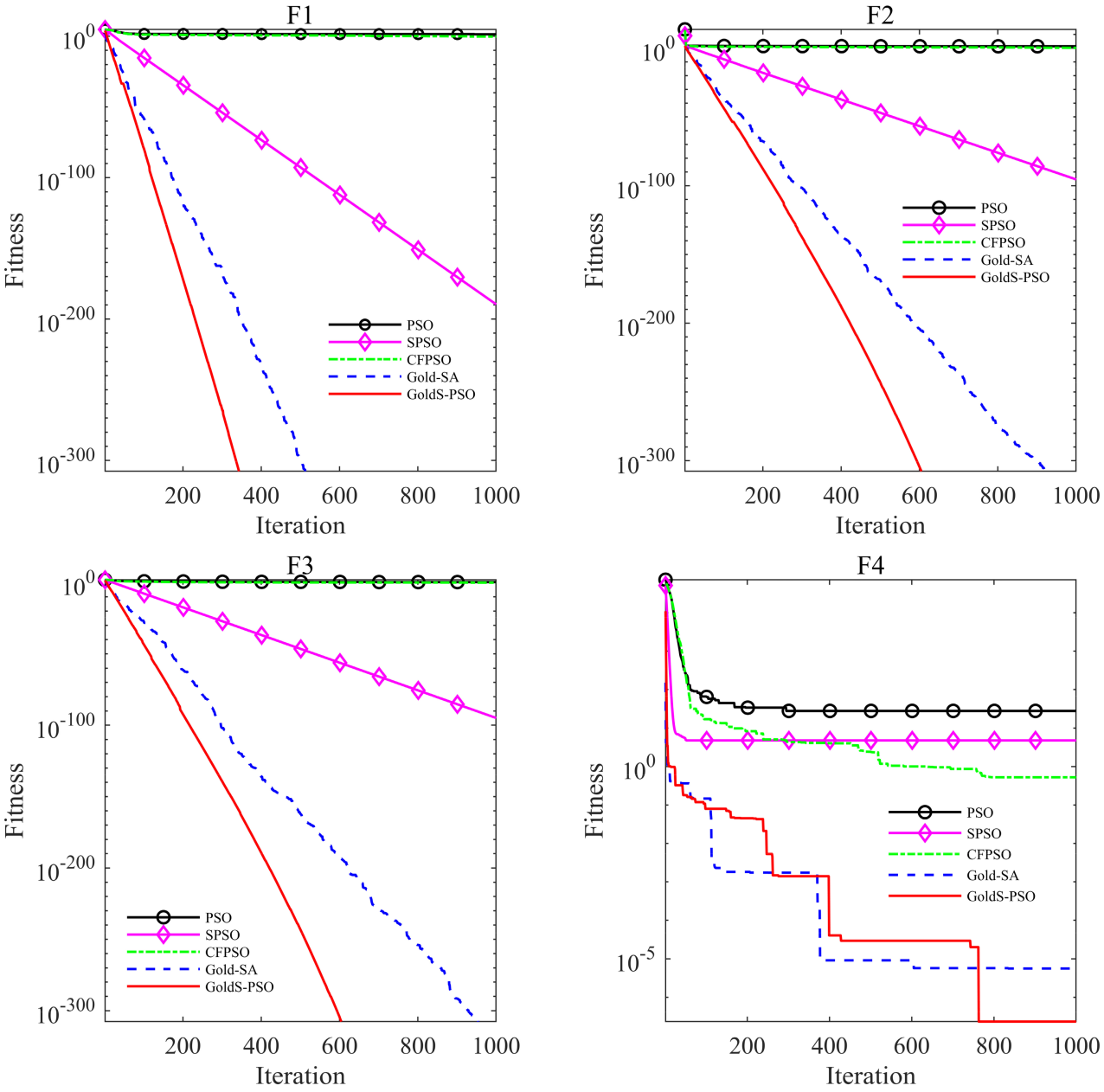

Figure 9. *Cont.*

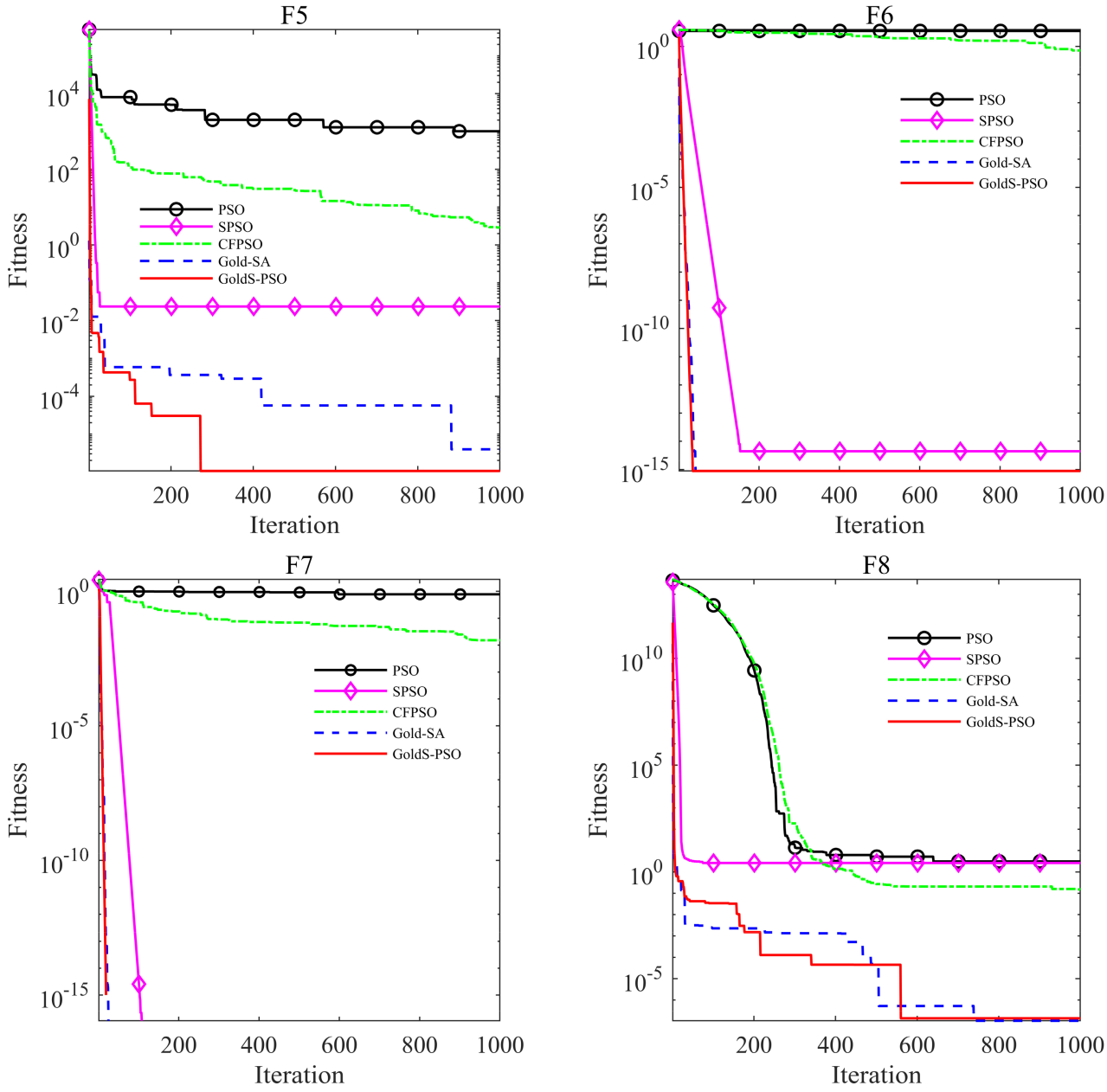

Figure 9. The best F_1–F_8 fitness evolution curves by using PSO, SPSO, CFPSO, Gold-SA, and GoldS-PSO in 30 times independent runs.

Table 3. The optimization results of the comparative testing experiment.

Function		PSO	CFPSO	SPSO	Gold-SA	GoldS-PSO
F_1	Mean	3.73248×10^{1}	0.12322×10^{1}	6.4918×10^{-190}	**0**	**0**
	Best	2.43271×10^{1}	5.3919×10^{-1}	2.5027×10^{-190}	**0**	**0**
	Std	0.46358×10^{1}	0.39174×10^{1}	1.4048×10^{-189}	**0**	**0**
F_2	Mean	2.59152×10^{1}	0.31992×10^{1}	7.6576×10^{-96}	1.3191×10^{-259}	**0**
	Best	2.31681×10^{1}	0.17982×10^{1}	3.6225×10^{-96}	**0**	**0**
	Std	0.16301×10^{1}	8.0076×10^{-1}	2.852×10^{-96}	**0**	**0**
F_3	Mean	0.23978×10^{1}	9.964×10^{-1}	1.3423×10^{-95}	2.4626×10^{-214}	**0**
	Best	0.20577×10^{1}	8.1201×10^{-1}	1.2058×10^{-95}	**0**	**0**
	Std	1.2016×10^{-1}	1.1032×10^{-1}	5.1565×10^{-97}	**0**	**0**

Table 3. *Cont.*

Function		PSO	CFPSO	SPSO	Gold-SA	GoldS-PSO
F_4	Mean	3.58749×10^{1}	0.12009×10^{1}	0.53438×10^{1}	6.3772×10^{-5}	$\mathbf{3.0695 \times 10^{-5}}$
	Best	2.7984×10^{1}	5.26×10^{-1}	0.4776×10^{1}	2.7074×10^{-7}	$\mathbf{2.3277 \times 10^{-7}}$
	Std	0.49812×10^{1}	4.6943×10^{-1}	4.2675×10^{-1}	1.6428×10^{-4}	$\mathbf{3.52212 \times 10^{-5}}$
F_5	Mean	1.5867973×10^{3}	0.75091×10^{1}	1.3191×10^{-1}	4.141×10^{-5}	$\mathbf{2.9164 \times 10^{-5}}$
	Best	1.0118771×10^{3}	0.28872×10^{1}	2.3587×10^{-2}	3.9512×10^{-6}	$\mathbf{1.0577 \times 10^{-6}}$
	Std	3.269182×10^{2}	0.38832×10^{1}	7.6126×10^{-2}	5.6872×10^{-5}	$\mathbf{2.3199 \times 10^{-5}}$
F_6	Mean	0.38518×10^{1}	0.16354×10^{1}	7.8752×10^{-15}	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$
	Best	0.35918×10^{1}	7.1535×10^{-1}	4.4409×10^{-15}	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$
	Std	1.15×10^{1}	4.6235×10^{-1}	6.4863×10^{-16}	**0**	**0**
F_7	Mean	8.6704×10^{-1}	5.0136×10^{-2}	1.1211×10^{-1}	**0**	**0**
