Hindawi
Journal of Robotics
Volume 2020, Article ID 6859589, 13 pages
https://doi.org/10.1155/2020/6859589

Research Article

A Dual-Thread Method for Time-Optimal Trajectory Planning in Joint Space Based on Improved NGA

Kaipeng Zhang,[1,2] **Ning Liu**,[1,2] **and Gao Wang**[1,2]

[1]*College of Information Science and Technology, Jinan University, Guangzhou 510632, China*
[2]*Robotics Research Institute of Jinan University, Guangzhou 510632, China*

Correspondence should be addressed to Kaipeng Zhang; cheungk.roc@gmail.com and Ning Liu; tliuning@jnu.edu.cn

Received 1 December 2019; Accepted 16 January 2020; Published 15 February 2020

Academic Editor: Keigo Watanabe

Copyright © 2020 Kaipeng Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

To solve the problem that the time-consuming optimization process of Genetic Algorithm (GA) can erode the expected time-saving brought by the algorithm, time-optimal trajectory planning based on cubic spline was used, after the modification to classical fitness sharing function of NGA, a dual-threaded method utilizing elite strategy characteristic was designed which was based on Niche Genetic Algorithm (NGA) with the fitness sharing technique. The simulation results show that the proposed method can mitigate the contradiction of the long term the optimization algorithm takes but a short running time the trajectory gets, demonstrating the effectiveness of the proposed method. Besides, the improved fitness sharing technique has reduced the subjective process of determining relevant parameters and the optimized trajectory results met performance constraints of the robot joints.

1. Introduction

Genetic Algorithm (GA), capable of avoiding local minimum, independent of gradient information, is extremely suitable for large-scale complex optimization problem that traditional methods found hard to model and solve. In the actual industrial application field such as package-palletizing and workpiece-welding, a large amount of GA variant has been developed to help obtain the industrial manipulators' performance-optimal trajectory in some kinds of specification.

Dong et al. [1] and Chen [2] applied GA to path optimization for palletizing manipulators, improving manipulators' efficiency. In this scenario, the palletizing work of manipulators can be classified as a pick-and-place job. Liu et al. [3] used vision-based industrial robot improvement GA, planning the trajectory of pig abdominal cutting. Hou et al. [4] used a GA based multiobjective optimization algorithm to study the workpiece welding problem of a two-robot cooperative system. An improved multi-island migration GA was utilized by Du et al. [5] to plan the falling motion of a crossbeam in a hydraulic press system, so as to obtain the trajectory with strong pressure shock suppression ability. Ren et al. [6] used a GA to solve the motion planning problem of asynchronous parallel disassembly (aPDP) in regenerative manufacturing, which is a promising technology compared with serial disassembly.

Most of the current research studies are devoted to accelerating the local convergence of GA and extending its ability to search on the global solution space. Two main categories can achieve the target, first of which would be combining GA with another optimization algorithm. Since GA has the global optimization ability, quite a few methods utilized an algorithm with prevailing in local search to combine. Zhang et al. [7] combined whale optimization algorithm (WOA) with GA by adding WOA operations to solve the dynamic modeling problem of manipulators. A similar method was also proposed by Zhou et al. [8], in which grey wolf optimization (GWO) was utilized, and grey wolf hunting behaviors were introduced before GA's operations. The proposed hybrid GA with GWO can provide input weights and biases of an extreme learning machine. The quasi-Newton solver is a classical method with fast local convergence, and it was combined with GA by Khawli [9] to

complete the calibration job of a manipulator with the help of a laser tracker. In [10], independent evolution and atavistic migration mechanism with duo populations have been introduced to GA, in which the Chaos Search Algorithm (CSA) was utilized to provide newly born individuals to populations periodically. There are also hybrid GA algorithms regarding the premature defects of GA as the advantage of fast convergence. In [11], the CSA was used to traverse the entire solution space with the purpose of narrowing the possible scope where the optimal solution would appear, while a GA with premature defect completed the local search. The second category would be hybrid algorithms that get heuristic from bioecology or other academic disciplines to improve GA. The muti-island migration model of IMGA in [5] is derived from the theory of natural migration, in which populations are isolated by imaginary islands and reproduce independently. After a few generations, individuals migrate between islands, guaranteeing genetic diversity. Liu et al. [12] adopted a concept of elitist groups to overcome the premature defect and enhance the global searching capacity of GA by replacing the worse group with the most elitist group. The quantum genetic algorithm (QGA) was used by Chen and Zhou [13] to perform trajectory planning for a space manipulator with a floating base. It borrows the concept from quantum theory and encodes GA's chromosome with quantum states, while population propagation is accomplished by a quantum rotate gate, enriching GA's genetic diversity.

The purpose of optimal time optimization is to shorten the trajectory runtime, improving efficiency and productivity. However, the heavy calculation burden and time-consuming optimization process of GA has a negative effect on the efficiency, which will erode the results of the algorithm. Research studies on how to improve the practicability of GA in practical implementation are seldom mentioned. When using GA to plan an optimal trajectory for manipulators, the conventional method always arranges the optimization algorithm and manipulators' execution algorithm into a sequential structure. More specifically speaking, the optimization results will not be returned unless the GA termination conditions were satisfied, after which manipulators continue executing the optimal trajectory. In reality, however, the trajectory may be accomplished in less than a few seconds, but optimizing it often takes multiple times as long as its original runtime. If the trajectory only needs to be executed once, it may not be worth optimizing it, especially in those typical manufacture scenarios. Even for repetitive trajectories, it is necessary to consider whether the improvement brought by the investment in optimization time is cost-effective. In the case of inconsiderable improvement, it is also futile to invest time resources to optimize the time-optimal trajectory. Therefore, the problem of whether the termination condition of the algorithm is reasonable and the granularity of the traversal domain is appropriate under the conventional scheme is particularly prominent. The negative result of the coarse design of the termination conditions is that the optimization process time will negate or even devour the ideal advantage of trajectory optimization.

In this paper, the Niche Genetic Algorithm (NGA) with elite strategy and modified fitness sharing function of NGA was adopted to construct a dual-thread algorithm concerning parallel processing of the optimal trajectory optimization task and the real-time execution task of robot joint space. It is proposed to solve the problem that the coarse design of the termination conditions and the granularity of the traversal domain can erode the expected time-saving when GA is used routinely in the situation that the trajectory running time relative to the optimization process is short-lived.

2. Time-Optimal Trajectory Planning Based on the Cubic Spline

Generally speaking, to reduce the impulse shock to robot joints, ensuring the end effector of robots moving smoothly, a higher-order smooth function should be chosen as the interpolation function, which is devoted to calculate the interpolating trajectory between given points. Attention should be paid while designing the optimal purpose. The optimal trajectory runtime was chosen in our research since efficiency is always the first index to industrial robots. Last but not the least, proper simulation results can help to choose robot joint actuators economically. For these reasons, the joint position commands sent to the servo system carrying time information, which were always obtained through inverse kinematics calculation and interpolation, should try to exploit the potential of the driven system in the field of velocity, acceleration, and jerk limitation but also in the range of joint position bounds. In such considerations, the trajectory is optimized to the maximum extent concerning the upper limit performance of the manipulator body.

The cubic spline was chosen as the core interpolation function since it has the property of C^2 conductivity, ensuring continuous acceleration, and it can be wrapped into a matrix form exactly suitable for optimization using a genetic algorithm. If higher-order continuity and smoother trajectory were pursuing, the B-spline function could be another competitive candidate.

