2024 7th International Conference on Intelligent Robotics and Control Engineering (IRCE) | 979-8-3503-5239-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/IRCE62232.2024.10739825

Time-Jerk Optimal Trajectory Planning of Robot based on Dung Beetle Optimizer Algorithm

Chao Yang[1,2], Qirong Tang[2]*, Wenrui Wang[2], Hao Yang[2], Rongfu Lin[2], Shuqing Wang[2], Qingyun Liu[1]
[1]School of Mechanical Engineering, Anhui University of Technology, Maanshan, China
[2]Laboratory of Robotics and Multibody System, Tongji University, Shanghai, China
e-mail: chaoyang7618@163.com, qirong.tang@outlook.com, wenruiwang@tongji.edu.cn, hao.yang12@outlook.com, rongfulin@tongji.edu.cn, 2330369@tongji.edu.cn, lqyahjx@ahut.edu.cn

***Abstract*—To address the issues of low efficiency and vibration caused by motion impact in robotic operation, a time-jerk optimal trajectory planning method based on the dung beetle optimizer (DBO) algorithm is proposed. Initially, the trajectory of joint space motion is constructed using the 3-5-3 polynomial interpolation method. Then, the time and jerk of robot motion are taken as optimization objectives, with the joint velocity of the robot as a constraint in the optimization problem. The DBO algorithm, initialized with the logistic-tent chaotic mapping, is utilized to optimize these objectives. Simulation and experiments on the UR5 robot demonstrate that the overall motion time of the robot and the maximum jerk on each joint during motion are significantly reduced after optimization using the DBO algorithm, effectively enhancing the operational efficiency and motion stability of the robot.**

Keywords— trajectory planning, 3-5-3 polynomial interpolation, time and jerk, dung beetle optimizer algorithm

I. INTRODUCTION

Robots, with their excellent spatial flexibility and efficient, precise characteristics, are gradually replacing labor-intensive tasks with low production efficiency and high labor costs. The transportation of objects by robots is a common scenario in robotic operations, and ensuring the robots operate more efficiently and smoothly during transportation is particularly important. Trajectory planning, as the cornerstone of robotics technology, profoundly influences task quality. Optimizing trajectories further enhances their performance, increasing both the efficiency of robotic work and motion stability [1,2].

There is an increasing trend in using intelligent algorithms to solve robot trajectory optimization problems. By optimizing the motion trajectory of robots through algorithms, it is possible to decrease working time, lower energy consumption, and minimize impacts during operation. Y. Du et al. [3] address the time-optimal trajectory optimization problem by proposing a locally chaotic particle swarm optimization algorithm based on piecewise polynomials, effectively improving the efficiency of robotic arms. J. Huang et al. [4] utilize fifth-order B-splines to construct trajectory curves, subsequently employing the non-dominated sorting genetic algorithm II (NSGA-II) to optimize both the overall trajectory time and jerk. J. Zhao et al. [5] introduce a trajectory optimization method for robotic arms based on a hybrid particle swarm and whale optimization algorithm. This approach considers the motion time and joint jerk of the robotic arm as optimization objectives, transforming the multi-objective optimization problem into a single-objective optimization problem via a weighted coefficient method, resulting in a substantial reduction in both the motion time and joint jerk of the robotic arm. Z. Wang et al. [6] introduce a multi-objective trajectory optimization technique for robotic arms, enhancing an improved elite non-dominated sorting genetic algorithm with the incorporation of three genetic operators, thus offering a more efficient and stable solution for point-to-point tasks of robotic arms. X. Cao et al. [7] present an improved multi-objective particle swarm optimization algorithm to optimize the time, energy, and jerk functions of a fruit-picking robotic arm. Karahan et al. [8] adopt the interpolation spline method to generate the joint-space trajectory of the robot manipulator and utilized the cuckoo search optimization algorithm (CS) to optimize the time and jerk of the manipulator.

The dung beetle optimization (DBO) algorithm is inspired by the collective behavior of dung beetle populations in nature [9]. This algorithm constructs a search framework founded on the "rolling-breeding-foraging-stealing" model, facilitating effective exploration and exploitation of search space. In contrast to alternative algorithms, it demonstrates attributes of rapid convergence, precision, and stability when confront with intricate high-dimensional optimization tasks. However, the DBO algorithm may suffer from reducing algorithm convergence performance due to population aggregation during population initialization.