	Best	7.7332×10^{-1}	1.5054×10^{-2}	**0**	**0**	**0**
	Std	4.6326×10^{-2}	2.0134×10^{-2}	2.6264×10^{-1}	**0**	**0**
F_8	Mean	0.56774×10^{1}	3.2971×10^{-1}	0.31016×10^{1}	$\mathbf{9.1171 \times 10^{-5}}$	9.823×10^{-5}
	Best	0.30884×10^{1}	1.4755×10^{-1}	0.26321×10^{1}	$\mathbf{1.0629 \times 10^{-7}}$	1.3947×10^{-77}
	Std	8.3954×10^{-1}	1.0153×10^{-1}	3.8235×10^{-1}	2.5837×10^{-4}	$\mathbf{1.0078 \times 10^{-4}}$

3. Experiments and Results

3.1. Time Optimal Trajectory Planning Process

In this section, GoldS-PSO was used to optimize the running time of woodworking manipulator within the constraints of each joint's maximum velocity and acceleration. As described in Section 2.2, the interpolation time t_i of each segment is an important parameter in 3-5-3 piecewise polynomial interpolation trajectory planning. Therefore, while ensuring a smooth and controllable trajectory, minimizing the t_i of each segment is necessary to improve the efficiency of manipulation. The optimization problem for the interpolation time of the ith joint is as follows:

$$f(t) = \min\ (t_{i1} + t_{i2} + t_{i3}), \tag{31}$$

$$\max\left\{\left|v_{ij}\right|\right\} \le V_{max},$$

$$\max\left\{\left|a_{ij}\right|\right\} \le A_{max}$$

where t_{ij}, v_{ij}, and a_{ij} are the interpolation time, velocity, and acceleration of the jth segment of the ith joint, respectively. V_{max} and A_{max} represent the maximum velocity and acceleration of each joint. The simultaneous movement of each joint was optimized separately to ensure that they meet the velocity and acceleration limits. The maximum value of each optimized joint in each segment was considered the optimization result. Tallying the optimization results of each segment to obtain the final optimized running time. Then, we could propose a woodworking manipulator time-optimal trajectory planning method based on 3-5-3 piecewise polynomial interpolation and the GoldS-PSO algorithm. Figure 10 shows a flowchart of the process.