2.1. Matrix Form of Joint Space Trajectory Using the Cubic Spline. The mathematic solution of joint space trajectory planning using the cubic spline can be derived from the 3rd-order polynomial trajectory planning method [14].

Consider the three continuity conditions that the intermediate point should have: $q(t)$, $v(t)$, and $a(t)$ are continuous along the trajectory. Use the continuity of velocity to write linear equations in a matrix form, as shown in equation (1). By substituting joint position vector $q\vec{V}$ representing all path points whose length is n, along with time interval vector $\vec{h}$ between path points, initial and final velocity v_s, v_e, and also initial and final acceleration aV_s and aV_e, consequently, the acceleration vector $a\vec{V}$ at the spline junction meeting the boundary conditions can be calculated.

The coefficient matrix **A** and the conditions placed at the right hand side with vector form **b** of the formula in equation (1) were, respectively, represented as equations (2) and (3).

Based on the fact that the location continuity constraint is not clearly defined at the beginning and the end of the trajectory, 2 virtual points should be introduced: one at the beginning and the other at the end. After adding these two virtual points, the number of elements of the position vector $q\overrightarrow{V}$ comes up to $n+2$.

The actual positions of these two virtual points are not explicitly specified but are rather determined by other current information based on continuity conditions, as shown in the formula in equation (4). The cubic spline curve ensures the continuous requirement of joint space motion in the field of position, velocity, and acceleration, which can effectively reduce joint stiffness and flexibility impact.

$$\mathbf{A}\left[aV_2, aV_3, \ldots, aV_{n+1}\right]^T = \mathbf{b}, \tag{1}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & \ldots & & & \ldots & 0 \\ a_{21} & a_{22} & a_{23} & 0 & \ldots & & \ldots & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & \ldots & \ldots & 0 \\ \vdots & 0 & a_{43} & a_{44} & a_{45} & 0 & \ldots & 0 \\ & & & \ddots & \ddots & \ddots & & \\ 0 & \ldots & & \ldots & 0 & a_{n-3,n-4} & a_{n-3,n-3} & a_{n-3,n-2} \\ 0 & \ldots & & & \ldots & 0 & a_{n-2,n-3} & a_{n-2,n-2} \end{bmatrix},$$

$$\text{where, } \begin{cases} a_{11} = \dfrac{h_1}{2} + \dfrac{h_2}{3} + \dfrac{h_1^2}{6h_2}, a_{21} = \dfrac{h_2}{6} - \dfrac{h_1^2}{6h_2}, \\ a_{i,i+1} = \dfrac{h_i}{6}, a_{i,i} = \dfrac{h_i + h_{i+1}}{3}, a_{i+1,i} = \dfrac{h_{i+1}}{6}, \\ a_{n-3,n-2} = \dfrac{h_{n-2}}{6} - \dfrac{h_{n-1}^2}{6h_{n-2}}, \\ a_{n-2,n-2} = \dfrac{h_{n-1}}{2} + \dfrac{h_{n-1}}{3} + \dfrac{h_{n-1}^2}{6h_{n-2}}, \qquad \text{for } i = 2, 3, \ldots, n-3, \end{cases} \tag{2}$$

$$\mathbf{b} = b_1, b_2, \ldots, b_{n+1} b_{n+2}^T,$$

$$\text{where, } \begin{cases} b_1 = \dfrac{qV_3}{h_2} + \dfrac{qV_1}{h_1} - \left(\dfrac{1}{h_1} + \dfrac{1}{h_2}\right)\left(qV_1 + h_1 V_s + \dfrac{h_1^2}{3} aV_1\right) - \dfrac{h_1 aV_1}{6}, \\ b_2 = \dfrac{1}{h_2}\left(qV_1 + h_1 V_s + \dfrac{h_1^2 aV_1}{3}\right) + \dfrac{qV_4}{h_3} - \left(\dfrac{1}{h_2} + \dfrac{1}{h_3}\right) qV_3, \\ b_i = \dfrac{qV_{i+2} - qV_{i+1}}{h_{i+1}} - \dfrac{qV_{i+1} - qV_i}{h_i}, \\ b_{n+1} = \dfrac{1}{h_{n-2}}\left(qV_n - h_{n-1} V_e + \dfrac{h_{n-1}^2 aV_n}{3}\right) + \dfrac{1}{h_{n-3}} qV_{n-3} - \left(\dfrac{1}{h_{n-2}} + \dfrac{1}{h_{n-3}}\right) qV_{n-2}, \\ b_{n+2} = \dfrac{qV_n}{h_{n-1}} + \dfrac{qV_{n-2}}{h_{n-2}} - \left(\dfrac{1}{h_{n-1}} + \dfrac{1}{h_{n-2}}\right)\left(qV_n - h_{n-1} V_e + \dfrac{h_{n-1}^2}{3} aV_n\right) - \dfrac{h_{n-1} aV_n}{6}, \qquad \text{for } i = 3, 4, \ldots, n, \end{cases} \tag{3}$$

$$\begin{cases} qV_2 = qV_1 + h_1 v_s + \dfrac{h_1^2}{3}\left(aV_s + \dfrac{aV_2}{2}\right), \\ qV_{n+1} = qV_{n+2} - h_{n+1} v_e + \dfrac{h_{n+1}^2}{3}\left(aV_e + \dfrac{aV_{n+1}}{2}\right). \end{cases} \tag{4}$$

Once all the accelerations are determined using equation (1), they can be substituted into the cubic spline interpolation formula (equation (5)). According to the predetermined control period, the input time variable t can be used to calculate the joint position and velocity that need to be periodically output to the servo system for execution.

$$\begin{cases} \text{Position: } Q_i(t) = \dfrac{(tV_{i+1}-t)^3}{6(tV_{i+1}-tV_i)}aV_i + \dfrac{(t-tV_{i+1})^3}{6(tV_{i+1}-tV_i)}aV_{i+1} \\ + \left[\dfrac{aV_{k+1}}{tV_{i+1}-tV_i} - \dfrac{(tV_{i+1}-tV_i)aV(i+1)}{6}\right](t-tV_i) \\ + \dfrac{tV_{i+1}-tV_i}{6}[aV_i - aV_{i+1}], \qquad \text{where} \quad tV_i = \sum_{j=1}^{i} h_j. \\ \text{Velocity: } \dot{Q}_i(t) = \dfrac{-(tV_{i+1}-t)^2}{2(tV_{i+1}-tV_i)}aV_i + \dfrac{(t-tV_{i+1})^2}{2(tV_{i+1}-tV_i)}aV_{i+1} \\ + \dfrac{qV_{i+1}-qV_i}{tV_{i+1}-tV_i} + \dfrac{tV_{i+1}-tV_i}{6}[aV_i - aV_{i+1}]. \end{cases} \tag{5}$$

