Mechatronic Systems and Control, Vol. 53, No. 10, 2025
An Open Access Paper

TIME-OPTIMAL TRAJECTORY PLANNING METHOD OF MANIPULATOR BASED ON NMSDBO ALGORITHM

Bo Xue,* Mengcheng Lin,* and Xiangyu Wu*

Abstract

In the trajectory planning process of a robotic arm, it is essential to ensure that the driving device meets the actual load requirements. Consequently, the selection of joint velocities and accelerations tends to be relatively conservative. This cautious approach results in an extended duration required to complete a set of actions, ultimately hindering the full utilisation of the robotic arm's continuity and stability based on its motion velocity and acceleration. To address the optimisation problem concerning the velocity and acceleration of various joints in a robotic arm, this paper focuses on the IRB2600 robotic arm as the subject of study. We introduce a 3-5-3 polynomial interpolation trajectory planning method within joint space and propose a time-optimal trajectory planning method based on a new multi-strategy improved DBO (NMSDBO) algorithm. This algorithm integrates the golden sine strategy, and incorporates adaptive T-distribution perturbation, which enhances the update of global optimal positions and improves both the accuracy and speed of the trajectory optimisation process. By comparing with WOA, HHO, SSA, and DBO algorithms in six benchmark functions, it is proved that the proposed NMDBO has good performance and robustness. The simulation results show that the time for each joint to complete the action is shortened from the initial set of 12 s to 5.5548 s, which achieves the effect of time optimisation and maintains the smooth and compliant motion of the manipulator.

Key Words

Robotic arm; polynomial interpolation; time-optimal trajectory planning; dung beetle optimiser algorithm; adaptive T-distribution

1. Introduction

Robot trajectory planning serves as the foundation for controlling manipulator movement, the quality of the trajectory planning has an essential influence on the quality of the operation [1], [2]. Furthermore, as a prerequisite for trajectory tracking control, effective trajectory planning influences various factors such as accuracy, motion efficiency, smoothness of movement, and energy consumption [3], [4]. The trajectory planning can be categorised into Cartesian space trajectory planning and joint space trajectory planning according to the planning space. Within joint space trajectory planning, methods can be further divided into general trajectory planning and optimal trajectory planning depending on whether an optimisation approach is employed to enhance performance. Joint space general trajectory planning refers to the process of maneuvering each joint of a robotic arm according to a specified planning methodology within the robot's joint space. This process involves calculating the functional relationships among joint angle, angular velocity, angular acceleration, and time for each individual joint. Optimal trajectory planning is a single objective optimisation based on time, energy, impact, *etc.,* or a multi-objective trajectory optimisation that integrates various objectives based on specific requirements, to further improve the performance of trajectory planning [5]. Currently, research on control algorithms for optimising robotic arm time is relatively in-depth, aiming to shorten operating time and improve work efficiency.

The primary optimisation algorithms employed for time-optimal trajectory planning problems encompass a range of techniques, including dichotomy, particle swarm optimisation (PSO), the grey wolf optimiser (GWO) algorithm, the sparrow search algorithm (SSA), the whale optimisation algorithm (WOA), the butterfly optimisation algorithm (BOA), as well as sine-cosine algorithms (SCA) [6] *etc.* Fan *et al.* [7] designed a time-optimal trajectory planning algorithm based on the improved GWO algorithm, they optimised the time parameters of the quintic polynomial, and applied it in the camellia fruit picking manipulator. Zhang *et al.* [8] employed cubic spline curve to establish the trajectory of manipulator in joint space, combined with the cosine annealing algorithm to design adaptive step factor, and realised time-optimal planning based on enhanced SSA. Zhao *et al.* [9] utilised quintic B-spline interpolation to construct the manipulator's trajectory within joint space, and developed an improved WOA that combines the PSO with the whale

* School of Electrical Information Engineering, Jiangsu University of Technology, Changzhou 213001, China; e-mail: dxxb@jsut.edu.cn; 245138035@qq.com; 1572270010@qq.com
Corresponding author: Bo Xue

Recommended by Jun Ma
(DOI: 10.2316/J.2025.201-0573)

optimisation technique (which is also referred as the hybrid whale PSO algorithm in [10]). This innovative method optimised trajectory time, resulting in more efficient operation of the manipulator. Zhou *et al.* [11] employed a polynomial interpolation algorithm to construct the trajectory of a six-degree-of-freedom manipulator. They enhanced the traditional BOA and applied it to optimise the time of the manipulator's trajectory. This approach not only effectively reduces the motion time of the manipulator but also ensures smooth operation during actual production processes. Pan *et al.* [12] proposed a time-optimal polynomial interpolation trajectory planning algorithm based on an improved sine-cosine algorithm, which dynamically adjusts the balance factor according to fitness levels, thereby optimising the local development capability of the algorithm. Patle *et al.* [13] conducted trajectory planning based on an S-shaped velocity curve and employed PSO to determine the optimal time parameters for this curve. Although extensive research has been conducted on the trajectory optimisation of robotic arms, they still face problems such as single or high polynomial interpolation orders, complex calculations, slow convergence speed of the optimisation algorithms, limited ability to explore the global optimal position, and long optimisation time.

Xue *et al.* [14] introduced the dung beetle optimiser (DBO) algorithm, which is inspired by the behaviour of beetles. This algorithm effectively combines characteristics of global exploration and local development, and it is known for its rapid convergence and high solution accuracy. Recently, scholars have made significant advancements in the algorithm. Pan *et al.* [15] introduced the MSADBO algorithm, which integrates an enhanced sine algorithm by employing Bernoulli chaotic mapping for initialisation and incorporating an adaptive Gaussian Cauchy mutation perturbation strategy. Guo *et al.* [16] proposed the MIDBO algorithm, which enhances the acceptance rate of optimal solutions by thief beetles. Additionally, it incorporates a perturbation strategy derived from the SSA and employs Cauchy Gaussian mutation techniques. Li *et al.* [17] proposed the IDBO algorithm, which initialises the population by integrating Fuch chaos and reverse learning strategies. This approach incorporates adaptive step size and convex lens imaging techniques, while also introducing a random difference mutation strategy. Research has demonstrated that the enhanced DBO algorithm exhibits superior optimisation capabilities and a faster convergence rate when compared to other path planning methodologies. Nevertheless, these advancements still face issues such as uneven distribution of population initialisation, imbalance between global exploration and local exploitation, and susceptibility to getting stuck in local optima, which can affect convergence performance.

In this paper, the motion trajectory is divided into three segments by combining the advantages of cubic and quintic trajectory planning, and the 3-5-3 segmented trajectory planning is implemented through a combination of cubic, quintic, and cubic polynomials. To tackle the challenge of optimising time parameter, we have designed a NMSDBO algorithm for trajectory optimisation that achieves time-optimal solutions while adhering to kinematic constraints. The algorithm converts constrained optimisation problems into unconstrained ones by incorporating penalty functions. By integrating the golden sine strategy and introducing adaptive T-distribution perturbations, the rolling behaviour of the beetle optimisation algorithm is enhanced, along with the evolution of the global optimal position. This approach significantly improves both the accuracy and speed of the optimisation process.

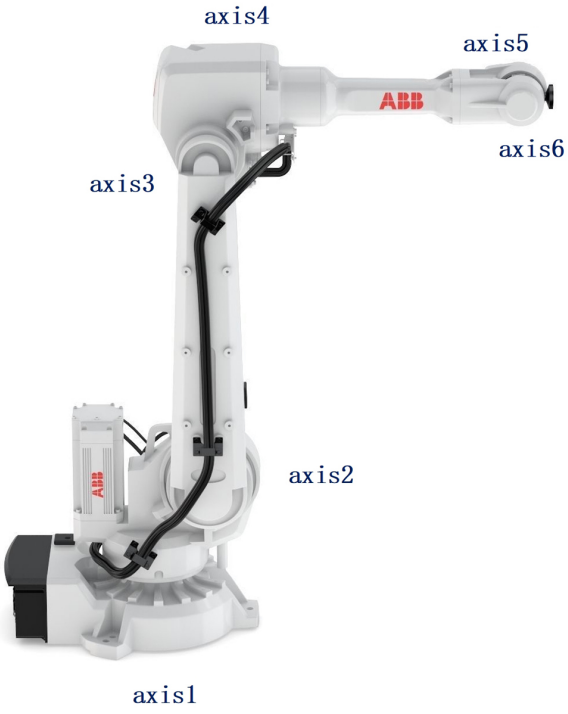

Figure 1. IRB2600 manipulator.