3.2. Time Optimization Experiment of Woodworking Manipulator

Time Optimization experiments were conducted on the four joints of woodworking manipulator to optimize runtime. The woodworking manipulator passed through four path points A, B, C, and D. The corresponding joint position and joint angle were obtained through the inverse kinematics of Section 2.1, as shown in Table 4. The basic PSO, Gold-SA and algorithms proposed in this paper were selected for time-optimal trajectory planning. For each algorithm, the population size was set to 50, the maximum iterations were 300, the maximum velocity of joints 1 and 2 was 0.6 m/s, the maximum acceleration was 0.8 m/s^2, the maximum velocity of joints 3 and 4 was 1.8 rad/s, the maximum acceleration was 2.4 rad/s^2, and the particle position range was constrained at [1,5]. Under velocity and acceleration limitations, the total optimal interpolation time and the three-stage interpolation time for each joint were recorded in Table 5. The shortest T_{total} and maximum t_i of each joint are highlighted in bold.

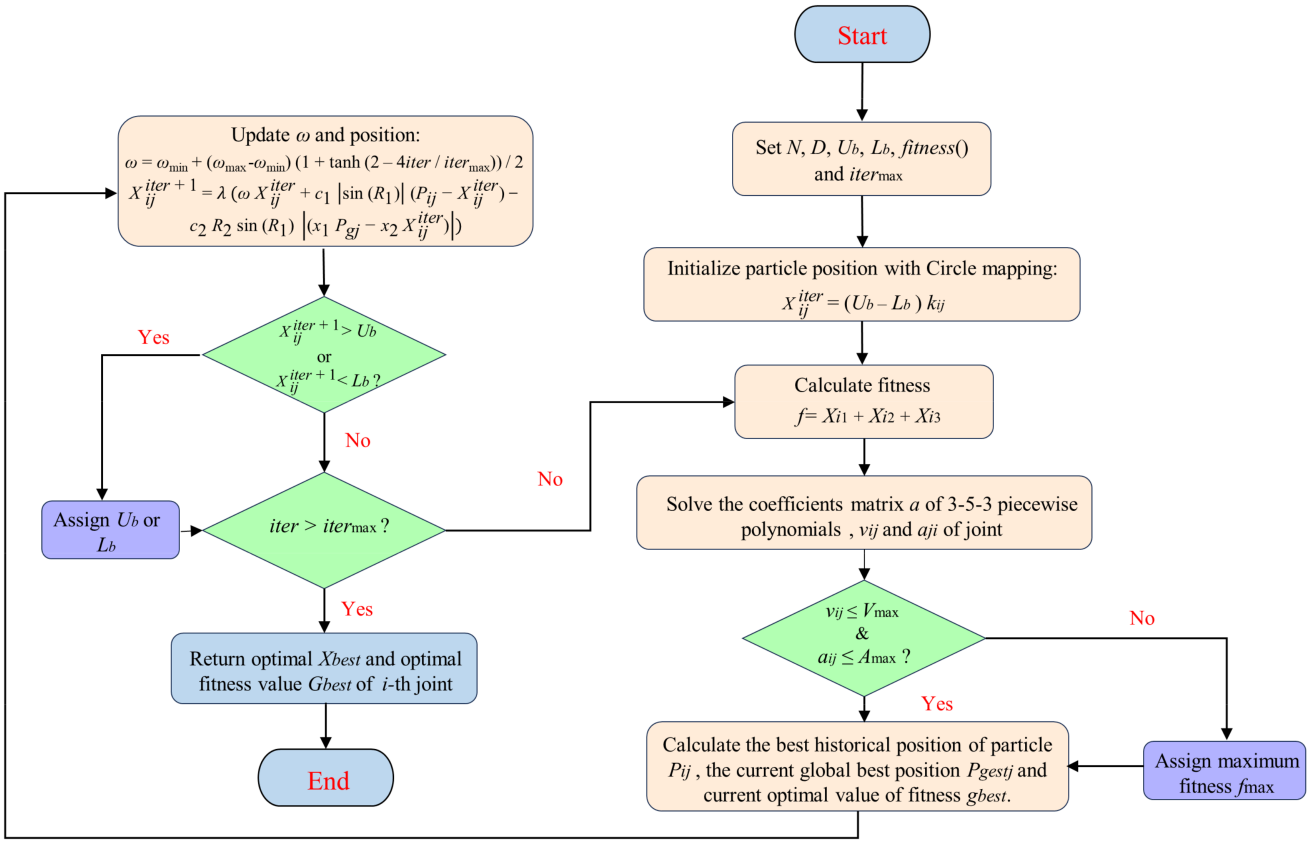

Figure 10. Time-optimal 3-5-3 piecewise polynomial trajectory planning process based on GoldS-PSO.

Table 4. Four joint positions and angle of path points.

Joint	A	B	B	D
1	0.800 m	0.578 m	0.688 m	0.459 m
2	1.200 m	0.868 m	0.567 m	0.964 m
3	3.141 rad	1.897 rad	1.468 rad	2.355 rad
4	0.000 rad	1.047 rad	1.771 rad	0.754 rad

Table 5. Interpolation time after optimizing each joint.