2.2. Expression of Time-Optimal Trajectory Goal and Constrains. From the point of view of productivity, the joint space constraint condition of the robot was set, which was guided by the motor and reducer to give a full play to their potential. The optimal trajectory running time is considered as the optimization objective, and the natural language expression of its constraint conditions is the corresponding velocity, acceleration, and jerk of each joint at every trajectory point do not exceed the upper limit of joint performance. The optimization target and constrains were shown in equation (6), where i is the index of time intervals, while j is the index of joints; moreover, $jC_j aC_j$, and vC_j, respectively, stand for the upper limit of the jth joint in the field of jerk acceleration and velocity. According to [15], the maximum of velocity acceleration and jerk along the trajectory can be calculated, which is also shown as follows:

$$\min T \quad \sum_{i=1}^{n+1} h_i,$$

$$\text{s.t.} \quad \begin{cases} \max\left|\dddot{Q}_{j,i}(t)\right| = \max\left\{\left|\dfrac{aV_{j,i+1}-aV_{j.i}}{h_i}\right|\right\} \le jC_j, \\ \max\left|\ddot{Q}_{j,i}(t)\right| = \max\left\{\left|aV_{j,i}\right|\right\} \le aC_j, \\ \max\left|\dot{Q}_{j,i}(t)\right| = \begin{cases} \max\left\{\left|\dot{Q}_{j,i}(t_i)\right|, \left|\dot{Q}_{j,i}(t_{i+1})\right|, \left|\dot{Q}_{j,i}(\overline{t_i})\right|\right\}, & \text{sgn}\left(aV_{j,i}\right) \ne \text{sgn}\left(aV_{j,i+1}\right) \\ \max\left\{\left|\dot{Q}_{j,i}(t_i)\right|, \left|\dot{Q}_{j,i}(t_{i+1})\right|\right\}, & \text{sgn}\left(aV_{j,i}\right) = \text{sgn}\left(aV_{j,i+1}\right) \end{cases} \le vC_j, \end{cases} \tag{6}$$

$$\text{for} \quad j = 1, 2, \ldots N;\ i = 1, 2, \ldots n+1,$$

$$\text{where,} \quad \begin{cases} \left|\dot{Q}_{j,i}(t_i)\right| = \left|\dfrac{-aV_i}{2}h_i + \tau\right|, \\ \left|\dot{Q}_{j,i}(t_{i+1})\right| = \left|\dfrac{aV_{i+1}}{2}h_i + \tau\right|, \\ \left|\dot{Q}_{j,i}(\overline{t_i})\right| = \dfrac{aV_i aV_{i+1}}{2(aV_{i+1}-aV_i)}h_i + \tau, \end{cases}$$

$$\tau = \frac{qV_{i+1}-qV_i}{h_i} + \frac{h_i}{6}(aV_i - aV_{i+1}).$$

3. Modification of NGA

Niche is a concept borrowed from the biology field, meaning those habitats and behaviors acclimatized to species. Genetic algorithm with niche behavior can be regarded as Niche Genetic Algorithm (NGA). For now, NGA has the ability to effectively increase differences among individuals, enriching the diversity of population, which has ensured genes from elites will not govern the whole population.

Many methods can be exerted to swift the standard genetic algorithm into a niche genetic algorithm, the first of which is the Fitness Sharing technique [16]. The fitness sharing technique is a common solution solving Pareto Front in a multiobjective optimization problem. The second common technique is Direct Crowing [17]. It would use some metric to divide the whole population into several small populations. Each small population individually produces its offspring, and resources competition would happen between offsprings and parents. Only the winner can be reserved. Different from these two gentle techniques mentioned before, Individuals Clearing [18] is a much more radical technique. It only reserves the best individual in the same niche population, in some perspective, meaning radical evolving.

3.1. Modification to Fitness Sharing Function. In this paper, the NGA algorithm with an elite strategy was implemented, in which niche behavior was developed from the classical fitness sharing technology. Fitness function was used to estimate niche crowding quantitatively. The classical fitness function is shown in equation (7), where $d_{i,j}$ is the distance (chromosome difference) between the ith individual and the jth individual; moreover σ_s is the radius of the niche, which is typically related to the maximum distance between individuals in the population, and the classical value of α is 1, which is usually used to control the steepness of the fitness function. The resource allocation manner of the classical fitness function is linear for different individuals in the same niche. This manner would lead to an underestimation of the crowding level between individuals with less chromosome difference, but an overestimation of those individuals with larger chromosome difference.

$$sh\left(d_{i,j}\right) = \begin{cases} 1 - \left(\frac{d_{i,j}}{\sigma_s}\right)^{\alpha}, & \text{if } d \le \sigma_s, \\ 0, & \text{if } d > \sigma_s. \end{cases} \tag{7}$$

Introducing niche behavior to GA can significantly reduce the number of similar individuals in the population, enriching the diversity of the gene pool in the population. For the two individuals with a small d value, it is expected to have a severe crowding degree estimation, to impose a higher penalty degree on the suboptimal individuals and reduce the probability of homogeneous genes being inherited to the next generation. Similarly, the two individuals with an immense d value are expected to have a moderately small crowding estimation. The general practice to achieve this purpose is adjusting the α value [19]. A sharper fitness function can be obtained by increasing α. However, there remained some extra considerations in determining α, and the optimal value is not easy to determine.

As shown in Figure 1(a), the classical fitness function and the standard normal distribution curve have a very high similarity of contour. Based on the fact that normal distribution has been widely used in simulating natural systems, this paper proposes to replace the classical fitness function with modified normal distribution function to reduce the subjective process of quantitative estimation of crowding degree. $sh_m(d)$ represents the fitness function after modification. To be used as the fitness function expression, two basic requirements of the original classical fitness function should be satisfied: first, the crowding degree of two same chromosomes equals to 1; second, when the difference of two chromosomes exceeds the niche radius, the crowding degree is estimated to be near and asymptotically zero. These two requirements can be expressed as

$$\begin{cases} sh_m(0) = 1, \\ sh_m(d') \simeq 0, \quad d' > \sigma_s. \end{cases} \tag{8}$$

In order to satisfy the first requirement, a deviation to the whole and a stretching on the amplitude can, respectively, apply to the Gaussian distribution; at the same time, using the "three-sigma principle" of the Gaussian distribution, let the three times of the standard deviation equal to the radius of the niche to meet the second requirement. Thus, equation (9) can be listed and substituted into equation (8) to calculate the coefficient K and the standard deviation sigma, as shown in equation (10). The fitness function based on the Gaussian distribution function after modification is shown in equation (11), and its comparison with the classical fitness function is shown in Figure 1(b).

$$\begin{aligned} sh_m(d) &= K f(d) \\ &= K \frac{1}{\sqrt{2\pi}\sigma} e^{d^2/2\sigma^2}, \quad \text{where } d \sim N\left(0, \sigma^2\right), \end{aligned} \tag{9}$$

$$\begin{cases} K = \sqrt{2\pi}\,\sigma, \\ \sigma = \frac{\sigma_s}{3}, \end{cases} \tag{10}$$

$$sh_m(d) = e^{-\left(9d^2/2\sigma_s^2\right)}. \tag{11}$$

3.2. Flow Description of NGA with Elite Strategy. The complete flow chart of the NGA algorithm is shown in Figure 2, and the algorithm steps are described as follows:

Step 1. (Population Initialization). A massive matrix representing the whole population should be initialized, while its number of rows is 1 more than the actual number of path points, and its number of columns equals the size of the population; each column is declared as an individual.