In conclusion, this paper aims to enhance the efficiency of robots and reduce joint vibration by comprehensively considering robot operation time and jerk through trajectory optimization research. A 3-5-3 polynomial interpolation technique is employed to craft interpolation curves in joint space, coupled with the integration of the DBO algorithm. Addressing the shortcomings of the DBO algorithm during population initialization, the population is initialized using the logistic-tent chaotic mapping method to improve the solution accuracy and convergence speed of the DBO algorithm. A fitness function incorporating time-jerk is formulated employing the weighted coefficient method, subsequently optimized by the DBO algorithm, thereby devising joint trajectories aimed at minimizing both time and jerk.

II. Trajectory Planning

A. 3-5-3 Polynomial Interpolation

In the process of robot trajectory planning, polynomial interpolation is a common method for trajectory construction. In joint space, the commonly used planning methods are generally cubic or quintic polynomial interpolation [10]. However, the former fails to ensure smooth acceleration changes, while the latter imposes a heavier computational burden. The 3-5-3 polynomial interpolation method adeptly amalgamates the benefits of both approaches, offering distinct advantages in multi-trajectory point planning scenarios [11]. The specific process entails employing cubic polynomial interpolation for the initial segment, quintic polynomial interpolation for the intermediate segment, and reverting to cubic polynomial interpolation for the final segment of the trajectory.

The general formula for the 3-5-3 polynomial interpolation function of the i-th joint is

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

where θ_{ij} represents the polynomial trajectory of the i-th joint in the j-th segment, while a_{ijk} denotes the k-th unknown coefficient of the interpolation polynomial function for the i-th joint's j-th segment. t_1, t_2, t_3 respectively represent the motion time for the 1st, 2nd, and 3rd segments of each joint.

Given the starting point x_{i0}, the path points x_{i1} and x_{i2}, and the ending point x_{i3}, with continuous displacement, velocity, and acceleration between path points, and zero velocity and acceleration at the starting and ending points, all coefficients in Equation (1) can be determined from these constraints. The relationship among the coefficients can be expressed as

$$\boldsymbol{A}=\begin{bmatrix}\boldsymbol{B} & \boldsymbol{C} & \boldsymbol{D}\\ 0 & 0 & \boldsymbol{E}\\ \boldsymbol{F} & 0 & \boldsymbol{G}\end{bmatrix}, \tag{2}$$

$$\boldsymbol{b}=[0\quad 0\quad 0\quad 0\quad 0\quad x_{i3}\quad 0\quad 0\quad x_{i0}\quad 0\quad 0\quad x_{i2}\quad x_{i1}], \tag{3}$$

$$\boldsymbol{a}=\boldsymbol{A}^{-1}\boldsymbol{b}=[\boldsymbol{A}_1\ \boldsymbol{A}_2\ \boldsymbol{A}_3]^T, \tag{4}$$

where

$$\boldsymbol{B}=\begin{bmatrix}t_1^3 & t_1^2 & t_1 & 1 & 0\\ 3t_1^2 & 2t_1 & 1 & 0 & 0\\ 6t_1 & 2 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & t_2^3\\ 0 & 0 & 0 & 0 & 5t_2^4\\ 0 & 0 & 0 & 0 & 20t_2^3\end{bmatrix},\quad \boldsymbol{C}=\begin{bmatrix}0 & 0 & 0 & 0 & -1\\ 0 & 0 & 0 & -1 & 0\\ 0 & 0 & -2 & 0 & 0\\ t_2^4 & t_2^3 & t_2^2 & t_2 & 1\\ 4t_2^3 & 3t_2^2 & 2t_2 & 1 & 0\\ 12t_2^2 & 6t_2 & 2 & 0 & 0\end{bmatrix},$$

$$\boldsymbol{D}=\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & -1\\ 0 & 0 & -1 & 0\\ 0 & -2 & 0 & 0\end{bmatrix},\quad \boldsymbol{E}=\begin{bmatrix}0 & t_3^3 & t_3^2 & t_3 & 1\\ 0 & 3t_3^2 & 2t_3 & 1 & 0\\ 0 & 6t_3 & 2 & 0 & 0\end{bmatrix},$$

$$\boldsymbol{F}=\begin{bmatrix}0 & 0 & 0 & 1 & 0\\ 0 & 0 & 1 & 0 & 0\\ 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\end{bmatrix},\quad \boldsymbol{G}=\begin{bmatrix}0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1\\ 1 & 0 & 0 & 0 & 0\end{bmatrix},$$

$$\begin{cases}\boldsymbol{A}_1=[a_{i13}\quad a_{i12}\quad a_{i11}\quad a_{i10}]\\ \boldsymbol{A}_2=[a_{i25}\quad a_{i24}\quad a_{i23}\quad a_{i21}\quad a_{i20}]\\ \boldsymbol{A}_3=[a_{i33}\quad a_{i32}\quad a_{i31}\quad a_{i30}]\end{cases}.$$