2. Kinematics Model and Trajectory Planning of Manipulator

2.1 Kinematics Model of IRB2600 Manipulator

Industrial robots consist of a variety of intricate components, the primary elements include mechanical arms, end-effectors, motor systems, sensors, and various associated accessories. As illustrated in Fig. 1, using ABB's IRB2600 industrial robot as a case study [18], the kinematic modelling of industrial robots is presented in detail. In the 1950s, Denavit and Hartenberg introduced the Denavit-Hartenberg (DH) method, which remains a widely utilised framework for robot description. The DH method is categorised into two types: the standard DH method (SDH) and the modified DH method (MDH). The former establishes the coordinate system at the distal end of connecting rod, while the latter defines it at the proximal end. Due to its enhanced adaptability, this paper adopts the MDH approach for modelling the IRB2600 manipulator.

Table 1
MDH Parameter Table of Manipulator

Connecting rod	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	d_1 (445)	θ_1
2	-90°	a_1 (150)	0	$\theta_2 + 90^\circ$
3	0	a_2 (−700)	0	θ_3
4	90°	a_3 (−115)	d_4 (795)	θ_4
5	-90°	0	0	θ_5
6	90°	0	d_6 (85)	θ_6

Figure 2 illustrates the coordinate system of the manipulator, which has been established in accordance with the MDH convention. Based on the structure of the IRB2600 depicted in Fig. 2, the corresponding MDH parameters can be derived and are presented in Table 1. In the table, α_{i-1}, a_{i-1}, d_i, and θ_i represents the torsion angle of the connecting rod, the length of the connecting rod, the offset of the connecting rod and the joint angle, respectively. Among these parameters, the torsion angle α_{i-1} denotes the rotation angle between the adjacent joint axes, the length a_{i-1} signifies the vertical distance between the adjacent axes, the offset distance d_i indicates the translation distance between the previous joint and the current joint axis, and the joint angle θ_i represents the angle formed by the common vertical line of the connecting rod before and after the translation and the rotation around the joint axis.

2.2 3-5-3 Polynomial Trajectory Planning

The cubic polynomial interpolation method for trajectory planning in joint space has gained widespread adoption due to its straightforward computational requirements. However, the trajectories generated by this method do not impose constraints on angular acceleration, which can lead to abrupt changes in the manipulator's joint angular acceleration. This phenomenon may result in mechanical vibrations, ultimately diminishing both the service life and control accuracy of the servo motor. Based on the cubic polynomial, the quintic polynomial enhances the constraints on joint angular acceleration and mitigates potential damage to the joint motor due to its higher order. However, employing quintic polynomial interpolation for each trajectory significantly increases computational demands. Therefore, we propose a 3-5-3 piecewise polynomial interpolation trajectory planning algorithm that integrates the advantages of both cubic and quintic trajectory planning, thereby achieving a balanced planning effect.

The 3-5-3 polynomial trajectory planning method divides the motion trajectory into three distinct phases, taking into account joint motion dynamics and smoothness requirements as follows.

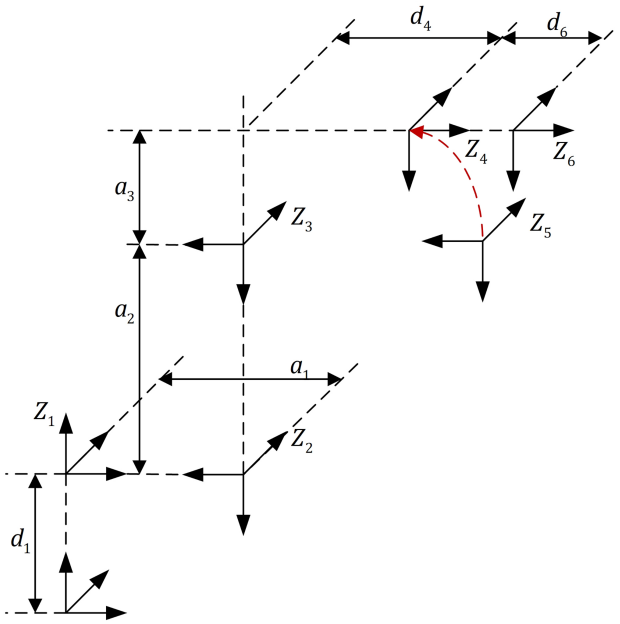

Figure 2. IRB2600 manipulator coordinate system.

(1) Initial phase (cubic polynomial), which is designed to swiftly attain an intermediate velocity while minimising acceleration jerk;
(2) Middle phase (quintic polynomial), which imposes stringent constraints on joint angular acceleration to ensure a smooth motion profile;
(3) Final phase (cubic polynomial), which gradually decelerates to achieve zero velocity. The formula for the 3-5-3 piecewise polynomial is as follows

$$\begin{cases} \theta_1(t) = a_{10} + a_{11}t + a_{12}t^2 + a_{13}t^3 \\ \theta_2(t) = a_{20} + a_{21}t + a_{22}t^2 + a_{23}t^3 + a_{24}t^4 + a_{25}t^5 \\ \theta_3(t) = a_{30} + a_{31}t + a_{32}t^2 + a_{33}t^3 \end{cases} \quad (1)$$

where, $\theta_1(t)$, $\theta_2(t)$, and $\theta_3(t)$ represents the joint angles at the time t for trajectory of the first, second and third segments, respectively. The coefficients a_{1i}, a_{2i}, and a_{3i} correspond to the first, second, and third trajectories, respectively.

The velocity expression of 3-5-3 polynomial trajectory planning as follows

$$\begin{cases} \dot{\theta}_1(t) = a_{11} + 2a_{12}t + 3a_{13}t^2 \\ \dot{\theta}_2(t) = a_{21} + 2a_{22}t + 3a_{23}t^2 + 4a_{24}t^3 + 5a_{25}t^4 \\ \dot{\theta}_3(t) = a_{31} + 2a_{32}t + 3a_{33}t^2 \end{cases} \quad (2)$$

where, $\dot{\theta}_1(t)$, $\dot{\theta}_2(t)$, and $\dot{\theta}_3(t)$ denote the joint velocity at the first, second, and third trajectories, respectively.

The acceleration expression of 3-5-3 polynomial trajectory planning as follows

$$\begin{cases} \ddot{\theta}_1(t) = 2a_{12} + 6a_{13}t \\ \ddot{\theta}_2(t) = 2a_{22} + 6a_{23}t + 12a_{24}t^2 + 20a_{25}t^3 \\ \ddot{\theta}_3(t) = 2a_{32} + 6a_{33}t \end{cases} \tag{3}$$

where, $\ddot{\theta}_1(t)$, $\ddot{\theta}_2(t)$, and $\ddot{\theta}_3(t)$ denote the joint acceleration at the first, second, and third trajectories, respectively.

In the context of 3-5-3 polynomial trajectory planning, to ensure the smoothness of the trajectory, it is necessary to set the constraints of angle, angular velocity, and angular acceleration at the connection of the three trajectories. This procedure requires solving 14 sets of 4 equations to determine 14 unknown coefficients, which will ultimately define the specific form of the polynomial. The detailed solution process is outlined as follows.

The joint angle equation, derived from the angular information, is as follows

$$\begin{cases} \theta_1(t_1) = \theta_2(0) \\ \theta_2(t_2) = \theta_3(0) \\ \theta_3(t_3) = \theta_3 \\ \theta_1(0) = \theta_0 \\ \theta_2(0) = \theta_1 \\ \theta_3(0) = \theta_2 \end{cases} \tag{4}$$

where, the equations are formulated based on the angle continuity of the three-segment trajectory, incorporating the known initial and final joint angles θ_0 and θ_3, as well as the interpolation points θ_1 and θ_2.