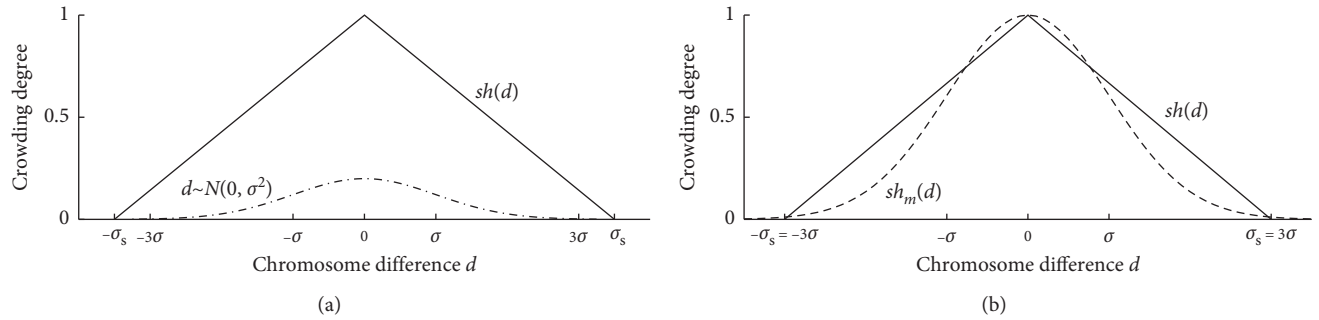

Figure 1: $sh(d)$, $sh_m(d)$, and Gaussian function. (a) $sh(d)$ vs. Gaussian function. (b) $sh(d)$ vs. $sh_m(d)$.

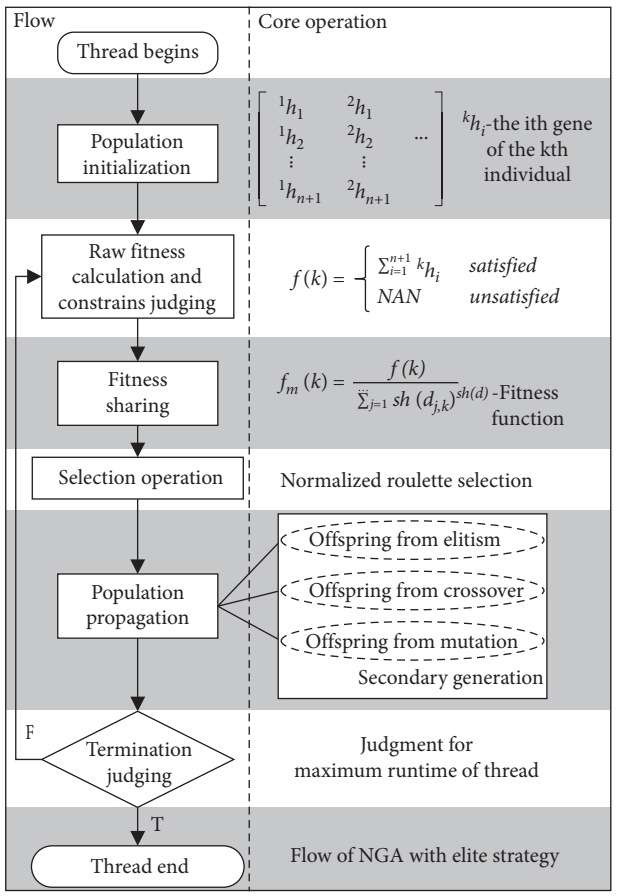

Figure 2: Flow chart for NGA with elite strategy.

Step 2. (Raw Fitness Calculation and Constrains Judging). Four bounding constrain judgments, including position bounding, velocity bounding, acceleration bounding, and jerk bounding, would be applied to every individual. Individuals who violate any of these constraints would be abandoned. If all constraints are satisfied, raw fitness scores would be calculated, which is exactly equal to trajectory traveling time.

Step 3 (Fitness Sharing) The fitness sharing technique can convert raw fitness scores into shared fitness. The function which was devoted to this conversion is termed fitness function. The classical fitness function is shown as equation (7) and the modified one is shown as equation(11). Also, the sum of fitness function values of individual i relative to j (including itself) is termed Niche Count, which would be used in the conversion of raw fitness. The niche count is a proper index for showing how heavily the competition has happened among individuals in the same niche. The larger of the niche count, the heavier of the competition.

Step 4 (Selection Operation). The unified roulette selection method would be used to choose the parent set from the population, who would have chances to produce offspring.

Step 5 (Population Propagation). The elite operation, crossover operation including mutate operation, would be simultaneously applied on the parent set, which is obtained from the previous step, producing three types of offspring constituting the next generation.

Step 6 (Termination Judging). NGA stopping criteria is judged in this step. If criteria are met, the optimization process finishes and otherwise jumps back to Step 2 and continues.

3.3. Dual-Thread Method Using the Characteristic of Elite Strategy. In the elite strategy, the elite individuals in each generation would directly enter the subsequence generation, and the most elite individual is the latest optimization results of the manipulator trajectory. These early optimization results may be eliminated from individual elite groups in the later stage with the birth of better results, but in fact, they are all feasible results, which is a kind of underutilization of resources in some sense. If the application method of GA is designed as a dual-thread method that both the manipulator worker thread and the GA optimization thread work simultaneously, the GA optimization thread can provide the optimal individual results of each generation for the robot worker thread to use at the earlier stage, and as a result, resources can be fully utilized.

The shared memory model, shown in Figure 3(a), plays a communication medium role, as well as Figure 3(b) shows the dual-thread framework in which the robot trajectory execution thread and the computational thread for trajectory optimization work simultaneously. The framework presented in Figure 3(b) consists of three parts: the manipulator

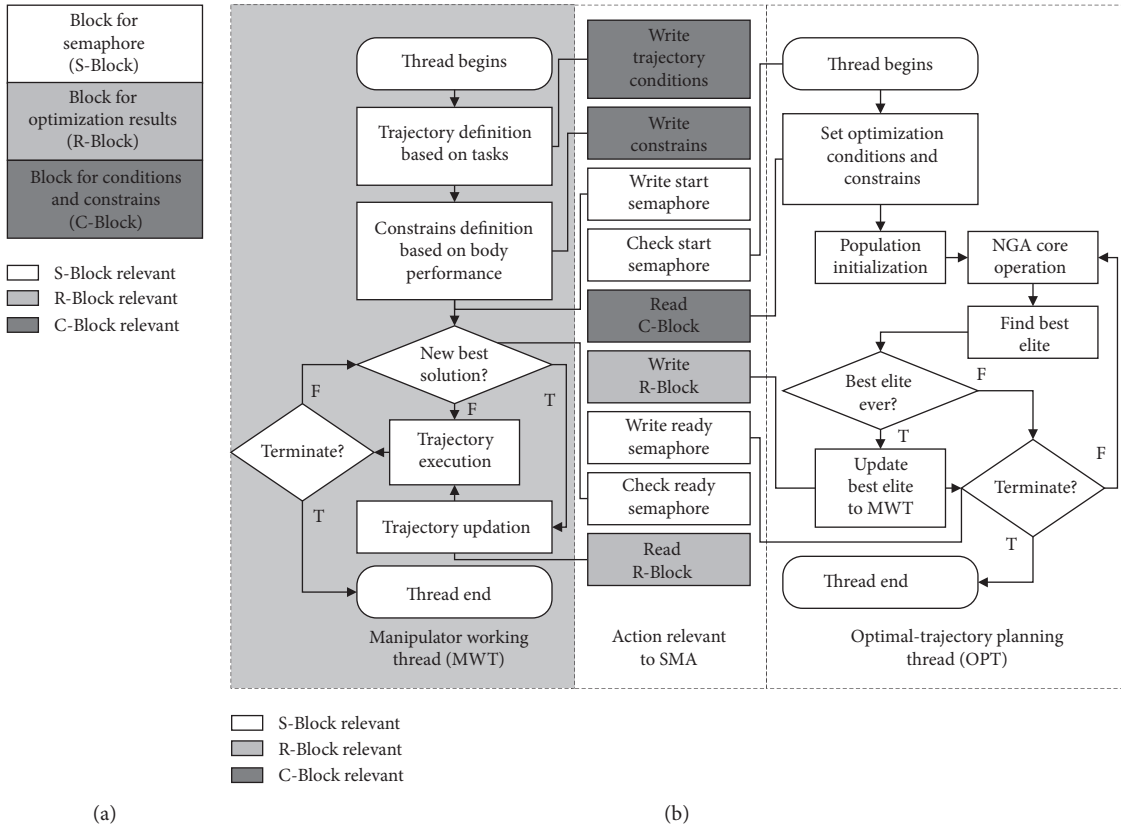

Figure 3: (a) Shared memory area. (b) Dual-thread framework.