B. Construction of Objective Function

The essence of trajectory optimization resides in the construction of the objective function. To enhance the efficiency of the robot's work while reducing the motion jerk at the joints and increasing the smoothness of the motion, this paper proposes the time-jerk optimization of the robot as the goal and constructs the time-based objective function P_1 and the pulsating jerk-based objective function P_2, which can be expressed as

$$P_1=\sum_{i=1}^{n-1} t_i=t_{all}, \tag{5}$$

$$P_2=\sqrt{\frac{1}{t_{all}}J_i^2dt}, \tag{6}$$

where t_{all} represents the duration of the robot's motion, and n-1 represents the number of segments in the robotic arms's motion trajectory. J_i represents the jerk of the i-th joint, which measures the smoothness of the robotic arms's trajectory.

Based on the established optimization objective function, the objective function is normalized by setting weighting coefficients. The fitness function is defined as

$$f=\omega_1\bullet P_1+\omega_2\bullet P_2, \tag{7}$$

where ω_1 and ω_2 are weighting coefficients, satisfying $\omega_1+\omega_2=1$.

III. Trajectory Optimization Based on Dung Beetle Optimization Algorithm

A. Dung Beetle Optimization

The dung beetle optimization algorithm is primarily inspired by the behaviors of beetles, including rolling, dancing, foraging, stealing, and breeding, it can be used to update positions and optimize. The specific updating process for the four positions of dung beetles is as follows.

1) Rolling dung beetles

The updating of the position of the rolling dung beetle can be categorized into two situations: with and without obstacles. When $\lambda<\gamma$ indicates a state without obstacles, here λ is a random number, and γ is a constant that satisfies $\lambda\in[0,1]$ and $\gamma=0.6$ respectively. The position update can be expressed as

$$\begin{cases}x_i(t+1)=x_i(t)+\alpha\times k\times x_i(t-1)+b\times\Delta x\\ \Delta x=|\ x_i(t)-X^w\ |\end{cases}, \tag{8}$$

where t represents the current iteration number, $x_i(t)$ denotes the position information of the i-th dung beetle at the t-th iteration, α is a natural coefficient indicating whether to deviate from the original direction, assigned either -1 or 1 based on a probabilistic method; $k\in(0,0.2)$ represents the deviation coefficient, $b\in(0,1)$ denotes a constant value; X^w represents the globally

worst position, and Δx is used to simulate changes in light intensity.

When encountering an obstacle denoted by $\lambda \geq \gamma$, the dung beetle cannot move forward. Instead, it simulates dancing behavior by using the tangent function to obtain a new rolling direction, thereby determining a new route. The position update for the rolling dung beetle is defined as

$$x_i(t+1)=x_i(t)+\tan(\theta) \mid x_i(t)-x_i(t-1) \mid , \quad (9)$$

where $\theta \in [0,\pi]$ represents the deflection angle. When θ equals 0, $\pi/2$, or π, the position of the dung beetle will not be updated.

2) Breeding dung beetles

Dung beetles roll dung balls to safe locations, where they bury them to create secure environments for their offspring. This process involves selecting appropriate oviposition sites. Dung beetles employ a boundary selection strategy to mimic the regions where female dung beetles deposit eggs. This strategy is characterized as

$$\begin{cases} Lb^*=\max\left(X^*\times(1-R),Lb\right) \\ Ub^*=\min\left(X^*\times(1+R),Ub\right) \end{cases}, \quad (10)$$

where X^* represents the current local best position, Lb^* and Ub^* respectively denote the lower and upper limits of the oviposition area; $R=1-\frac{t}{T_{\max}}$, $T_{\max}$ represents the maximum number of iterations; Lb and Ub respectively represent the lower and upper bounds of the optimization problem.

Following the determination of the oviposition area, the position of the brood ball dynamically adjusts according to this area. The position of the brood ball is described as

$$B_i(t+1)=X^*+b_1\times(B_i(t)-Lb^*)+b_2\times(B_i(t)-Ub^*), \quad (11)$$

where $B_i(t)$ represents the position information of the i-th brood ball at the t-th iteration, b_1 and b_2 represent two independent random vectors of size $1\times D$, and D represent the dimension of the optimization problem.