The joint velocity equation formulated based on the velocity information at the interpolation point is as follows

$$\begin{cases} \dot{\theta}_1(t_1) = \dot{\theta}_2(0) \\ \dot{\theta}_2(t_2) = \dot{\theta}_3(0) \\ \dot{\theta}_3(t_3) = 0 \\ \dot{\theta}_1(0) = 0 \end{cases} \tag{5}$$

where, the equations are formulated based on the continuity of the angular velocity of the two interpolation points of the three-segment trajectory. It is also given that both the initial and final joint angular velocities are zero.

By integrating (1) through (5), we can derive the expressions for the unknown coefficient matrix a, the joint interpolation angle matrix b, and the three-stage trajectory time as follows

$$Aa = b \tag{6}$$

where,

$$A = \begin{bmatrix} {t_1}^3 & {t_1}^2 & t_1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 3{t_1}^2 & 2t_1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 6t_1 & 2 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & {t_2}^5 & {t_2}^4 & {t_2}^3 & {t_2}^2 & t_2 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 5{t_2}^4 & 4{t_2}^3 & 3{t_2}^2 & 2t_2 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 20{t_2}^3 & 12{t_2}^2 & 6t_2 & 2 & 0 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & {t_3}^3 & {t_3}^2 & t_3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3{t_3}^3 & 2t_3 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6t_3 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

$$b = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \theta_3 & 0 & 0 & \theta_0 & 0 & 0 & \theta_2 & \theta_1 \end{bmatrix}^T \tag{8}$$

Through (6) to (8), the coefficient matrix a can be derived as follows

$$a = A^{-1}b = \begin{bmatrix} a_{13} & a_{12} & a_{11} & a_{10} & a_{25} & a_{24} & a_{23} & a_{22} & a_{21} & a_{20} \\ a_{33} & a_{32} & a_{31} & a_{30} \end{bmatrix}^T \tag{9}$$

The 3-5-3 piecewise polynomial function can be derived by substituting the coefficient matrix a back into (1).

3. Time-optimal Trajectory Planning Algorithm

3.1 Mathematical Model of Time-optimal Trajectory Planning

It can be observed from (7) that the objective function is solely dependent on time t. The motion process of the robot within the joint space is segmented into three interpolation trajectories, which are subsequently fitted using a 3-5-3 piecewise polynomial. By aiming to minimise the total time as the optimisation goal, the total time parameter for the 3-5-3 polynomial trajectory planning is denoted by Γ, and the objective function for time-optimal trajectory planning is presented in (10)

$$\Gamma = \min(t_{i1} + t_{i2} + t_{i3}) \tag{10}$$

where, t_{i1}, t_{i2}, and t_{i3} denote the three stages motion duration for the i-th joint in the 3-5-3 polynomial trajectory planning.

After formulating the objective function, the maximum angular velocity and maximum angular acceleration for

each joint rotation are employed as constraints. The constraint function is established as illustrated in (11)

$$\begin{cases} \left|\dot{\theta}_i\right| \leq \dot{\theta}_{\max} \\ \left|\ddot{\theta}_i\right| \leq \ddot{\theta}_{\max} \end{cases} \tag{11}$$

where, $\left|\dot{\theta}_i\right|$ denotes the absolute value of angular velocity for the i-th joint during motion, $\dot{\theta}_{\max}$ signifies the maximum allowable angular velocity, $\left|\ddot{\theta}_i\right|$ represents the absolute value of angular acceleration for the i-th joint while in motion, and $\ddot{\theta}_{\max}$ indicates the maximum permissible angular acceleration. The constraint conditions necessitate the determination of the maximum angular velocity and acceleration for each joint. This involves calculating the peak values of both the first and second derivatives of the interpolation function.

Addressing the time-optimal constraint problem directly will result in high complexity. To effectively simplify the solution process, we employ a penalty function strategy to transform the constraint issues related to velocity and acceleration of the manipulator into an unconstrained problem, thereby mitigating the complexity associated with optimisation challenges. The constraint functions presented in (10) through (11) are reformulated into an unconstrained function as shown in (12)

$$\begin{aligned} \Gamma = \min\left(t_{i1}+t_{i2}+t_{i3}\right) + \sigma \times \left(\sum_{i=1}^{3} \max\left(0, \left|\dot{\theta}_i\right| - \dot{\theta}_{\max}\right)^2 \right. \\ \left. + \sum_{i=1}^{3} \max\left(0, \left|\ddot{\theta}_i\right| - \ddot{\theta}_{\max}\right)^2\right) \end{aligned} \tag{12}$$

where, σ denotes the penalty factor coefficient with a value of 10^{10}.

3.2 Dung Beetle Optimiser Algorithm

The DBO is a single-objective global optimisation algorithm introduced by Xue *et al.* [14] in 2022. Drawing inspiration from the distinctive behaviours of dung beetles, this algorithm emulates various activities such as rolling, dancing, foraging, stealing, and breeding within their natural environment. It incorporates five unique location update strategies that not only facilitate rapid convergence of the algorithm but also ensure solution accuracy. Furthermore, it enables the identification of both global and local optimal solutions. The mathematical model is formulated based on the five behavioural rules of rolling, dancing, breeding, foraging, and stealing as observed in the dung beetle algorithm.

3.2.1 Ball Rolling Behaviour

The ball rolling behaviour of dung beetles can be categorised into barrier-free mode and obstacle mode. In the absence of obstacles during their movement, dung beetles utilise light intensity to determine a path for both position and direction while rolling. As they engage in this rolling activity, the position of the dung beetle is updated as described in (13)

$$x_i\left(t+1\right) = x_i\left(t\right) + \alpha \times k \times x_i\left(t-1\right) + b \times \left|x_i\left(t\right) - X^W\right| \tag{13}$$

where, $x_i\left(t\right)$ represents the position information of the first dung beetle in the i-th iteration, t denotes the current number of iterations, α signifies the coefficient of natural factors (such as wind and uneven terrain), with a value of either -1 or 1, k indicates the deflection coefficient, typically set at 0.1; b is a constant within a specified range, usually taken to be 0.3, and finally, X^W represents the global worst position.

3.2.2 Dancing Behaviour

When the dung beetle encounters obstacles on the way forward, it must reorient itself by performing a dance. The algorithm employs the tangent function to determine a new direction, and the position update is illustrated in (14)

$$x_i\left(t+1\right) = x_i\left(t\right) + \tan\left(\theta\right)\left|x_i\left(t\right) - x_i\left(t-1\right)\right| \tag{14}$$

where, θ represents the deflection angle, $\theta \in (0, \pi)$. It is important to note that when $\theta = 0, \frac{\pi}{2}, \pi$, the position of the dung beetle remains unchanged. This indicates that the position update of the dung beetle is closely linked to both its current position $x_i\left(t\right)$ and historical positional information $x_i\left(t-1\right)$.

3.2.3 Breeding Behaviour

In nature, dung beetles must take into account the living environment of their offspring and will seek out safe and suitable habitats that promote the survival of young dung beetles. Building on this understanding, the boundary restriction strategy of (15) is proposed to effectively simulate the breeding offspring behaviour of dung beetles

$$\begin{cases} Lb^* = \max\left(X^* \times \left(1-R\right), \mathrm{Lb}\right) \\ Ub^* = \min\left(X^* \times \left(1+R\right), \mathrm{Ub}\right) \end{cases} \tag{15}$$

$$R = 1 - \frac{t}{T_{\max}} \tag{16}$$

where, Lb^* and Ub^* denote the lower and upper limits of the spawning area, while X^* indicates the current local optimal position. Additionally, Lb and Ub represents the lower and upper bounds of the optimisation problem, and $T_{\max}$ is the maximum number of iterations.

The conceptual diagram illustrating the boundary processing strategy is presented in Fig. 3. The current local optimal position is denoted as X^*, while the surrounding black dots represents the breeding balls. The boundary parameters Lb^*, Ub^*, Lb, and Ub are indicated by red dots within the conceptual map.