working thread, the optimal-trajectory planning thread, and the shared memory area. Among them, the manipulator working thread (MWT) searches the executable trajectory information in the shared memory area and performs them as the real-time movement task; the optimal-trajectory planning thread (OPT) performs NGA with the elite strategy to optimize the joint space trajectory of the manipulator; the shared memory area (SMA) accomplishes data exchange missions between threads. However, the realization of obstacle avoidance and path geometry requirements of Cartesian space, as well as the transformation to joint space trajectory, are provided by the path planning algorithm of a higher level. Figure 3(a) shows the three partitions of SMA in order:(1) the memory block for semaphore (S Block), which is used to set the data access permission between processes, (2) the memory block for optimization results restoring (R Block), which were calculated by OPT and would be transferred to MWT, and (3) the memory block for trajectory constrains restoring (C Block), which is devoted to storing all constraints about trajectory, including geometry constrains, kinematic constraints, and dynamic constraints; last but not the least, data inside C Block are shared from MWT to OPT.

The dual-thread framework makes full use of elite individuals in each generation. The optimal elite individual in each generation of the population is shared to the MWT periodically through the SMA. The MWT can be started to execute the optimized trajectory before the final calculation of the OPT is completed. Moreover, the robot controller can obtain more and more optimized update trajectory from the OPT when dealing with a repetitive path. This framework can effectively solve the problems caused by improper design of algorithm termination conditions and traversal granularity when adopting a genetic algorithm for trajectory time-optimal planning and further improve the practicability of genetic algorithm in industrial application.

4. Simulation Experiment

4.1. Description of Simulation Environment. The robot object used in the simulation experiment is a Puma-like manipulator STR6-05, and its d-h parameters are shown in Table 1. The corresponding joint constraints are listed in Table 2. The path object going to be optimized in the simulation experiment is the same path described by multiple intermediate points, as shown in Table 3.

4.2. Normal Framework vs. Dual-Thread Framework. In order to prove the effectiveness of the proposed dual-thread framework, a simulation comparison experiment is carried out. By using the conventional single-thread method and the dual-thread method proposed in this paper, the execution efficiency of the simulation manipulator on a same trajectory (execution counts in the same limited time) was the comparing index.

Table 1: DH parameters of STR6-05.

#	θ (rad)	d	a	α (rad)	Joint variant	Length (mm)
1	$\theta_1 + \pi$	0	a_1	$-(\pi/2)$	θ_1	$a_1 = -27.5$
2	$\theta_2 + \pi/2$	0	a_2	0	θ_2	$a_2 = 450$
3	θ_3	0	a_3	$\pi/2$	θ_3	$a_3 = 35$
4	θ_4	d_4	0	$-(\pi/2)$	θ_4	$d_4 = 450$
5	θ_5	0	0	$\pi/2$	θ_5	
6	θ_6	d_6	0	0	θ_6	$d_6 = 100$

Table 2: Upper limits of different constrains for each joints.

#Joint	Velocity (deg/sec)	Acceleration (deg/sec^2)	Jerk (deg/sec^3)
1	100	45	160
2	95	40	60
3	100	75	55
4	150	70	70
5	130	90	75
6	110	80	70

Table 3: Path defined by multiple points.

Via points	#Joint					
	1°	2°	3°	4°	5°	6°
1	10	15	45	5	10	6
2	60	25	180	20	30	40
3	75	30	200	60	−40	80
4	130	−45	120	110	−60	70
5	110	−55	15	20	10	−10
6	100	−70	−10	60	50	10
7	−10	−10	100	−100	−40	30
8	−50	10	50	−30	10	20

Especially for the dual-thread method, as shown in Figure 4, the MATLAB memory-mapping file can help realize the simulation of the dual-thread method. The core is the memory mapping function "memmapfile" from MATLAB, which creates a mapping from a file on the hard disk to a piece of continuous memory. After establishing the mapping, what is stored in the memory is the filehandle to the corresponding location of the disk file and the length of the preset data element. After the process obtains the filehandle and relevant data length information in memory, it can access and modify the data in the specified position in the file in real-time, so as to realize the real-time modification of the same disk file by different program processes, that is, using the memory-mapping file to simulate the SMA. The MWT and the OPT individually occupy an independent MATLAB process to run but establish a unified memory structure to the same memory-mapping file. The MWT simulation process has run a manipulator simulation GUI, which was developed based on the MATLAB RTB (Robotics ToolBox) [20], providing kinematic simulation verification with the manipulator linkage model, as shown in Figure 5. On the other hand, the OPT simulation process has run an NGA based on the GA algorithm in MATLAB Global Optimization Toolbox. The specific processing flow between the two MATLAB processes obeys the suggested dual-thread framework.

As for the single-thread framework, the MWT and the OPT run in the same MATLAB process and have the sequential execution relationship: the simulation GUI needs to wait for the completion of NGA calculation, and the final result must be obtained before the execution of trajectory.

Several comparative groups of the single-thread framework and one dual-thread framework group with different termination conditions were designed. Their optimization algorithm termination conditions and NGA parameters setting are shown in Table 4. Since the maximum distance (MD) between individuals changes as the population evolves, if the niche radius was set relevant to MD, additional uncertainty was introduced to the population evolution. In order to reduce uncertainty, a fixed niche radius was used in the simulation experiment. After trial and error, 0.1 is a nonoptimal but reasonable setting value for the niche radius. Nine rounds of simulation experiments were carried out for each group, with the maximum running time set for each round ranging from 5 minutes to 45 minutes.