3) Small dung beetles

Following hatching, dung beetles search for food within the optimal foraging area. The optimal foraging area is characterized as

$$\begin{cases} Lb^b=\max\left(X^b\times(1-R),Lb\right) \\ Ub^b=\min\left(X^b\times(1+R),Ub\right) \end{cases}, \quad (12)$$

where X^b represents the current local best position, Lb^b and Ub^b respectively denote the lower and upper limits of the optimal foraging area, and other parameters are defined in Equation (10). Therefore, the update of the dung beetle's position is

$$x_i(t+1)=x_i(t)+C_1\times(x_i(t)-Lb^b)+C_2\times(x_i(t)-Ub^b), \quad (13)$$

where $x_i(t)$ represents the position information of the i-th dung beetle at the t-th iteration, C_1 represents a random number following a normal distribution, and C_2 represents a random vector belonging to the interval (0,1).

4) Thief dung beetles

The optimal food source is the most suitable place for competing for food. Thief dung beetles steal dung balls from other dung beetles, and the position information update for thief dung beetles is

$$x_i(t+1)=X^b+S\times g\times(\mid x_i(t)-X^*\mid+\mid x_i(t)-X^b\mid), \quad (14)$$

where $x_i(t)$ represents the position information of the i-th thief at the t-th iteration, g is a random vector of size $1\times D$ following a normal distribution, and S represents a constant, with S=0.2 in this paper.

The effectiveness of swarm intelligence optimization algorithms heavily relies on the quality of population initialization, which significantly influences algorithm performance. The original dung beetle algorithm employs a random population generation method for initialization. However, this random approach diminishes algorithmic diversity, resulting in a degree of blindness in individual optimization. Chaos mapping is a deterministic and random method in nonlinear dynamic systems. During position initialization, chaotic variables substitute random variables within the intelligent algorithm, effectively broadening the solution space exploration beyond traditional probability-based random search strategies. The logistic map sequence has strong global traversal but weak deep search capabilities, while the tent sequence has good chaotic disturbance capabilities [12]. Therefore, this paper employs the logistic-tent chaotic mapping as a method to improve the diversity of population initialization in the dung beetle algorithm, thereby enhancing the solution accuracy and convergence speed of the dung beetle algorithm. The logistic-tent chaotic mapping formula is

$$x_{i+1}=\begin{cases} (rx_i(1-x_i)+(4-r)x_i/2)\bmod 1, & x_i<0.5 \\ (rx_i(1-x_i)+(4-r)(1-x_i)/2)\bmod 1, & x_i\geqslant 0.5 \end{cases} \quad r\in(0,4), \quad (15)$$

where x_i represents the current value of the chaotic variable, and r is the chaotic parameter, with r=1.1 in this paper.

B. The Process Description of the Dung Beetle Optimization Algorithm

Optimizing robot trajectories employing the DBO algorithm with time and jerk as multi-objective functions. The flowchart of the algorithm is shown in Fig. 1.

IV. Simulation and Experiments

A. Simulation Results and Analysis

This paper employs the UR5 robotic arm to simulate the process of module transportation. During the transportation process, the module undergoes "lifting-transfer-lowering" from its initial position to the assembly position. Due to the fixed relative position between the assembly point on the module and the robot end-effector, the spatial coordinates of the four interpolation points representing the module transportation path in the Cartesian coordinate system are transformed into the spatial coordinates of four interpolation points at the end of the robotic arm, denoted as P_0, P_1, P_2, and P_3, respectively. The pose matrices for each interpolation point are derived using the Robotics Toolbox. Subsequently, the pose matrices are transformed into angular interpolation points within the joint space via inverse kinematics, as demonstrated in Table 1.