Algorithm	Joint	T_{total}	t_1	t_2	t_3
PSO	1	7.2210	1.7301	3.5905	1.9008
	2	7.4472	2.0576	2.2782	3.0889
	3	7.0339	2.6648	2.3479	2.0272
	4	6.9658	2.0908	2.5381	2.3368
Gold-SA	1	7.1210	1.9488	3.1077	2.0645
	2	7.5203	2.2059	2.2647	3.0497
	3	6.9526	2.4483	2.4637	2.0406
	4	7.1518	2.3862	2.4097	2.3560
GoldS-PSO	1	**7.0672**	2.1107	**2.9540**	2.0025
	2	**7.2177**	2.0009	2.6121	**2.6047**
	3	**6.8970**	**2.3189**	2.5846	1.9935
	4	**6.8713**	2.0205	2.5096	2.3412

According to Table 5, GoldS-PSO achieved the best results compared to basic PSO and Gold-SA, with the optimized total running time T_{total} of each joint being 7.0672 s, 7.2177 s, 6.8970 s, and 6.8713 s, respectively. Moreover, the maximum interpolation time of each joint segment under GoldS-PSO optimization was also the smallest compared to basic PSO and Gold-SA. The fitness iterative evolution

curve for each joint is depicted in Figure 11. It better reflects the performance of each algorithm in the optimization process.

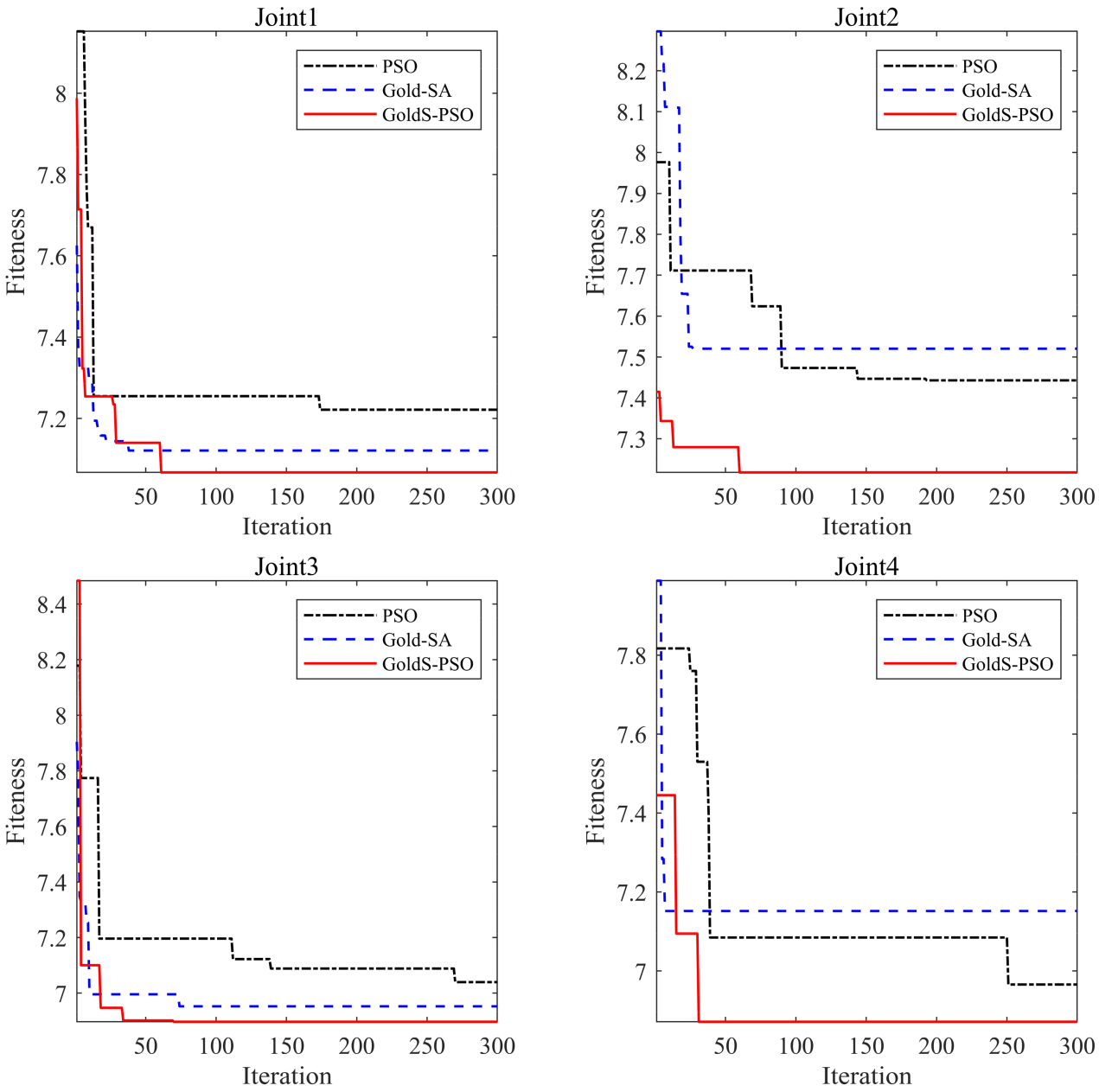

Figure 11. Four joints' iterative process by using PSO, Gold-SA, and GoldS-PSO.

Figure 11 shows that GoldS-PSO had a faster rate of convergence than PSO. Out of all the optimization processes, the iterations of GoldS-PSO were within the 60 generations to complete global convergence, whereas PSO required at least the 120 generations. Compared to Gold-SA, GoldS-PSO's rate of convergence did not significantly improve; however, its ability to jump out of local extreme values was stronger and its convergence accuracy was the highest. Although Gold-SA achieved global convergence within the 50 generations, it was hard to adjust the evolutionary direction in the middle or later iteration and easily fell into the dilemma of local extremum. Even in the optimization processes of joints 2 and 4, Gold-SA's convergence accuracy was lower than that of PSO. In a word, GoldS-PSO has a good convergence rate and is not easily trapped in local convergence. It balances the rapidity and accuracy of convergence, which indicates better performance for such engineering optimization problems.

Subsequently, we selected the maximum interpolation time of each joint segment as the final t_1, t_2, and t_3. Then, the total interpolation time obtained after optimizing the PSO, Gold-SA, and

GoldS-PSO was 9.3442 s, 8.6057 s, and 7.8776 s, respectively. The GoldS-PSO algorithm reduced time consumption by 47.483% compared to before optimization, 15.695% compared to PSO optimization, and 8.461% compared to Gold-SA optimization, demonstrating the effectiveness and superiority of the algorithm. The final planning times t_1 = 2.3189 s, t_2 = 2.9540 s, and t_3 = 2.6047 s of each segment was substituted into the 3-5-3 piecewise polynomial interpolation function in Section 2.2 to obtain the displacement, velocity, and acceleration simulation curves of each joint.