Figure 6(a) shows the corresponding simulation experiment results of 5 groups. An intuitive conclusion could be drawn: the trajectory execution counts of the dual-thread framework are higher than that of all single-thread framework groups. At the situation of the preset maximum running time of $t_{set} = 5$ min, the trajectory optimization of all single-thread groups has not given the result; there is no

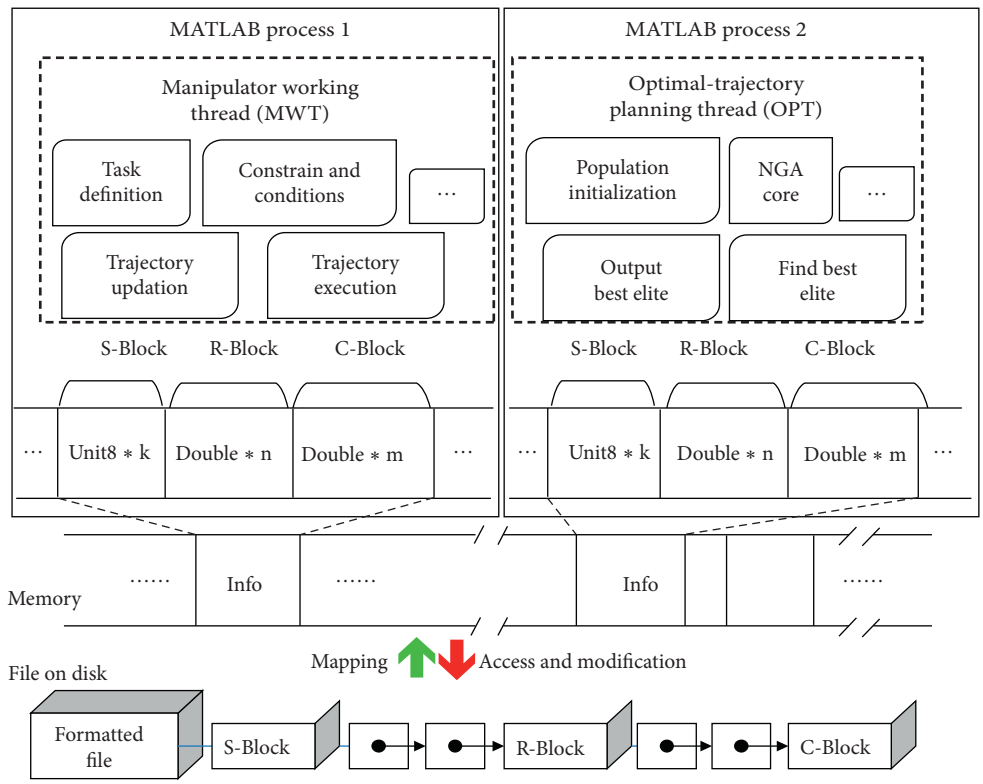

Figure 4: Simulation of dual-thread framework using two MATLAB processes.

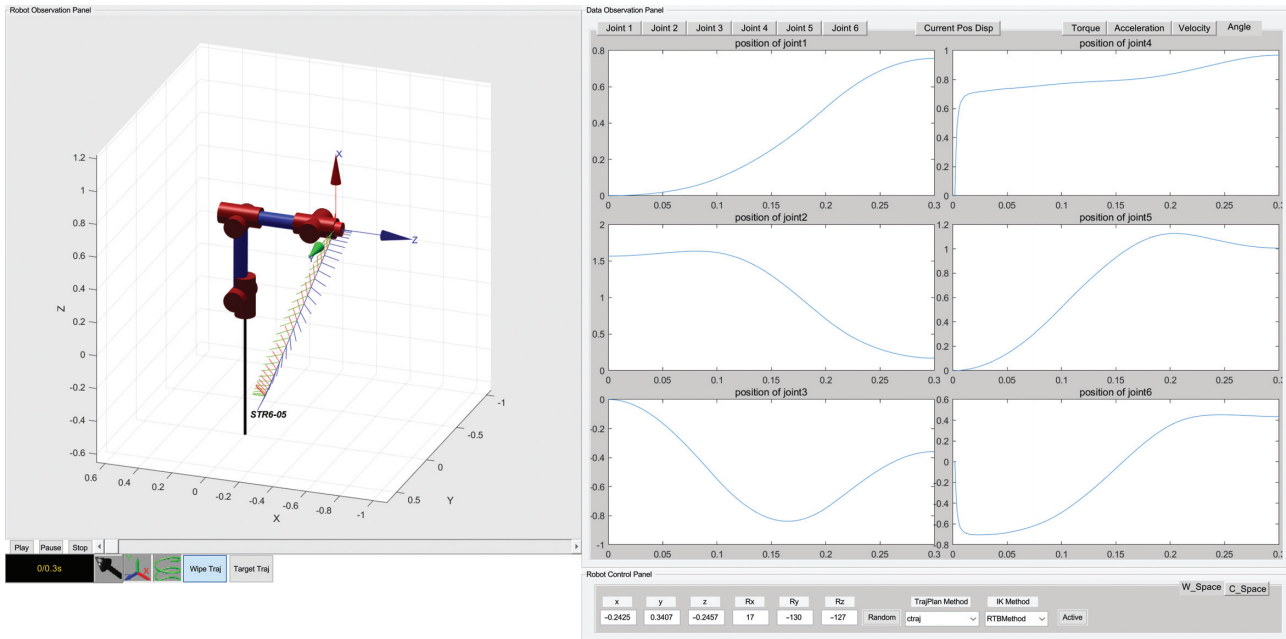

Figure 5: Simulation tool based on RTB.

executable trajectory, so the current execution counts of repeated trajectories are all zeros since the MWT is sequential running after the OPT. The dual-thread framework effectively utilizes the optimal elite individuals in the early stage of the population, so that the MWT can obtain an acceptable executable solution in advance and start the task earlier than the conventional single-thread framework. Figure 6(b) shows the execution counts varying with time for each group in a single round where the maximum running time $t_{set} = 35$ min. From Figure 6(b), it can be more intuitive to see the task startup sequence under different threshold conditions and frameworks. The earlier the intersection

Table 4: Table of comparative group settings.

Group number	Framework type	Termination conditions	Preset threshold	Population	σ_s
#1	Single-thread	(a) The preset maximum running time has reached (b) The relative change is less than the preset threshold	10^{-3}	150	0.1
#2	Single-thread		10^{-8}		
#3	Single-thread		10^{-12}		
#4	Single-thread		10^{-20}		
#5	Dual-thread				

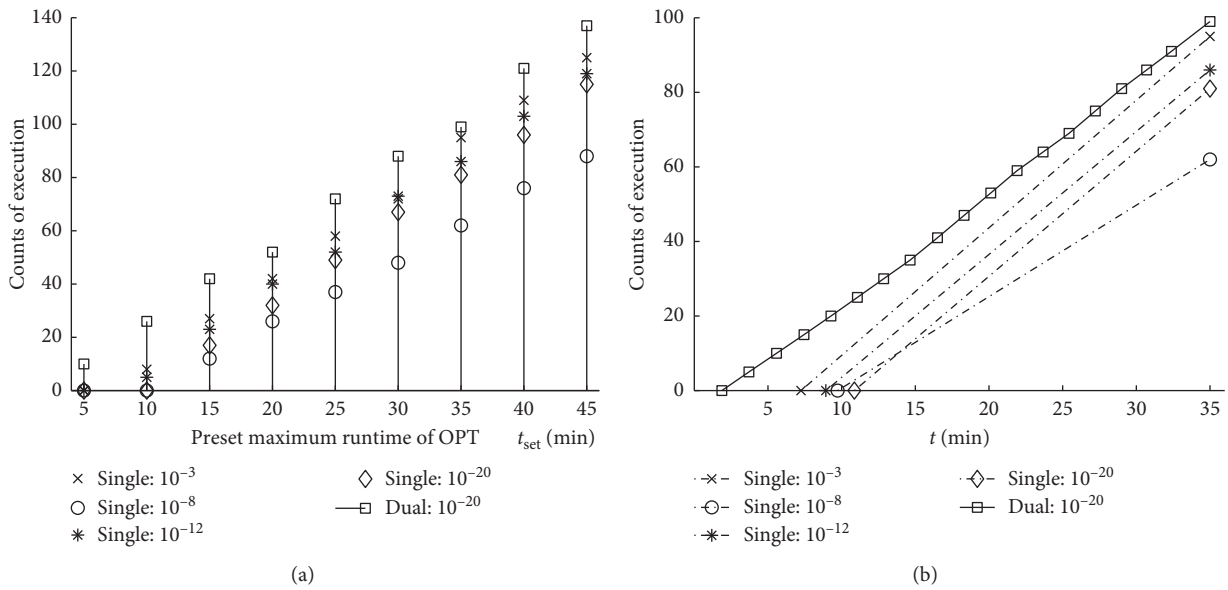

Figure 6: Comparison between single/dual-thread framework. (a) Stem plot of execution counts with respect to different preset maximum runtime of OPT with 9 rounds. (b) Plot of execution counts with respect to time for the single round with preset maximum runtime $t_{set} = 35$ min.