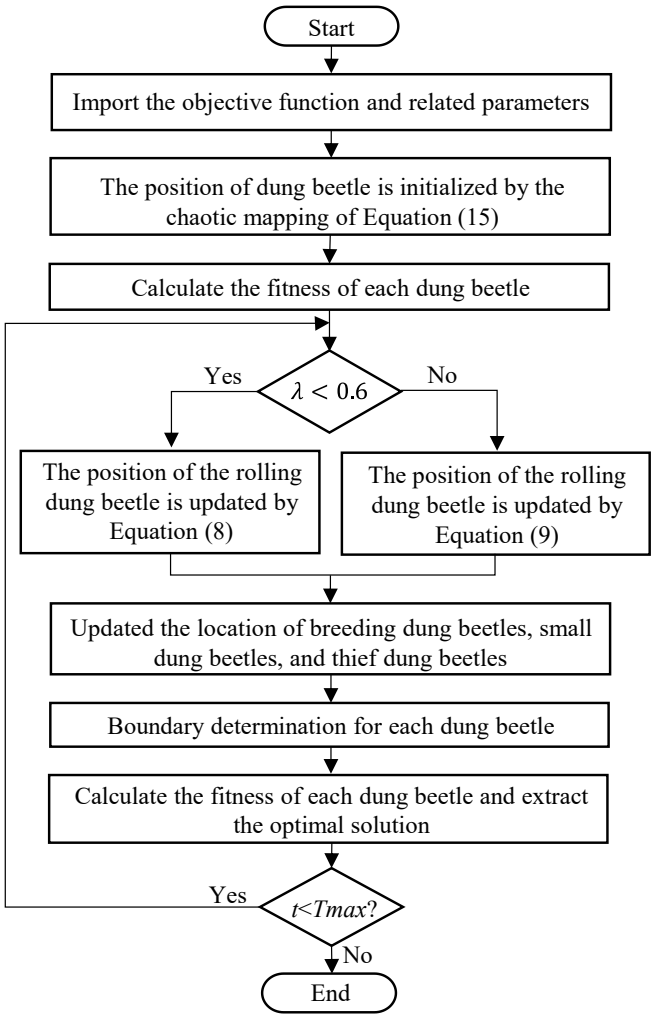

Fig. 1. The flowchart of the dung beetle optimization algorithm

The trajectory planning of the robotic arm employs the 3-5-3 polynomial interpolation method integrated with the DBO algorithm to optimize both the time and jerk aspects of the trajectory. Before optimization, it is defined that each of the three stages of motion of the robotic arm takes 4s, with a total time of 12s. The velocity, acceleration, and jerk at both the initial and final points are set to 0, with the maximum joint velocity limited to 2.5°/s.

TABLE 1. INTERPOLATION POINTS OF JOINT ANGLES.

joint variable	θ_1/(°)	θ_2/(°)	θ_3/(°)	θ_4/(°)	θ_5/(°)	θ_6/(°)
P_0	-7	-115	-100.8	35.7	0	0
P_1	-7	-100.8	-68.4	-10.8	0	0
P_2	-7	-68.4	50.4	-151.2	0	-10.8
P_3	-7	-62.7	91.21	-171.9	0	-36.3

The parameters for the DBO algorithm are configured as follows: the population size N is set to 60, the maximum number of iterations $Tmax$ is 200, and the fitness function weighting coefficients are ω_1=0.6 and ω_1=0.4 . The lower bound is denoted as Lb=[0 0 0], and the upper bounds are denoted as Ub=[4 4 4]. After parameter tuning, with k=0.1 and b=0.3, the DBO algorithm can maintain a certain level of global search capability while ensuring the refinement of local searches. For trajectory planning simulation validation, the most heavily loaded joints, joints 2, 3, and 4, of the robotic arm are selected among the six joints. By recording the optimal positions of the population in each iteration, the population iteration process for joints 2, 3, and 4 is obtained, as shown in Fig. 2.

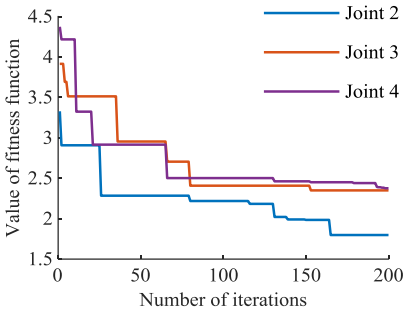

Fig. 2. Time-Jerk optimal population iteration process

Figure 2 clearly illustrates that, in the trajectory optimization algorithm, the populations of the three joints evolve continuously towards time-jerk optimization through iterative processes. The convergence speed is relatively fast in the early iterations and tends to stabilize in the later iterations. The fitness values steadily decrease over time, ultimately converging to the optimal fitness value. Figures 3~5 depict the robot's position, velocity, and acceleration curves before and after optimization using the DBO algorithm. Dashed lines represent the curves before optimization using the DBO algorithm, whereas solid lines represent the curves after optimization.