3.3. Simulation Results

According to the simulation curve in Figure 12, the displacement and velocity curves of each joint are smooth and continuous, and there is no sudden change in the acceleration, and none exceeds the limit range. In the velocity and acceleration curves, the maximum velocity and acceleration of the joint tend toward the maximum limit value, indicating improved joint operation efficiency. In conclusion, the above results proved the successful of the time optimal trajectory planning method proposed in this paper. It not only improved the operational efficiency but also effectively ensured the continuity and stability of the manipulator.

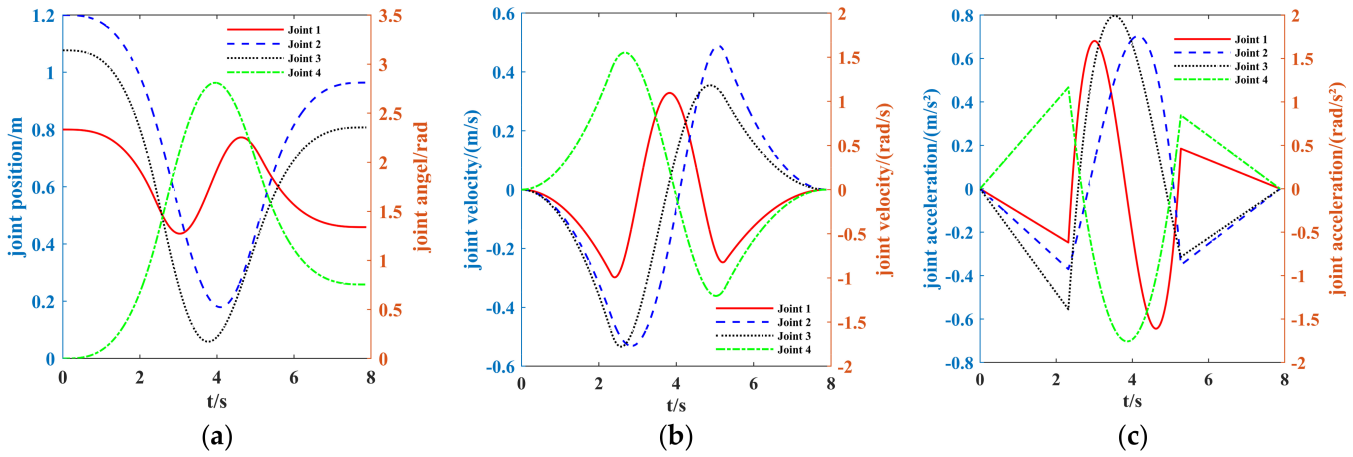

Figure 12. Woodworking manipulator displacement, velocity, and acceleration of each joint (**a**) Joint displacement running curve; (**b**) joint velocity running curve; (**c**) joint acceleration running curve.

Figure 13 depicts three trajectories from starting point A to ending point D of the woodworking manipulator end-effector under optimal time and a simulation model of the woodworking manipulator by simplifying the woodworking manipulator into a series of connecting rods in the MATLAB Robotics Toolbox [42]. As shown in the following figure, the connecting rod can stretch and rotate to simulate the movement and rotation of the four joints of the woodworking manipulator. The red curve represents the end-effector trajectory of the woodworking manipulator and the four black dots marked represent the four path points A, B, C, and D in the Cartesian space. Figure 13a–d shows the entire operation process in detail. The trajectory of the end-effector passes through all paths point stably without interruption. Our results proved that the proposed algorithm achieved the time-optimal trajectory planning of the woodworking manipulator while ensuring stability and operational accuracy. In other words, we demonstrated that the proposed method is feasible.