After a dung beetle identifies the spawning area, it engages in breeding behaviour, with each dung beetle producing only one egg per iteration. Due to the dynamic adjustment of the spawning area, the position of the small

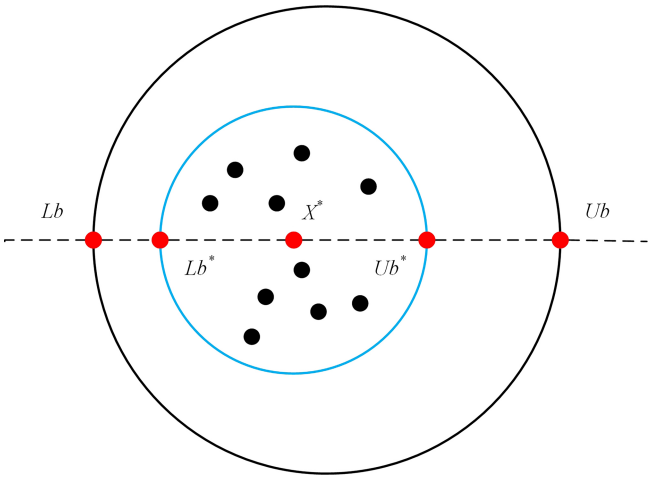

Figure 3. Boundary selection strategy model.

dung beetle during the feeding process remains uncertain. The update mechanism for each dung beetle's breeding ball position is determined according to (17)

$$x_i(t+1) = X^* + b_1 \times (x_i(t) - Lb^*) + b_2 \times (x_i(t) - Ub^*) \tag{17}$$

where, $x_i(t)$ denotes the positional information of the i-th breeding dung beetle at the t-th iteration, while b_1 and b_2 represents random vectors of size $1 \times D$.

3.2.4 Foraging Behaviour

The optimal foraging range Lb^b and Ub^b are calculated using (17), while the foraging position of dung beetle is determined by (18)

$$\begin{cases} Lb^b = \max\left(X^b \times (1-R), \mathrm{Lb}\right) \\ Ub^b = \min\left(X^b \times (1+R), \mathrm{Ub}\right) \end{cases} \tag{18}$$

$$x_i(t+1) = x_i(t) + C_1 \times \left(x_i(t) - Lb^b\right) + C_2 \times \left(x_i(t) - Ub^b\right) \tag{19}$$

where, $x_i(t)$ represents the position information of the i-th foraging dung beetle during the t-th iteration, X^b denotes the global optimal position, while Lb^b and Ub^b signify the lower and upper limits of the foraging area of the dung beetle, respectively. Additionally, C_1 is a random number that follows a normal distribution, and C_2 is a random vector with values ranging between 0 and 1.

3.2.5 The Theft Behaviour

In the dung beetle population, certain individuals will pilfer food from those located near the globally optimal position X^b, and their position updates are calculated according to (20)

$$x_i(t+1) = X^b + S \times g \times \left(|x_i(t) - X^*| + \left|x_i(t) - X^b\right|\right) \tag{20}$$

where, $x_i(t)$ represents the position information of the i-th stealing dung beetle during the t-th iteration, g is a random vector of $1 \times D$ dimension that follows a normal distribution, and S is a constant value set at 0.5.

3.3 Trajectory Optimisation Method Based on NMSDBO Algorithm

This paper highlights specific limitations of the DBO algorithm, including suboptimal performance in ball rolling behaviour and a restricted ability to explore globally optimal positions during the later stages of execution. In response to these issues, we propose a new multi-strategy improved dung beetle optimisation algorithm. By incorporating multiple strategies, this enhanced version improves both the rolling behaviour inherent in dung beetle optimisation and its ability to identify global optimal positions, thereby increasing overall efficiency and accuracy of the algorithm.

3.3.1 The Golden Sine Strategy Integrated for Improving the Ball Rolling Behaviour

In the DBO algorithm, the ball rolling behaviour of dung beetles leads the entire population and plays a crucial role in enhancing both the global search capability and convergence speed of the algorithm. However, this rolling ball behaviour primarily explores optimal solutions through linear movement, which results in an incomplete exploration of the problem space and consequently low global search efficiency. To improve the global search ability of the algorithm, we introduce the golden sine strategy into this rolling behaviour. The Golden sine algorithm (Golden-SA), proposed by Tanyildizi *et al.* in 2017 [19], is a novel optimisation technique that employs a sine function combined with a golden section coefficient to enhance search performance. This approach demonstrates strong capabilities for global searching.

According to the research conducted by Tanyildizi *et al.* adjusting the amplitude and frequency of the sine wave enhances the algorithm's efficiency in exploring local regions of complex optimisation problems, thereby improving solution accuracy. Consequently, integrating the golden section with the sine function enables faster identification of optimal values while mitigating the risk of the algorithm converging to local optima. By substituting the individual update equation during the ball rolling behaviour phase with a golden sine strategy, significant optimisation can be achieved in updating individual positions within this context. The rolling ball behaviour described in (13) is replaced by the mathematical model based on golden sine as presented in (21)

$$\begin{aligned} x_i(t+1) = {} & x_i(t) \times |\sin(r_1)| - r_2 \times \sin(r_1) \times |c_1 \\ & \times X^b - c_2 \times x_i(t)| \end{aligned} \tag{21}$$

where, $x_i(t)$ represents the position of the i-th dung beetle in the population at the t-th iteration. The variable r_1 denotes the random number with a value of $0 \sim 2\pi$, the variable r_2 indicates another random number with a value of $0 \sim \pi$, and X^b signifies the global optimal position. Additionally, $c_1$and c_2 are referred to as golden sine coefficients, which are calculated as follows

$$c_1 = \pi \times (1-\tau) - \pi \times \tau, c_2 = \pi \times \tau - \pi \times (1-\tau) \quad (22)$$

where, $\tau = (\sqrt{5}-1)/2$ is the golden section coefficient.

By introducing the golden sine guidance mechanism, the population range is precisely defined, and the optimal search area is established. This mechanism directs the entire optimisation process and significantly enhances the algorithm's performance. Simultaneously, the search area is reduced according to the golden ratio, allowing for a rapid convergence of the population towards the optimal region, thereby accelerating convergence speed.

3.3.2 Adaptive T-Distribution Perturbation

The behaviours of rolling, dancing, foraging, and stealing in each iteration of the DBO algorithm are centered around the global optimal position. Consequently, this global optimal position plays a crucial role within the algorithm. However, the DBO algorithm does not fully leverage this global optimal position. Drawing inspiration from [20], we propose that a T-distribution disturbance factor can be employed to randomly perturb the global optimal position, thereby preventing the algorithm from converging prematurely to local optima. The distribution characteristics of the T-distribution disturbance factor are contingent upon its degrees of freedom n. Specifically, when the degrees of freedom $n = 1$, it resembles a Cauchy distribution which facilitates broader exploration; conversely, as it increases towards infinity, it approximates a Gaussian distribution that enhances local search capabilities [21]. By selecting an appropriate degree of freedom to construct the T-distribution operator and integrating it into the DBO algorithm, we can effectively utilise both Cauchy mutation and Gaussian mutation advantages.

To enhance the mutation probability of the global optimal individual and improve the algorithm's ability to escape from local optima during previous iterations, it is observed that the algorithm tends to concentrate around the optimal solution as the number of iterations increases. In this context, we introduce an adaptive T-distribution disturbance factor based on an exponential function, as proposed in [22]

$$T = T\left(\frac{1}{m} \times \exp\left(\frac{\ln\left(\frac{\mathrm{iter}_{\max}}{5}\right)}{\frac{\mathrm{iter}_{\max}}{5}}\right)^{\mathrm{iter}}\right) \quad (23)$$

where, m represents the scaling factor, which is set to 10, iter denotes the current number of iterations,and $\mathrm{iter}_{\max}$ is the maximum number of iterations.

For the globally optimal individual X^b, the perturbation derived from the adaptive T-distribution is as follows

$$x_{\mathrm{new}} = X^b + X^b \times T\left(\frac{1}{m} \times \exp\left(\frac{\ln\left(\frac{\mathrm{iter}_{\max}}{5}\right)}{\frac{\mathrm{iter}_{\max}}{5}}\right)^{\mathrm{iter}}\right) \quad (24)$$

where, x_{new} represents the individual a solution perturbed by an adaptive T-distribution. During the initial stages of iteration, the degrees of freedom for the adaptive T-distribution approach 0.1. At this point, there is a significant probability that variables drawn from the T-distribution will yield extreme values far from zero. This characteristic enhances the global search capability of the algorithm and helps prevent it from converging prematurely to local optimal solutions in these early iterations. As iterations progress into later stages, the degrees of freedom for the adaptive T-distribution tend toward infinity, resembling a Gaussian distribution. Consequently, perturbations applied to individuals become more moderate, which facilitates the algorithm's ability to identify local optimal solutions and improves its convergence speed during this phase.