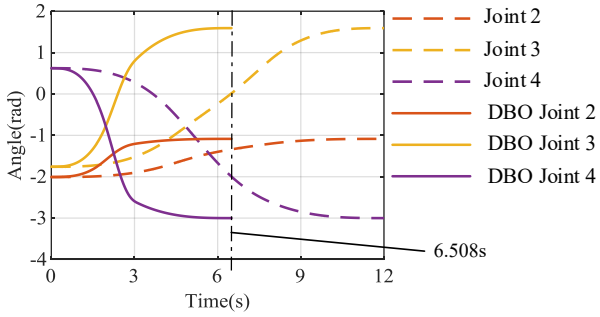

Fig. 3. The joint position curves of 3-5-3 polynomial interpolation

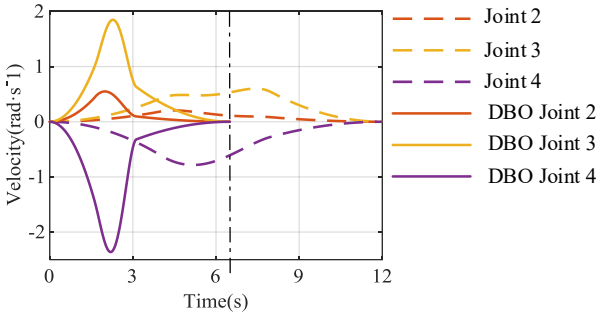

Fig. 4. The joint velocity curves of 3-5-3 polynomial interpolation

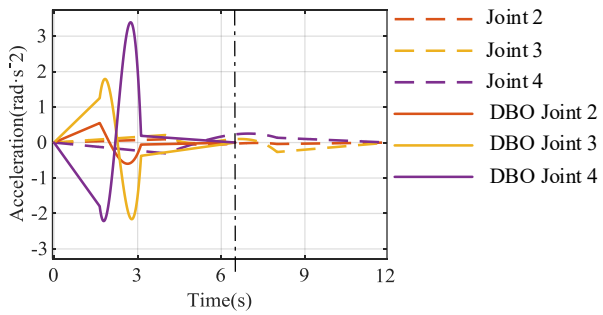

Fig. 5. The joint acceleration curves of 3-5-3 polynomial interpolation

Figures 3~5 reveal that, before optimizing the robot's trajectory for time and jerk using the DBO algorithm, the robot failed to reach its maximum speed due to sluggish motion, consequently leading to reduced motion efficiency. After optimization, the position changes of each joint are smooth, and the velocity and acceleration are continuous. Furthermore, each joint adheres to velocity constraints. The overall motion time is reduced to 6.508s, which is 5.492s lower than before optimization, significantly improving the efficiency of robot operation.

To further validate the effect of the algorithm on joint jerk optimization, the fitness function coefficients are modified by increasing the time weighting coefficient and decreasing the jerk weighting coefficient. Specifically, the fitness function coefficients are adjusted from the original values of ω_1=0.6 and ω_2=0.4 to ω_1=0.2 and ω_2=0.8, where optimizing time becomes the primary objective. The joint jerk curves before and after adjustment are depicted in Fig. 6, with the solid lines illustrating the joint jerk curve before adjustment and the dashed lines representing the curve after adjustment.

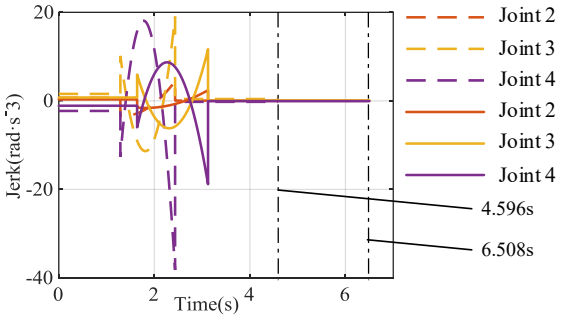

Fig. 6. The joint jerk curves of 3-5-3 polynomial interpolation

Figure 6 indicates that the running time of the time-jerk optimal trajectory algorithm after adjusting the fitness function coefficients is 4.596s. Before adjustment, the average runtime is 1.912s longer compared to after adjustment. However, the maximum jerk generated by the robotic arm's three joints is reduced by an average of 9.591°/s³. This effectively ensures smooth robot motion, thereby validating the effectiveness of the trajectory optimization algorithm proposed in this paper for jerk optimization.

Figure 7 illustrates the trajectory curve of robotic arm module transportation generated by the 3-5-3 polynomial interpolation following dung beetle optimization. It is evident that the resulting smooth trajectory embodies the "lifting-transfer-lowering" process and traverses through the interpolation points. This facilitates the robot in smoothly and efficiently completing the transportation task, thereby validating the effectiveness of this approach.