point between the curve and the X-axis mean the earlier the trajectory task startup time. The trajectory execution efficiency of each single-thread framework group was 3.42 counts/min, 2.45 counts/min, 3.30 counts/min, 3.36 counts/min, respectively, corresponding to threshold setting to 10^{-3}, 10^{-8}, 10^{-12}, 10^{-20}, and the execution efficiency of dual-thread framework group increased from 2.72 counts/min at the first stage to 3.06 counts/min at the last stage, with an efficiency increase of 12.5%.

Since the second joint has a broader range of motion and massively influences the position of the end-effector, it has been picked as the representative joint. Figures 7(a) and 7(b), respectively, show the comparison results of the final trajectories of the dual-thread and single-thread framework with the second joint of str6-05 at $t_{set} = 35$ min. The $j/a/v/p$ symbols on the vertical axis, respectively, represent the jerk, acceleration, velocity, and position of the joint. The trajectory curve has conformed to the cubic spline curve. According to the final time result of its horizontal axis shown and Figure 7(a), it can be seen that although the trajectory optimization result of the single-thread framework is slightly better than that of the dual-thread framework, the trajectory execution counts of the dual-thread framework are still more. The randomness of the GA result causes the unfortunate dual-thread result in this round. However, as long as the running time continues to be extended, the dual-thread framework still has the chance to produce and update better results.

4.3. Comparison Among Different Fitness-Sharing Schemes. With the purpose of comparing the effects of using modified fitness function, classical fitness function, and no niche technology, the aforementioned dual-thread framework was adopted, and the termination condition of the algorithm was reaching the upper limit of the preset maximum running time $t_{set} = 60$ min, or the relative changes of the best elite individual are less than the threshold 10^{-20}. In groups with niche behavior, the niche radius σ_s was 0.1. In the later stage of the simulation experiment, the population matrix data would be recorded. The scatter graph depicts the distribution of raw fitness scores for the whole population, in which the distance calculation of each individual was relative to the best elite individual in the generation. A histogram was further introduced to the distribution. Two graphs have been drawn on the same figure. Then, zooming in on the part within the niche radius σ_s, observation would be performed for analyzing the effect.

Figure 8 shows the results, and in Figure 8(a), $sh_m(d)$ is applied instead of $sh(d)$ to fitness sharing, and in Figure 8(b), $sh(d)$ fitness sharing is applied; moreover, in Figure 8(c), fitness sharing is not used. According to

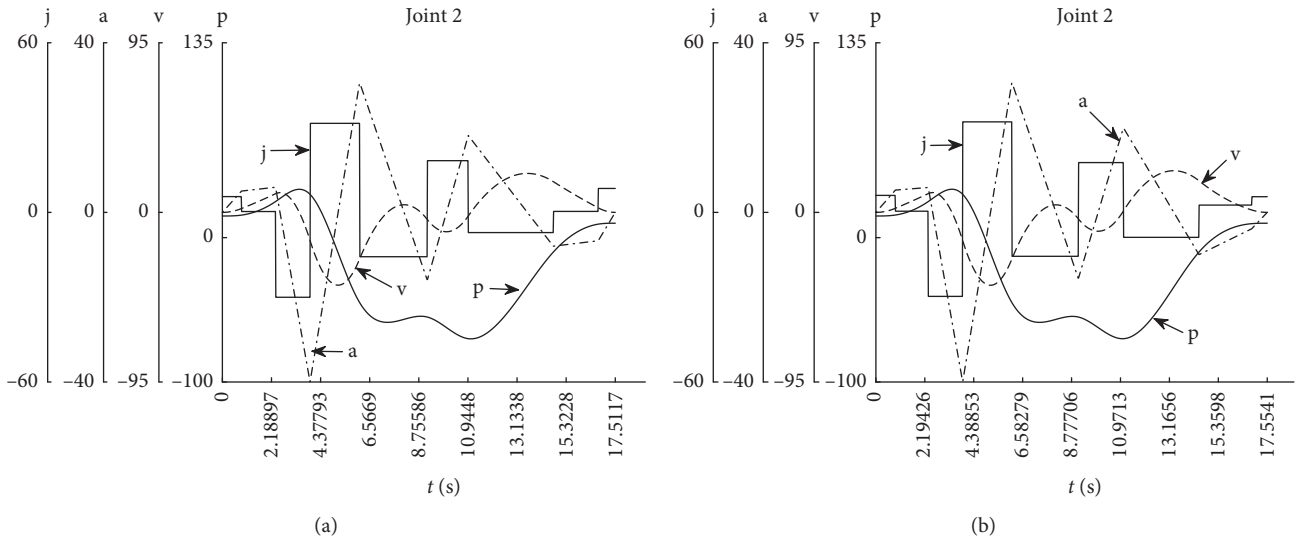

Figure 7: Trajectory result of single/dual-thread at Joint 2. (a) *j*/*a*/*v*/*p* plot with respect to time of the single-thread framework. The trajectory runtime equals to 17.5117 s. (b) *j*/*a*/*v*/*p* plot with respect to time of the dual-thread framework. The trajectory runtime equals to 17.5541 s.