Not every perturbation leads to improved outcomes. Consequently, superior individuals are preserved according to a greedy selection criterion. The specific methodology is outlined in (25)

$$X^b = \begin{cases} X_{\mathrm{new}}, f\left(X_{\mathrm{new}}\right) < f\left(X^b\right) \\ X^b, others \end{cases} \quad (25)$$

In the equation, $f\left(X_i\right)$ represents the fitness function. By comparing the fitness values before and after the disturbance, we can select the superior individual as the new global optimal individual.

As illustrated in Algorithm 1, on the basis of 3-5-3 polynomial trajectory planning, the NMSDBO algorithm is employed to compute the optimal time parameters, thereby enabling smoother operation of the manipulator and minimising movement duration. Initially, the trajectory time parameters for each joint of the manipulator are optimised individually. Subsequently, a comparison is made among the trajectory durations of each joint within the same motion phase, with the maximum value being selected as the time parameter for 3-5-3 polynomial trajectory planning. Ultimately, this process yields the shortest operational trajectory for the manipulator throughout its entire running cycle.

4. Simulation Experiment and Analysis

4.1 Performance Analysis of NMSDBO Algorithm

To assess the development capability, exploration capacity, and the ability to escape local optimal solutions of the NMSDBO algorithm, this paper selects six benchmark functions, which include both unimodal and multimodal functions. Specifically, F1 to F3 are unimodal functions characterised by a single local extremum, thereby demonstrating the global development capability of various optimisation algorithms. In contrast, F4 to F6 are multimodal functions featuring multiple local extrema; these serve to illustrate the exploration capacity of different optimisation algorithms as well as their ability to transcend local optimal solutions. A detailed description of these benchmark functions is provided in Table 2.

All simulation experiments are conducted in a consistent experimental environment. The hardware specifications include an AMD 5600 H CPU with a main frequency of 3.3 GHz, complemented by 16 GB of DDR4

Algorithm 1
The Proposed Time-optimal Trajectory Planning Algorithm

Input: Cartesian space interpolation points, population size M, number of iterations $T_{\max}$ Output: Optimal fitness value and global optimal time parameter t_{i1}, t_{i2}, and t_{i3}	
Step1	Four interpolation points in Cartesian space are inversed into six groups of interpolation points in joint space through the application of inverse kinematics.
Step2	According to the number of populations M and the number of iterations $T_{\max}$, the population position is randomly initialised.
Step3	According to (9), the joint coefficient matrix b under the time parameters of group M is solved, the joint angle, velocity, and acceleration of different interpolation stages are calculated according to the (1), (2), and (3).
Step4	The fitness value of the initial time parameter is calculated and the global optimal fitness value X^b is obtained.
Step5	When the dung beetle rolls the ball straight forward, the position is calculated according to (21). If an obstacle is encountered, the position is calculated according to (14).
Step6	According to the fitness value after the ball rolling behaviour, the local optimal position X^* is updated.
Step7	The positions of breeding, foraging, and stealing are calculated according to (17), (19), and (20).
Step8	The fitness values of all individuals are computed, from which the optimal fitness value and the local optimal time parameter are derived. Subsequently, the global optimal time parameter is preserved in accordance with the greedy rule.
Step9	According to the adaptive T-distribution perturbation strategy outlined in (24), the global optimal time parameter is updated.
Step10	Repeat the aforementioned steps until the iteration is complete, and subsequently output both the optimal fitness value and the globally optimal time parameter.

Table 2
Benchmark Functions

Function	Function Expressions	Value Ranges	Theoretical Value
F1	$f(x)=\sum_{i=1}^{n} x_i^2$	[-100,100]	0
F2	$f(x)=\sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	[-10,10]	0
F3	$f(x)=\sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	[-100,100]	0
F4	$f(x)=\sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$	[-500,500]	-12569.5
F5	$f(x)=\sum_{i=1}^{n} \left[x_i^2 - 10cos\left(2\pi x_i\right) + 10\right]$	[-5.12,5.12]	0
F6	$-20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right) + 20 + e$	[-32,32]	0

memory. The operating system utilised is Windows 11, and the testing software employed is MATLAB R2019b. Each simulation experiment is executed independently for a total of 30 runs on each test function. For comparative analysis, the WOA [23], Harris Hawks optimisation (HHO) [23], SSA [24], and DBO algorithms have been selected. The relevant parameters for these algorithms are detailed in Table 3.

The simulation results are presented in Table 4. Three indicators—optimal value (Best), mean value (Mean), and standard deviation (Std)—are employed to comprehensively assess the optimisation capability and stability of the algorithm. The optimal value reflects the best performance achieved by the algorithm during the optimisation process, while the mean value indicates its overall performance level. Additionally, the standard deviation provides further insight into the stability of algorithmic optimisation. As shown in Table 4, NMSDBO demonstrates superior optimisation capabilities. For unimodal functions F1 and F3, only NMSDBO successfully identifies the theoretical optimal value; both its mean and standard deviation align with this theoretical optimal value of 0, thereby underscoring its exceptional accuracy and stability. In contrast, for unimodal function F2, DBO exhibits a slight advantage over WOA and HHO; however, the improved NMSDBO shows a more pronounced effect with an accuracy enhancement ranging from 100 to 120 orders of magnitude. Regarding multimodal functions, NMSDBO also displays robust stability. In multimodal function F4, DBO's performance is poor; nevertheless, following enhancements made to it, NMSDBO's optimisation ability and stability have significantly improved. For multimodal

Table 3
Parameter Configuration of Different Algorithms

Algorithm	Parameter Configuration
WOA	Population size $M = 30$, maximum number of iterations $iter_{\max} = 500$, hunting behaviour probability $P_s = 0.6$, logarithmic spiral shape parameter $b = 1$.
HHO	Population size $M = 30$, the maximum number of iterations $iter_{\max} = 500$.
SSA	The population size $M = 30$, the maximum number of iterations $iter_{\max} = 500$, the number of discoverers $PD = 0.2 * n$, the number of alarms $SD = 0.2 * n$, and the alarm threshold $ST = 0.8$.
DBO	Population size $M = 30$, the maximum number of iterations $iter_{\max} = 500$.
NMSDBO	Population size $M = 30$, the maximum number of iterations $iter_{\max} = 500$.

Table 4
Test Results of Different Algorithms

Function	Parameter	WOA	HHO	SSA	DBO	NMSDBO
F1	Best	8.99E-84	1.59E-112	8.81E-161	2.66E-162	**0.00E+00**
	Mean	7.18E-73	2.04E-100	3.74E-50	5.90E-106	**0.00E+00**
	Std	2.92E-72	1.03E-99	2.05E-49	3.23E-105	**0.00E+00**
F2	Best	1.99E-57	4.31E-60	3.54E-176	2.30E-87	**3.76E-208**
	Mean	2.60E-52	5.03E-51	7.67E-31	3.28E-58	**4.37E-156**
	Std	7.55E-52	2.59E-50	3.65E-30	1.77E-57	**2.39E-155**
F3	Best	3.20E-58	8.19E-97	3.70E-201	3.16E-140	**0.00E+00**
	Mean	7.94E-51	1.42E-79	6.33E-23	3.04E-33	**0.00E+00**
	Std	4.08E-50	5.30E-79	3.47E-22	1.66E-32	**0.00E+00**
F4	Best	**-1.26E+04**	**-1.26E+04**	-9.98E+03	-1.23E+04	**-1.26E+04**
	Mean	-1.03E+04	**-1.26E+04**	-8.78E+03	-8.38E+03	-1.11E+04
	Std	1.70E+03	**4.88E-01**	5.93E+02	1.54E+03	1.60E+03
F5	Best	0.00E+00	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**
	Mean	3.79E-15	0.00E+00	0.00E+00	5.51E+00	**0.00E+00**
	Std	2.08E-14	0.00E+00	0.00E+00	3.02E+01	**0.00E+00**
F6	Best	8.88E-16	8.88E-16	8.88E-16	8.88E-16	**8.88E-16**
	Mean	4.44E-15	8.88E-16	8.88E-16	8.88E-16	**8.88E-16**
	Std	2.64E-15	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**

functions F5 and F6, differences among HHO, SSA, and NMSDBO are minimal; nonetheless, NMSDBO continues to maintain a dominant position. Compared to DBO, NMSDBO sustains a commendable degree of accuracy and stability when dealing with multi-class functions.