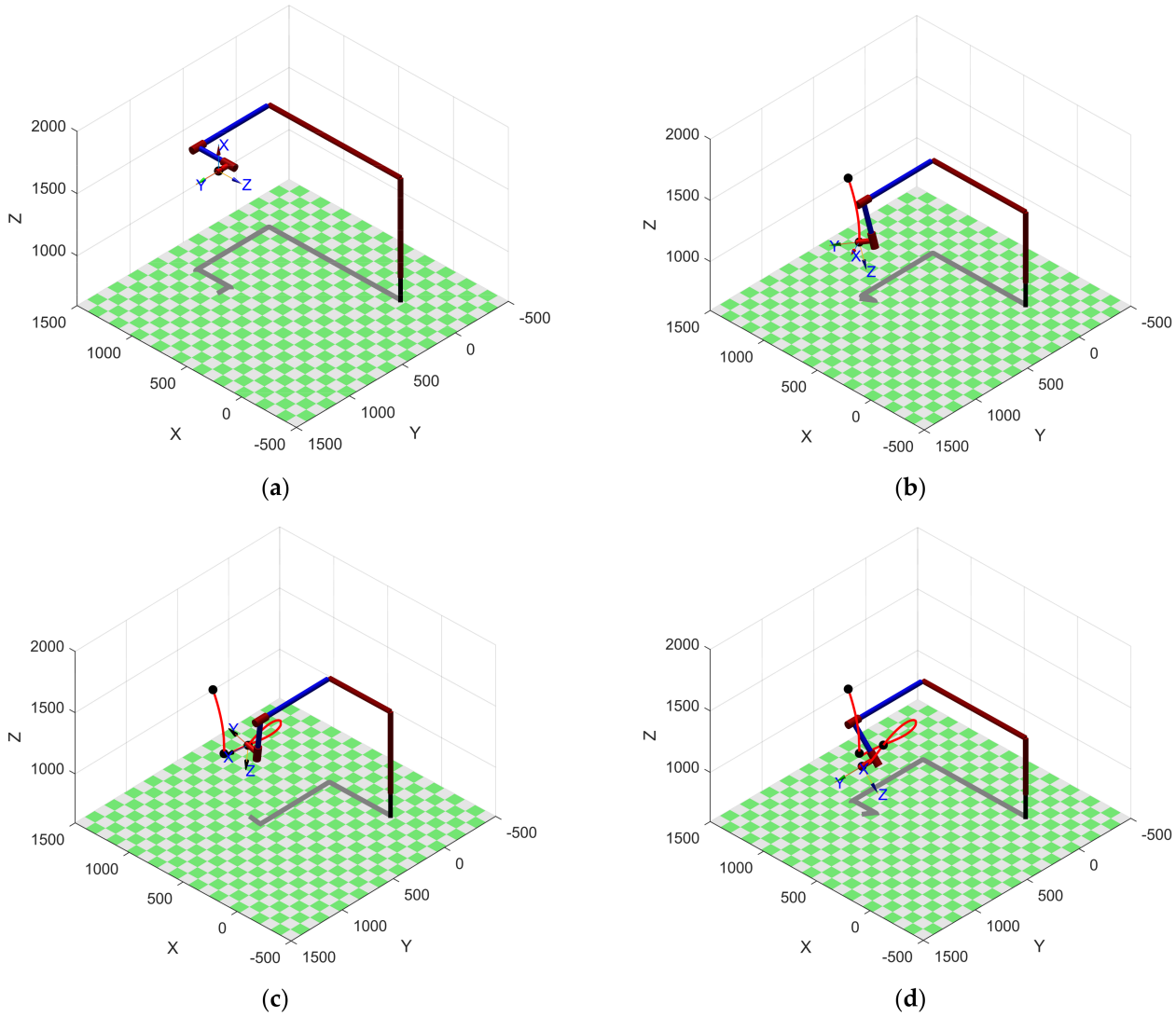

Figure 13. The woodworking manipulator end-effector trajectory under optimal time (**a**) The end-effector at starting point A; (**b**) the end-effector at point B and the trajectory from A to B; (**c**) the end-effector at point C and the trajectory from A to C; (**d**) the end-effector at end point D and the trajectory from A to D.

4. Discussion

For the woodworking manipulator, joints 1 and 2 are rectilinear motion joints, while joints 3 and 4 are revolute joints. Therefore, the limitations of joint velocity and acceleration are different. In the experiment of optimizing joint running time, joints 1 and 2 can be divided into one group experiment, and joints 3 and 4 are another group. From Table 5, the difference in total running time of joints 1 and 2 optimized by GoldS-PSO is 0.1505 s, and the difference in total running time of joints 3 and 4 is 0.0257 s, which is smaller than the 0.2262 s and 0.0681 s of basic PSO and the 0.3993 s and 0.1992 s of Gold-SA, indicating that GoldS-PSO has better stability. The simulation results in Section 3.3 show that the displacement and velocity curves of the manipulator maintain good operational smoothness. However, due to the trajectory being divided into three segments, there is a clear convergence in the operation patterns of the first and third segments, which may not be conducive to meeting the special needs of some machining tasks. Additionally, because of the different interpolation functions used, the acceleration curve can maintain continuity at the connection of two trajectories, but it is not smooth enough and can result in non-differentiable points. But in summary, the simulation results meet the working requirements of the woodworking manipulator.

The method proposed in this paper achieved satisfactory results in time-optimal trajectory planning of woodworking manipulator. However, some areas must be considered for future research.

Firstly, from the perspective of the proposed GoldS-PSO algorithm, further testing is needed to evaluate the effectiveness of different parameter combinations that may need strict mathematical derivations [43]. Moreover, currently we only applied GoldS-PSO to time-optimal trajectory planning of woodworking manipulators. More experiments are needed to assess the effectiveness of other optimization problems in agricultural and forestry engineering, which may require adjustments to the algorithm structure again. Then, after completing the time-optimal trajectory planning, it is necessary to comprehensively consider the effect of energy consumption, impact, and other factors on the operation and carry out multi-objective optimization [44,45] of the trajectory to enhance the general performance of manipulator. The design of the manipulator control system and trajectory tracking control is also the focus of subsequent studies.

Recently, some advanced technologies, such as machine vision, image processing, artificial intelligence algorithms, etc., have been integrated into automated manipulator and robot systems, which has provided unprecedented opportunities for traditional agriculture and forestry industries to achieve an automate decision-making processes and improve efficiency [46,47]. In this process, the GoldS-PSO algorithm proposed in this paper can be used for the training of neural networks and optimization of control model structure in the intelligent control system [48,49], as well as for the optimization of target recognition, positioning, and tracking technology in visual detection systems [50,51]. Meanwhile, some relevant studies indicate that it has great potential in optimizing forestry management structure [52].

5. Conclusions

In this paper, we proposed a time-optimal trajectory planning method based on a 3-5-3 piecewise polynomial interpolation combined with GoldS-PSO algorithm. Aiming to optimize the trajectory of woodworking manipulator in joint space and improve processing efficiency. Firstly, a brief introduction was provided to the structure and operating principles of the woodworking manipulator. Then, we used a modified D-H parameter method to establish a kinematic model of the manipulator for forward and inverse kinematic analysis and used a 3-5-3 piecewise polynomial interpolation to fit three segment trajectories between four path points in joint space. By introducing a Circle chaotic map for population initialization, proposing an S-curve type nonlinear decreasing method to update inertia weight, and integrating the Gold-SA for improving the particle position update formula to obtain the GoldS-PSO, we utilized the commonly used benchmark functions to test the performance of GoldS-PSO, and compared it to basic PSO, SPSO, CFPSO and Gold-SA. Afterward, under the limitations of joints' displacement, velocity, and acceleration, the GoldS-PSO was used to optimize the trajectory interpolation time of each joint, with the target of obtaining the minimum total running time. The experimental results demonstrated that GoldS-PSO decreased each segment's maximum time of 5 s by 2.6811 s, 2.046 s, and 2.3953 s, respectively. Therefore, the total running time obtained via GoldS-PSO optimization was 7.8876 s, which had reduced by 47.483%, 15.695%, and 8.461% form 15 s before optimization, the 9.3442 s for PSO optimization, and the 8.6057 s for Gold-SA optimization, respectively. These findings indicate that GoldS-PSO converges faster and more accurately. The optimized trajectory displacement and velocity curves are smooth and stable, and the acceleration curve is continuous without sudden changes. Our results demonstrate the feasibility and superiority of the proposed method by continuously passing through all the given path points in the Cartesian space in optimal time. In future work, the structure of GoldS-PSO needs further adjustment to improve algorithm performance, and we will carry out multi-objective trajectory optimization for the woodworking manipulator in terms of energy consumption, time, and impact. The GoldS-PSO algorithm proposed in this paper can provide guidance for relative agricultural and forestry engineering optimization problems.