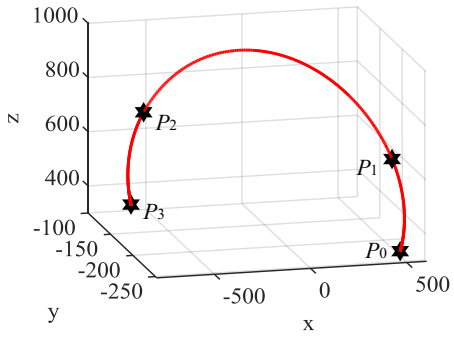

Fig. 7. Optimized robot end trajectory planning curve

B. *Experiments*

To further validate the effectiveness of the proposed trajectory planning method, an experimental platform for robotic arm module transportation is constructed, as shown in Fig. 8. The platform includes an upper computer, UR5 robotic arm, hexagonal prism module, and bracket.

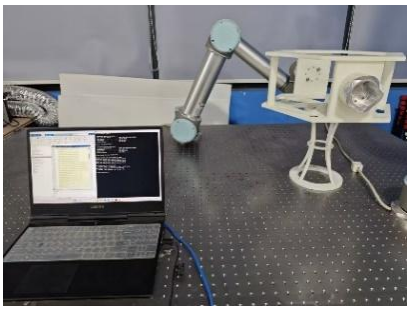

Fig. 8. The experimental platform for robotic arm module transportation

The communication method of the UR5 robot system is illustrated in Fig. 9. After establishing a connection between the upper computer and the robotic arm, control commands and tasks are sent from the upper computer program to the robotic arm via socket. The robotic arm receives the time and angles for joint movements, enabling the joints to move according to the predetermined joint trajectories. The pose graph of the robotic arm passing through four interpolation points during operation is shown in Fig. 10.

Experimental results demonstrate that the robotic arm successfully transports the module from its initial position to the assembly point along the planned trajectory. Moreover, the transportation time is significantly reduced, and the robotic arm exhibits smooth movements with continuous speed variations during the transportation process. These observations further validate the simulation findings and effectively corroborate the feasibility of the trajectory planning approach proposed in this article.

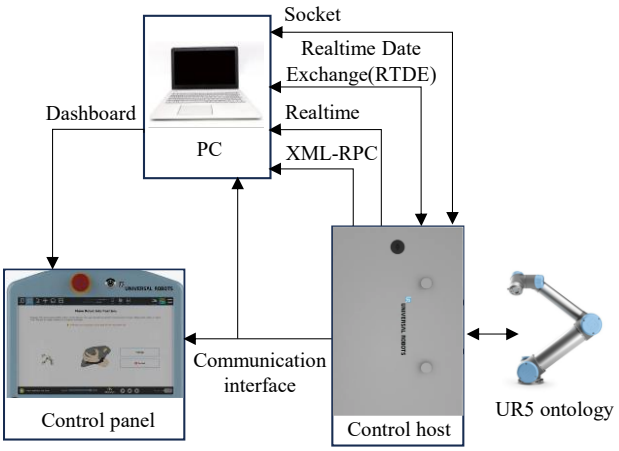

Fig. 9. Communication interface of the UR5 robotic arm system

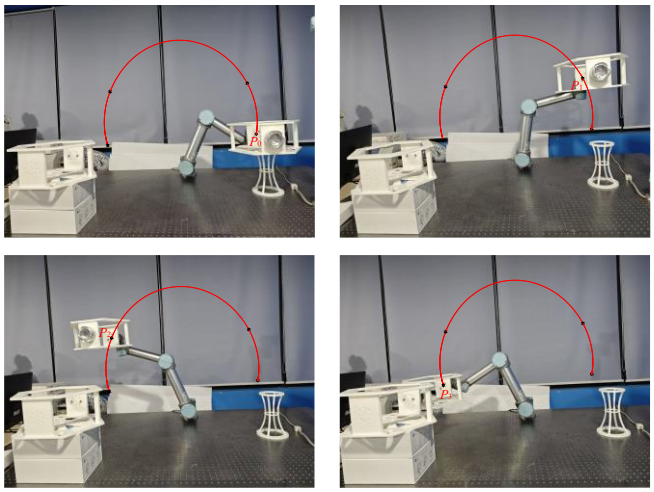

Fig. 10. Working poses of the robotic arms at the four interpolation points

V. Conclusion

This paper addresses the problem of time-jerk optimal trajectory planning for robots in joint space. Based on the 3-5-3 interpolation trajectory planning, the high convergence and stability of the DBO algorithm are employed to optimize robot trajectories. In the simulation, the joint positions, velocities, and acceleration curves before and after optimization with the dung beetle algorithm are compared, as well as the jerk curves before and after adjusting the weighting coefficients. The simulation results highlight the substantial reduction in motion time, the effective mitigation of maximum joint jerk, the enhancement of robot efficiency, and the improvement in motion stability achieved through trajectory optimization utilizing the DBO algorithm. In the experiments, the robotic arm can efficiently and smoothly transport the module from the initial position to the assembly position, effectively validating the feasibility of the trajectory planning method proposed in this paper. The energy consumption of the robotic arm during operation is not considered in this paper. In future research, energy consumption will be added as an optimization objective to achieve multi-objective comprehensive optimization, providing further assistance in improving the working performance of the robotic arm.