The convergence curve serves as a crucial metric for assessing the performance of optimisation algorithms. By analysing the convergence curve, we can directly observe the convergence speed, trend, and quality of the final solution when addressing specific optimisation problems. Figure 4 illustrates the convergence curves of five algorithms applied to six benchmark functions. As shown in Fig. 4(a), NMSDBO exhibits the fastest convergence speed, consistently decreasing to achieve the lowest objective function value throughout its iterations. The performance of DBO follows closely behind NMSDBO, while WOA, HHO, and SSA demonstrate similar performances with comparatively larger fitness values on their curves. From Fig. 4(b), it is evident that NMSDBO experiences rapid decline during its initial phase before steadily dropping to significantly lower fitness values than those achieved by the other four algorithms. In Fig. 4(c), we observe

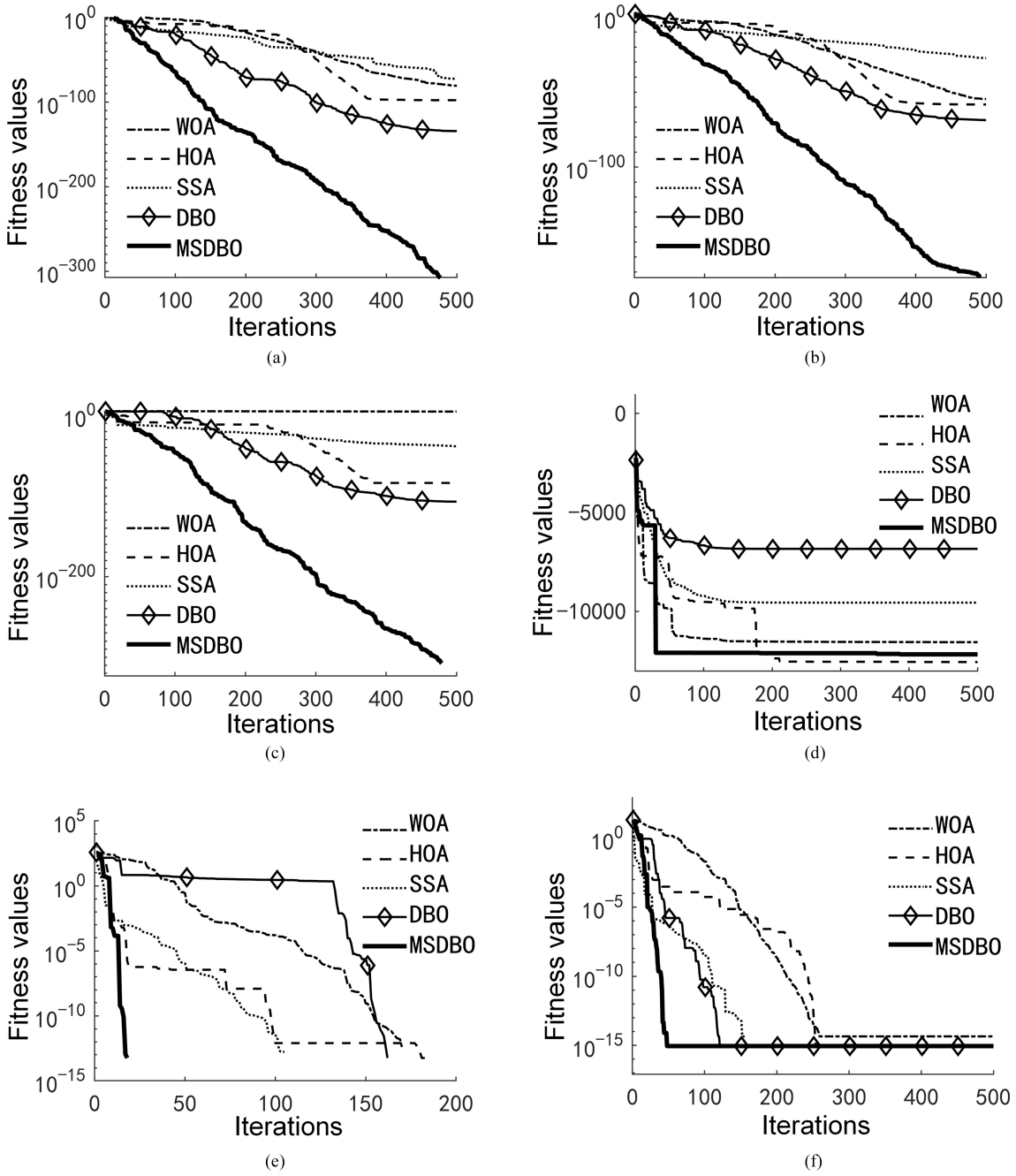

Figure 4. The convergence curves of six benchmark functions tested by different algorithms: (a) F1 convergence curve; (b) F2 convergence curve; (c) F3 convergence curve; (d) F4 convergence curve; (e) F5 convergence curve; and (f) F6 convergence curve.

that NMSDBO not only performs well but also converges quickly; its objective function value is substantially lower than those produced by other algorithms. Although DBO shows a rapid decrease at certain points during its mid-iterations, both early and late stages exhibit slower rates of decline. Figure 4(d) reveals the weak global search ability of the DBO algorithm. Although the NMSDBO encountered an iterative bottleneck period around the 30th generation, it soon resumed falling again and jumped out of the local optimal solution, demonstrating a strong global optimal solution search ability. As illustrated in Fig. 4(e), NMSDBO successfully identifies global optimal solutions; moreover, when compared with other algorithms capable of achieving theoretical optimality as well, it does so at an accelerated pace. Finally, Fig. 4(f) confirms that NMSDBO not only has superior convergence speed but also excels in rapidly locating final objective function values. In summary, when compared to the other four algorithms, the NMSDBO exhibits a superior convergence rate and enhanced global search capability.

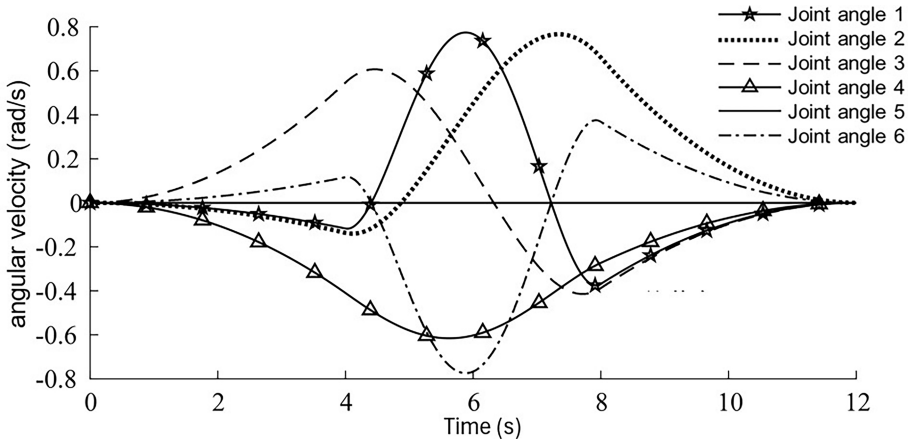

Figure 5. The angular velocity curve of each joint before optimisation.

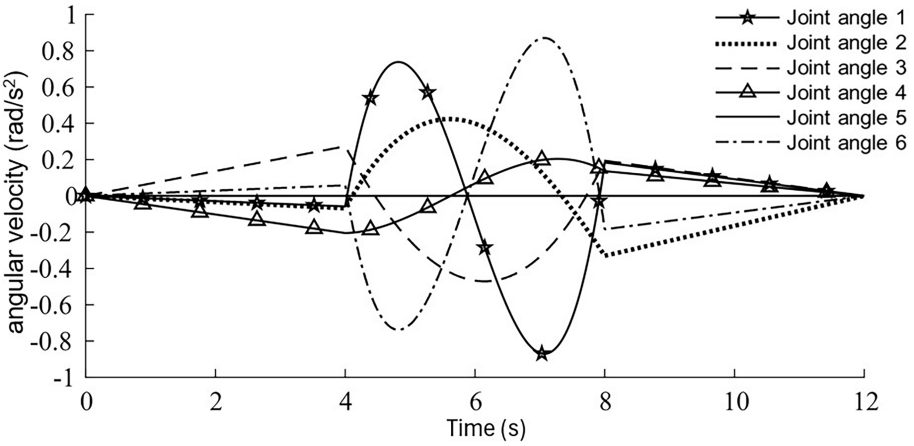

Figure 6. The angular acceleration curve of each joint before optimisation.