Author Contributions: Conceptualization, S.C. and C.Z.; funding acquisition, C.Z.; methodology, S.C.; resources, C.Z.; software, S.C.; supervision, C.Z.; validation, S.C. and J.Y.; visualization, S.C.; writing—original draft, S.C.; writing—review and editing, S.C. and J.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Fundamental Research Funds of CAF, grant number CAFYBB2020SY043.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, J.; Yang, L.; Liu, H. Green and efficient processing of wood with supercritical CO_2: A Review. *Appl. Sci.* **2021**, *11*, 3929. [CrossRef]
2. Medvedev, S.; Mokhirev, A.; Rjabova, T. Methodical approach to increase efficiency of use of wood resource potential of the region. In Proceedings of the IOP Conference Series: Earth and Environmental Science, Saint Petersburg, Russia, 22–24 May 2019.
3. Pantscharowitsch, M.; Kromoser, B. Influence of machining parameters on subtractive manufacturing of elementary geometries in glued-laminated timber using an industrial robot. *Wood Mater. Sci. Eng.* **2023**, *18*, 472–490. [CrossRef]
4. Ji, M.; Zhang, W.; Diao, X.; Wang, G.; Miao, H. Intelligent automation manufacturing for Betula solid timber based on machine Vision detection and optimization grading system applied to building materials. *Forests* **2023**, *14*, 1510. [CrossRef]
5. Cunha, J.; Ferreira, R.; Lau, N. Computer vision and robotic manipulation for automated feeding of cork drillers. *Mater. Des.* **2015**, *82*, 290–296. [CrossRef]
6. Rossander, M.; Lideskog, H. Design and implementation of a control system for an autonomous reforestation machine using finite state machines. *Forests* **2023**, *14*, 1340. [CrossRef]
7. Zhu, Y.; Qiao, J.; Guo, L. Adaptive sliding mode disturbance observer-based composite control with prescribed performance of space manipulators for target capturing. *IEEE Trans. Ind. Electron.* **2018**, *66*, 1973–1983. [CrossRef]
8. Junge, K.; Hughes, J.; Thuruthel, T.G.; Iida, F. Improving robotic cooking using batch Bayesian optimization. *IEEE Robot. Autom. Lett.* **2020**, *5*, 760–765. [CrossRef]
9. Kim, Y.J.; Cheng, S.; Kim, S.; Iagnemma, K. A stiffness-adjustable hyper redundant manipulator using a variable neutral-line mechanism for minimally invasive surgery. *IEEE Trans. Robot.* **2014**, *30*, 382–395. [CrossRef]
10. Abdelsalam, A.; Happonen, A.; Kärhä, K.; Kapitonov, A.; Porras, J. Toward autonomous vehicles and machinery in mill yards of the forest industry: Technologies and proposals for autonomous vehicle operations. *IEEE Access* **2020**, *10*, 88234–88250. [CrossRef]
11. Qiao, W.; Wang, Z.; Wang, D.; Zhang, L. A new mortise and tenon timber structure and its automatic construction system. *J. Build. Eng.* **2021**, *44*, 103369. [CrossRef]
12. Gao, R.; Zhang, W.; Wang, G.; Wang, X. Experimental research on motion analysis model and trajectory planning of GLT palletizing robot. *Buildings* **2023**, *13*, 966. [CrossRef]
13. Zhao, J.; Wang, S.; Jiang, A.; Xiao, J.; Wang, B. Trajectory planning of 6-dof manipulator based on gaussian process regression method. *Int. J. Robot. Autom.* **2020**, *35*, 209–220. [CrossRef]
14. Huang, J.; Hu, P.; Wu, K.; Zeng, M. Optimal time-jerk trajectory planning for industrial robots. *Mech. Mach. Theory* **2018**, *121*, 530–544. [CrossRef]
15. Stilman, M. Global manipulation planning in robot joint space with task constraints. *IEEE Trans. Robot.* **2010**, *26*, 576–584. [CrossRef]
16. Xu, W.; Li, C.; Wang, X.; Liu, Y.; Liang, B.; Xu, Y. Study on non-holonomic cartesian path planning of a free-floating space robotic system. *Adv. Robot.* **2009**, *23*, 113–143. [CrossRef]
17. Thompson, S.E.; Patel, R.V. Formulation of joint trajectories for industrial robots using B-splines. *IEEE Trans. Ind. Electron.* **1987**, *2*, 192–199. [CrossRef]
18. Gasparetto, A.; Boscariol, P.; Lanzutti, A.; Vidoni, R. Path planning and trajectory planning algorithms: A general overview. *Mech. Mach. Sci.* **2015**, *29*, 3–27.
19. Gasparetto, A.; Zanotto, V. A new method for smooth trajectory planning of robot manipulators. *Mech. Mach Theory* **2007**, *42*, 455–471. [CrossRef]
20. Zhang, X.; Xiao, F.; Tong, X.; Yun, J.; Liu, Y.; Sun, Y.; Tao, B.; Kong, J.; Xu, M.; Chen, B. Time optimal trajectory planning based on improved sparrow search algorithm. *Front. Bioeng. Biotech* **2022**, *10*, 852408.
21. Wu, Y.; You, Y.; Jiang, J. New predictor-corrector methods based on piecewise polynomial interpolation for milling stability prediction. *Mach. Sci. Technol.* **2020**, *24*, 688–718. [CrossRef]
22. Sun, J.; Han, X.; Zuo, Y.; Tian, S.; Song, J.; Li, S. Trajectory planning in joint space for a pointing mechanism based on a novel hybrid interpolation algorithm and NSGA-II algorithm. *IEEE Access* **2020**, *8*, 228628–228638. [CrossRef]
23. Hu, X.; Wu, H.; Sun, Q.; Liu, J. Robot Time Optimal Trajectory Planning Based on Improved Simplified Particle Swarm Optimization Algorithm. *IEEE Access* **2023**, *11*, 44496–44508. [CrossRef]
24. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools. Appl.* **2021**, *80*, 8091–8126. [CrossRef] [PubMed]
25. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
26. Xue, J.; Shen, B. A novel swarm intelligence optimization approach: Sparrow search algorithm. *Syst. Sci. Control Eng.* **2020**, *8*, 22–34. [CrossRef]
27. Mazhoud, I.; Hadj-Hamou, K.; Bigeon, J.; Joyeux, P. Particle swarm optimization for solving engineering problems: A new constraint-handling mechanism. *Eng. Appl. Artif. Intell.* **2013**, *26*, 1263–1273. [CrossRef]
28. Fang, S.; Ru, Y.; Liu, Y.; Hu, C.; Chen, X.; Liu, B. Route planning of helicopters spraying operations in multiple forest areas. *Forests* **2021**, *12*, 1658. [CrossRef]