Acknowledgment

This work is supported by the project of National Natural Science Foundation of China (No.62373285), the Shanghai 2021 “Science and Technology Innovation Action Plan” with Special Project of Biomedical Science and Technology Support (No.21S31902800), and the Key Pre-Research Project of the 14th-Five-Year-Plan on Common Technology. Meanwhile, this work is also partially supported by the “National High Level Overseas Talent Plan” project, the “National Major Talent Plan” project (No. 2022-XXXX-XXX-079), one key project (No.XM2023CX4013), and the Shanghai Pujiang Programme (No. 23PJD103). It is also partially sponsored by the fundamental research project (No. XXXX2022YYYC133), the Shanghai Industrial Collaborative Innovation Project (Industrial Development Category, No. HCXBCY-2022-051), the project of Shanghai Key Laboratory of Spacecraft Mechanism (No. 18DZ2272200), the project of Space Structure and Mechanism Technology Laboratory of China Aerospace Science and Technology Group Co. Ltd (No. YY-F805202210015), as well as the project of National Laboratory of Space Intelligent Control (No. HTKJ2023KL502016). All these supports are highly appreciated.

References

[1] G. Li and Y. Wang. Industrial robot optimal time trajectory planning based on genetic algorithm. In 2019 IEEE International Conference on Mechatronics and Automation (ICMA), pp. 136-140, August 4-7, 2019, Tianjin, China.

[2] K. Wang, J. Li, Y. Zou, C. Wang, and L. Liang.Trajectory planning for a articulated robot. Applied Mechanics and Materials, 742: 576-581, 2015.

[3] Y. Du and Y. Chen. Time optimal trajectory planning algorithm for robotic manipulator based on locally chaotic particle swarm optimization. Chinese Journal of Electronics, 31(5): 906-14, 2022.

[4] J. Huang, P. Hu, K. Wu, and M. Zeng. Optimal Time-jerk Trajectory Planning for Industrial Robots. Mechanism and Machine Theory, 121: 530-544, 2018.

[5] J. Zhao, X. Zhu, and T. Song. Serial manipulator time-jerk optimal trajectory planning based on hybrid iwoa-pso algorithm. IEEE Access, 10: 6592-6604, 2022.

[6] Z. Wang, Y. Li, K. Shuai, W. Zhu, B. Chen, and K. Chen. Multi-objective trajectory planning method based on the improved elitist non-dominated sorting genetic algorithm. Chinese Journal of Mechanical Engineering, 35(1): 7, 2022.

[7] X. Cao, H. Yan, Z. Huang, S. Ai, Y. Xu, R. Fu, and X. Zou. A multi-objective particle swarm optimization for trajectory planning of fruit picking manipulator. Agronomy, 11(11): 2286, 2021.

[8] O. Karahan, H. Karci, and A. Tangel. Optimal trajectory generation in joint space for 6r industrial serial robots using cuckoo search algorithm. Intel Serv Robot, 15: 627–648, 2022.

[9] J. Xue and B. Shen. Dung beetle optimizer: A new meta-heuristic algorithm for global optimization. The Journal of Supercomputing, 79(7): 7305-7336, 2023.

[10] Z. Ang, C. Ang, W. Lim, L. Yu, and M. Solihin. Development of an artificial intelligent approach in adapting the characteristic of polynomial trajectory planning for robot manipulator. International Journal of Mechanical Engineering and Robotics Research, 9(3): 408-414, 2020.

[11] B. Liang, X. Lin, G. Liu, J. Lei, and W. Wang. Trajectory analysis and optimization of sea buckthorn fruit vibration separation manipulator based on I-PSO algorithm. Scientific Reports, 13(1): 20124, 2023.

[12] J. Hou, W. Jiang, Z. Luo, L. Yang, X. Hu, and B. Guo. Dynamic Path Planning for Mobile Robots by Integrating Improved Sparrow Search Algorithm and Dynamic Window Approach. Actuators, 13(1): 24, 2023.