Table 5
Joint Space Position Points of Manipulator

Joint	X_0	X_1	X_2	X_3
1	-2.7182	-2.8735	-1.7550	-2.2460
2	-0.9328	-1.1149	0.4071	1.2877
3	1.5064	2.2350	2.7263	2.2095
4	0.9971	0.4507	-1.5626	-1.9264
5	-1.5708	-1.5708	-1.5708	-1.5708
6	1.1474	1.3027	0.1842	0.6752

4.2 Performance Analysis of Time-optimal Trajectory Planning Algorithm

We selected two target points of the end effector of the manipulator in the Cartesian coordinate system, along with two transitional points during the motion process. Utilising inverse kinematics, the joint angles corresponding to the four interpolation points listed in Table 5 within joint space are obtained.

In order to verify the efficiency of the proposed NMSDBO algorithm in reducing the running time of the manipulator, a comparison is made with the DBO algorithm. The relevant parameters for this algorithm are set as follows: population size $M = 30$, the interval between adjacent path points is 4 s, the maximum number of iterations is 500, and the constraints of each joint are presented in Table 6. Each method was executed ten times, and the time parameters for the 3-5-3 polynomial interpolation trajectory of each joint were recorded.

Under the constraint conditions of each joint, the optimal interpolation time identified by the two algorithms for each joint are presented in Table 7. When comparing the optimal interpolation time provided by different algorithms, it is essential to ensure that the motion of each joint adheres to time synchronisation requirements. This is necessary because the motors of all joints must operate cohesively during actual movement. Consequently, a comparison is made regarding the motion time of each joint within the same trajectory planning segment as shown in Table 7, and the longest motion time among all joints in each trajectory is selected as the optimal interpolation time.

Table 8 records the time parameters for the 3-5-3 polynomial trajectory planning optimised by various

Table 6
Constraint Conditions of Each Joint

Constraint Condition	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
$\lvert v_{\max}\rvert\,(\mathrm{rad}/s)$	3.05	3.05	3.05	6.28	6.28	8.73
$\lvert a_{\max}\rvert\,(\mathrm{rad}/s^2)$	3.05	3.05	3.05	6.28	6.28	8.73

Table 7
The Interpolation Time of Each Joint after NMSDBO Optimisation

Joint Number	DBO			NMSDBO		
	$t_1\,(s)$	$t_2\,(s)$	$t_3\,(s)$	$t_1\,(s)$	$t_2\,(s)$	$t_3\,(s)$
1	0.7855	2.9164	1.5995	0.7702	2.7831	1.5569
2	0.9420	1.7876	1.4113	0.9421	1.7886	1.4102
3	1.4985	1.7640	1.4488	1.2148	1.7493	1.4359
4	0.7225	0.6930	0.5906	0.7225	0.6930	0.5906
5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.4557	1.6702	0.9323	0.4578	1.6712	0.9291

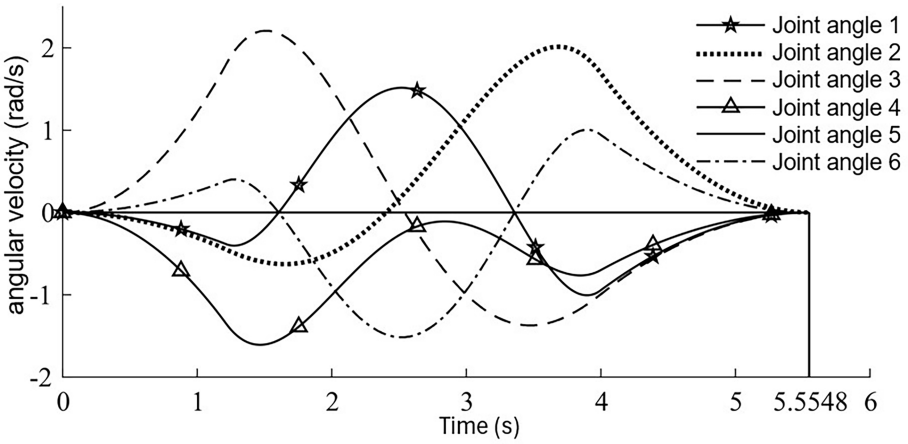

Figure 7. The velocity curve of each joint after optimisation.

algorithms, along with the total duration of the trajectory planning process. The experimental results indicate that by introducing NMSDBO algorithm, the time parameters of the 3-5-3 polynomial trajectory planning is successfully optimised. Following optimisation, the time required for each joint to complete its action has been reduced from an initial preset of 12 sto 5.5548 s. This represents a reduction of 7.64% compared to the DBO algorithm's duration of 6.0144 s.

The trajectory curves utilising the preset 3-5-3 polynomial trajectory planning are illustrated in Figs. 5 and 6, it is evident that during the manipulator's motion planning, both the angular velocity and angular acceleration of each joint exhibit continuity and smoothness. However, it is noteworthy that their peak values fall significantly below the preset velocity and acceleration constraints.

Table 8
The Interpolation Time Optimisation Results of Each Joint of Different Algorithms

Algorithm	$t_1\,(s)$	$t_2\,(s)$	$t_3\,(s)$	Total Duration (s)
3-5-3	4	4	4	12
DBO	1.4985	2.9164	1.5995	6.0144
NMSDBO	1.2148	2.7831	1.5569	5.5548

This observation indicates considerable potential for optimising the manipulator's motion performance. By adjusting the parameters of the planning algorithm, enhancements can be made to both speed and acceleration,

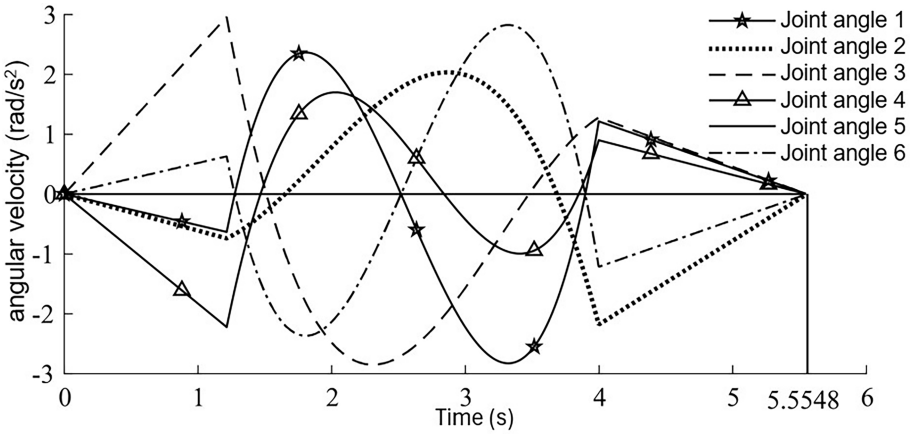

Figure 8. The acceleration curve of each joint after optimisation.

thereby reducing the time required to complete tasks effectively.

Substituting the optimised interpolation time $t_1 = 1.2148s$, $t_2 = 2.7831s$, $t_3 = 1.5569s$ into the 3-5-3 polynomial trajectory planning allows for the determination of changes in angular velocity and angular acceleration for each joint, as illustrated in Figs. 7 and 8. The diagrams clearly demonstrate that the time-optimal trajectory planning method based on the NMSDBO algorithm proposed in this paper not only adheres to the predefined speed and acceleration constraints but also significantly enhances both speed and acceleration across all joints. Furthermore, it is evident from the graphs that the velocity and acceleration curves for each joint over time are smooth and continuous. This characteristic ensures fluid motion during operation, thereby mitigating mechanical wear associated with abrupt changes in acceleration. Additionally, a smooth motion trajectory contributes to greater stability of the manipulator during its operations, which is particularly crucial when executing precision operations.