29. Zhao, J.; Zhu, X.; Song, T. Serial manipulator time-jerk optimal trajectory planning based on hybrid IWOA-PSO algorithm. *IEEE Access* **2022**, *10*, 6592–6604. [CrossRef]
30. Kamel, M.A.; Yu, X.; Zhang, Y. Real-time fault-tolerant formation control of multiple WMRs based on hybrid GA–PSO algorithm. *IEEE Trans. Autom. Sci. Eng.* **2020**, *18*, 1263–1276. [CrossRef]
31. Song, Q.; Yu, L.; Li, S.; Hanajima, N.; Zhang, X.; Pu, R. Energy dispatching based on an improved PSO-ACO algorithm. *Int. J. Intell. Syst.* **2023**, *2023*, 3160184. [CrossRef]
32. Tanyildizi, E.; Demir, G. Golden sine algorithm: A novel math-inspired algorithm. *Adv. Electr. Comput. Eng.* **2017**, *17*, 71–78. [CrossRef]
33. Singh, A.; Singla, A.; Soni, S. Extension of DH parameter method to hybrid manipulators used in robot-assisted surgery. *Proc. Inst. Mech. Eng. Part H* **2015**, *229*, 703–712. [CrossRef]
34. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the IEEE 6th International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43.
35. Hu, W.; Li, S. A simpler and more effective particle swarm optimization algorithm. *J. Softw.* **2007**, *18*, 861–868. (In Chinese) [CrossRef]
36. Varol Altay, E.; Alatas, B. Bird swarm algorithms with chaotic mapping. *Artif. Intell. Rev.* **2020**, *53*, 1373–1414. [CrossRef]
37. Hu, S.; Liu, H.; Feng, Y.; Cui, C.; Ma, Y.; Zhang, G.; Huang, X. Tool Wear Prediction in Glass Fiber Reinforced Polymer Small-Hole Drilling Based on an improved circle chaotic mapping grey wolf algorithm for bp neural network. *Appl. Sci.* **2023**, *13*, 2811. [CrossRef]
38. Chauhan, P.; Deep, K.; Pant, M. Novel inertia weight strategies for particle swarm optimization. *Memet. Comput.* **2013**, *5*, 229–251. [CrossRef]
39. Fan, E. Extended tanh-function method and its applications to nonlinear equations. *Phys. Lett. A* **2000**, *277*, 212–218. [CrossRef]
40. Clerc, M.; Kennedy, J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.* **2002**, *6*, 58–73. [CrossRef]
41. Jin, Y. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm Evol. Comput.* **2011**, *1*, 61–70. [CrossRef]
42. Corke, P.I. A robotics toolbox for MATLAB. *IEEE Robot. Autom. Mag.* **1996**, *3*, 24–32. [CrossRef]
43. Xu, J.; Ren, C.; Chang, X. Robot time-optimal trajectory planning based on quintic polynomial interpolation and improved Harris Hawks algorithm. *Axioms* **2023**, *12*, 245. [CrossRef]
44. Yang, J.; Liu, R.; Tong, Q.; Yang, X.; Liu, Q.; Yao, A. Multi-objective optimization of LCC-S-compensated IPT system for improving misalignment tolerance. *Appl. Sci.* **2023**, *13*, 3666. [CrossRef]
45. Jin, R.; Rocco, P.; Geng, Y. Cartesian trajectory planning of space robots using a multi-objective optimization. *Aerosp. Sci. Technol.* **2021**, *108*, 106360. [CrossRef]
46. Hosseini, S.M.; Peer, A. Wood products manufacturing optimization: A survey. *IEEE Access* **2022**, *10*, 121653–121683. [CrossRef]
47. Kunic, A.; Naboni, R.; Kramberger, A.; Schlette, C. Design and assembly automation of the robotic reversible timber beam. *Autom. Constr.* **2021**, *123*, 103531. [CrossRef]
48. Dong, W.; Xiong, X.; Ma, Y.; Yue, X. Woodworking tool wear condition monitoring during milling based on power signals and a particle swarm optimization-back propagation neural network. *Appl. Sci.* **2021**, *11*, 9026. [CrossRef]
49. Mukherjee, S.; Kumar, R.; Borah, S. An intelligent fast controller for autonomous wheeled robot path navigation in challenging environments. *Ind. Robot* **2022**, *50*, 107–121. [CrossRef]
50. Wang, F.; Xie, B.; Lü, E.; Zeng, Z.; Mei, S.; Ma, C.; Guo, J. Design of a Moisture content detection system for yinghong no. 9 tea leaves based on machine vision. *Appl. Sci.* **2023**, *13*, 1806. [CrossRef]
51. Ji, W.; Chen, G.; Xu, B.; Meng, X.; Zhao, D. Recognition method of green pepper in greenhouse based on least-squares support vector machine optimized by the improved particle swarm optimization. *IEEE Access* **2019**, *7*, 119742–119754. [CrossRef]
52. Qiu, H.; Zhang, H.; Lei, K.; Hu, X.; Yang, T.; Jiang, X. A New Tree-Level Multi-Objective Forest Harvest Model (MO-PSO): Integrating Neighborhood Indices and PSO algorithm to improve the optimization effect of spatial structure. *Forests* **2023**, *14*, 441. [CrossRef]