5. Conclusion

In this paper, we present a 3-5-3 piecewise polynomial interpolation trajectory planning method specifically designed for the IRB2600 manipulator. To reduce the time parameters associated with the trajectory planning, we propose a trajectory optimisation approach based on NMSDBO algorithm. By incorporating penalty functions, the constrained optimisation problem is transformed into an unconstrained optimisation framework. The integration of the golden sine strategy, along with the introduction of adaptive T-distribution perturbations, enhances both the rolling behaviour of DBO algorithm and the evolution of its global optimal position. Through comparative analysis of experimental performance against WOA, HHO, SSA, and DBO algorithms, it is demonstrated that the NMSDBO algorithm exhibits superior accuracy, rapid convergence speed, and exceptional global search capabilities. The simulation experiments focused on time-optimal trajectory planning utilising NMSDBO indicate that this method can significantly decrease the running time of the manipulator while ensuring smoothness and continuity in its movements. This aligns more closely with modern large-scale industrial production requirements for efficient, safe, and stable robotic operations. Future research should concentrate on investigating methodologies for the simultaneous optimisation of multiple indicators, and complex environments with obstacles, thereby offering more comprehensive and efficient solutions for the trajectory optimisation of robotic arms.

References

[1] J.Q. Cao, and X.S. Han, Review of research methods for industrial robot trajectory planning, *Information and Control*, *53*(4), 2024, 471-486.

[2] Z. Wang, Prediction and analysis of robotic arm trajectory based on adaptive control, *Mechatronic Systems and Control*, *51*(10), 2023, 1-9.

[3] M. Jasim Mohamed, B.K. Oleiwi, A.T. Azar, and A.R. Mahlous, Hybrid controller with neural network PID/FOPID operations for two-link rigid robot manipulator based on the zebra optimization algorithm, *Frontiers in robotics and AI*, *11*, 2024, 1386968.

[4] M.J. Mohamed, B.K. Oleiwi, A.T. Azar, and I.A. Hameed, Coot optimization algorithm-tuned neural network-enhanced PID controllers for robust trajectory tracking of three-link rigid robot manipulator, *Heliyon*, *10*(13), 2024, e32661.

[5] C.G. He, R.Z. Zhu, T.B. Huang, X.X. Wang, Z.D. Huang, and J.H. Liu, Research progress on trajectory planning and optimization of industrial robots, *Modern Manufacturing Engineering*, *7*, 2023, 150–159.

[6] A. Shrivastava, exploring optimal motion strategies: A comprehensive study of various trajectory planning schemes for trajectory selection of robotic manipulator, *Journal of The Institution of Engineers (India): Series C*, *106*(2), 2025, 691-710.

[7] Z.Y. Fan, L.J. Li, Y.H. Li, H. Lv, X.H Fu, and J. Li, Trajectory planning of camellia fruit picking manipulator based on improved grey wolf algorithm, *Mechanical Design and Research*, *38*(1), 2022, 195–201.

[8] X.F. Zhang, F. Xiao, X.L. Tong, J.T. Yun, Y. Liu, Y. Sun, B. Tao, J.Y. Kong, M.M. Xu, and B.J. Chen, Time optimal trajectory planning based on improved sparrow search algorithm, *Frontiers in Bioengineering and Biotechnology*, *10*, 2022, 1–14.

[9] J. Zhao, X. Zhu, and T. Song, Serial manipulator time-jerk optimal trajectory planning based on hybrid IWOA-PSO algorithm, *IEEE Access*, *10*, 2022, 6592-6604.

[10] H. Zhang, A new hybrid whale particle swarm optmisation algorithm for robot trajectory planning and tracking control, *Mechatronic Systems and Control*, *52*(1), 2024, 48-57.
[11] M.Y. Zhou, M.W. Zhou, G.Z. Liu, and C. Cheng, Time optimal trajectory planning of manipulator based on improved butterfly algorithm, *Computer Science*, *50*(S2), 2023, 119-126.
[12] M.Z. Pan, L. Pan, C.L. Li, K. Liang, and B.W. Yao, Trajectory planning of manipulator based on improved sine cosine algorithm, *Modular Machine Tool and Automatic Machining Technology*, *1*, 2024, 43–46.
[13] B.K. Patle, S.-L. Chen, A. Singh, and S.K. Kashyap, Optimal trajectory planning of the industrial robot using hybrid S-curve-PSO approach, *Robotic Intelligence and Automation*, *43*(2), 2023, 153-174.
[14] J.K. Xue, and B. Shen, Dung beetle optimizer: A new meta-heuristic algorithm for global optimization, *The Journal of Supercomputing*, *79*(7), 2023, 7305-7336.
[15] J.C. Pan, S.B. Li, P. Zhou, G.L. Yang, and D.C. Lv, Dung beetle optimization algorithm guided by improved sine algorithm, *Computer Engineering and Applications*, *59*(22), 2023, 92–110.
[16] Q. Guo, and Q. Zheng, Multi-strategy improved dung beetle optimizer and its application, *Journal of Frontiers of Computer Science Technology*, *18*(4), 2024, 930-946.
[17] Y. Li, K. Sun, Q. Yao, and L. Wang, A dual-optimization wind speed forecasting model based on deep learning and improved dung beetle optimization algorithm, *Energy*, *286*, 2024, 129604.
[18] C. Ban, G.Y. Ren, B.R Wang, X.J. Chen, Z. Xue, and L. Wang, Kinematic calibration of industrial robots with weighted SVD algorithm, *Journal of Metrology*, *42*(9), 2021, 1128–1135
[19] E. Tanyildizi, and G. Demir, Golden sine algorithm: a novel math-inspired algorithm, *Advances in Electrical Computer Engineering*, *17*(2), 2017, 71-78.
[20] Z. Tian, and J. Wang, Variable frequency wind speed trend prediction system based on combined neural network and improved multi-objective optimization algorithm, *Energy*, *254*, 2022, 124249.
[21] J. Liu, and Z. Wang, A hybrid sparrow search algorithm based on constructing similarity, *IEEE Access*, *9*, 2021, 117581-117595.
[22] R. Wu, H. Huang, J. Wei, C. Ma, Y. Zhu, Y. Chen, and Q. Fan, An improved sparrow search algorithm based on quantum computations and multi-strategy enhancement, *Expert Systems with Applications*, *215*, 2023, 119421.
[23] M.H. Nadimi-Shahraki, H. Zamani, Z. Asghari Varzaneh, and S. Mirjalili, A systematic review of the whale optimization algorithm: Theoretical foundation, improvements, and hybridizations, *Archives of computational methods in engineering : state of the art reviews*, 2023, 1-47.
[24] M. Shehab, I. Mashal, Z. Momani, M.K.Y. Shambour, A. AL-Badareen, S. Al-Dabet, N. Bataina, A.R. Alsoud, and L. Abualigah, Harris Hawks optimization algorithm: Variants and applications, *Archives of Computational Methods in Engineering*, *29*(7), 2022, 5579-5603.
[25] J. Xue, and B. Shen, A novel swarm intelligence optimization approach: Sparrow search algorithm, *Systems Science & Control Engineering*, *8*(1), 2020, 22-34.

Biographies

Bo Xue received the B.S. degree in electronic information engineering and the M.S. degree in signal and information processing from Yangzhou University, Yangzhou, China, in 2003 and 2006, respectively, and the Ph.D. degree in signal and information processing from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2018. Currently, he serves as an Associate Professor with the Department of Electric Information Engineering, Jiangsu University of Technology, Changzhou, China. His research interests focus on statistical signal processing and machine learning for wireless sensor networks.

Mengcheng Lin received the B.S. degree in electronic information engineering from Hangzhou Normal University Qianjiang College, Hangzhou, China, in 2021, and the M.S. degree in mechanical engineering from Jiangsu University of Technology, Changzhou, China, in 2024. Currently, he serves as a Software Engineer with the Delixi Electric Ltd., Hangzhou, China. His research interests focus on robot control technology.

Xiangyu Wu received the B.S. degree in communication engineering from Nanchang University of Transportation, Nanchang, China, in 2022. He is currently pursuing the master's degree in mechanical engineering with Jiangsu University of Technology, Changzhou, China. His research field is the mechanical and electrical product testing and intelligent control.