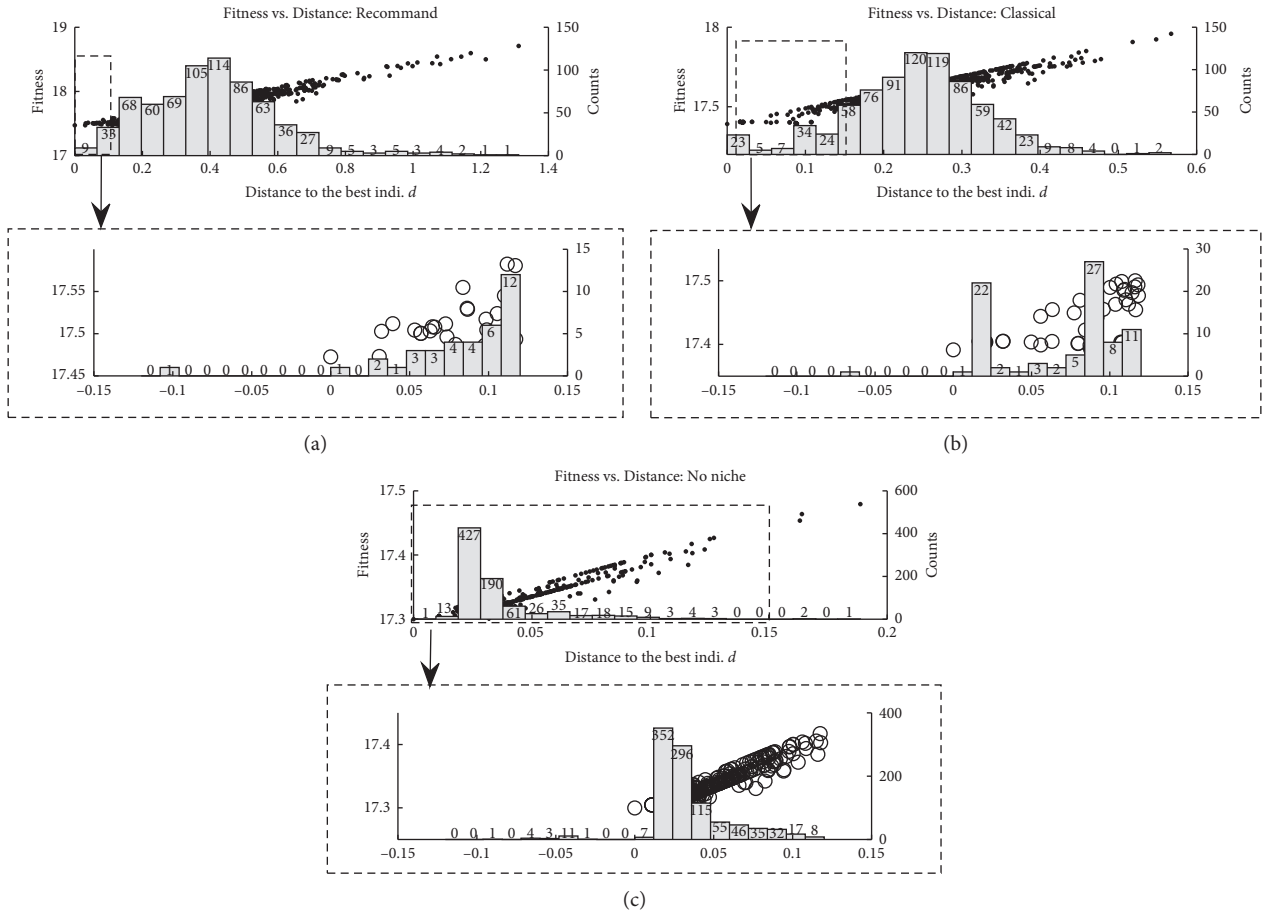

Figure 8: Comparison among 3 types of fitness sharing scheme. The upper half part of the figure is the origin plot, while the lower half is the result after zooming in to the niche radius. (a) Scheme with modified fitness sharing. (b) Scheme with classical fitness sharing. (c) Scheme without fitness sharing.

Figure 8(c), by observing the histogram of scatter points within the radius s of niche after zooming in, it can be seen that the phenomenon of population clustering near the optimal individual appears obviously without niche technology, and the closer to the optimal individual, the higher the clustering degree. This phenomenon would eventually lead to "premature" genetic algorithm and deterioration of the dual-thread framework proposed in this paper. By comparing the X-axis coordinate range of the original Figures 8(a), 8(b), and 8(c), it can be found that the range span of the population is larger after the use of niche technology; furthermore, the proposed modified fitness sharing has the most extensive range span of the population, signing that the NGA with modified fitness sharing can effectively increase the genetic diversity of the population.

By further comparison on the population distribution of the two different fitness sharing behavior, it can be seen that modified fitness sharing can effectively enhance the punishment applying on the individuals who are closer to the optimal individual since more individuals appeared near the border of the radius. As a result, the species gene pool would not massively be occupied by the optimal individual genes. For individuals whose location in the distribution is less than zero, it is either an individual that violates the constraint and would be eliminated in the next generation or a new feasible result that is better than the current optimal one.

5. Conclusion

(1) The modification of the classical fitness function based on the Gaussian distribution was proposed to reduce the subjective process of adjusting the effect of fitness function while rationalizing the estimation of population individual crowding degree. It also helps the NGA algorithm get rid of the "premature" defect.

(2) The dual-thread framework utilizing the characteristics of the elite strategy of NGA is proposed to make exhaust use of the early-stage optimal elite individuals in each generation of population, to solve the problem caused by the course decision of the termination conditions of genetic algorithm and the difficulty in determining the granularity of traversal domain.

Data Availability

The data, simulation tool without third-party libraries, and developed algorithm script of this study are available from the corresponding author upon request.

Acknowledgments

This work was supported by the Science and Technology Project of Guangdong Province (China) (Grant no. 2017B090910012).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] H. Dong, G.-s. Du, D. Liu, and M. Cong, "Joint-space trajectory planning for palletizer robot based on the genetic algorithm," *Modular Machine Tool & Automatic Manufacturing Technique*, vol. 5, p. 8, 2017.

[2] X.-F. Chen, "Research on the trajectory planning algorithm of palletizing robots," *Control Engineering of China*, vol. 5, pp. 925–929, 2018.

[3] Y. Liu, M. Cong, D. Liu, and F. Zhu, "Trajectory planning for porcine abdomen cutting based on an improved genetic algorithm and machine vision for industrial robot," *Robot*, vol. 3, pp. 377–384, 2017.

[4] Y. Hou, T. Wang, J. Yue, and Z. Jia, "Path planning for dual-robot coordinate welding based on multi-objective genetic algorithm," *China Mechanical Engineering*, vol. 29, no. 16, p. 1984, 1989.

[5] H. Du, Z. Lin, Y. Zhang, H. Chen, and Z. Xu, "Improved migration genetic algorithm optimization for variable-speed falling trajectory of mobile crossbeam in composite hydraulic press with pressure shock," *IEEE Access*, vol. 7, pp. 39459–39473, 2019.

[6] Y. Ren, C. Zhang, F. Zhao, H. Xiao, and G. Tian, "An asynchronous parallel disassembly planning based on genetic algorithm," *European Journal of Operational Research*, vol. 269, no. 2, pp. 647–660, 2018.

[7] L. Zhang, J. Wang, J. Chen, K. Chen, B. Lin, and F. Xu, "Dynamic modeling for a 6-dof robot manipulator based on a centrosymmetric static friction model and whale genetic optimization algorithm," *Advances in Engineering Software*, vol. 135, Article ID 102684, 2019.

[8] Z. Zhou, C. Wang, Z. Zhu, Y. Wang, and D. Yang, "Sliding mode control based on a hybrid grey-wolf-optimized extreme learning machine for robot manipulators," *Optik*, vol. 185, pp. 364–380, 2019.

[9] T. Al Khawli, M. Anwar, A. Sunda-Meya, and S. Islam, "A calibration method for laser guided robotic manipulation for industrial automation," in *Proceedings of the IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, pp. 2489–2495, IEEE, Washington, DC, USA, October 2018.

[10] W. Deng, Z. Qiwan, P. Liu, and R. Song, "Optimal time trajectory planning based on dual population genetic and chaotic optimization algorithm," *Computer Integrated Manufacturing Systems*, vol. 24, no. 1, pp. 101–106, 2018.

[11] D. Kang and M. Chen, "Planning and simulation of robot optimal trajectory based on improved chaotic search algorithm," *Computer Engineering and Applications*, vol. 53, no. 14, pp. 143–147, 2017.

[12] Y. Liu, C. Guo, and Y. Weng, "Online time-optimal trajectory planning for robotic manipulators using adaptive elite genetic algorithm with singularity avoidance," *IEEE Access*, vol. 7, pp. 146301–146308, 2019.

[13] Z. Chen and W. Zhou, "Path planning for a space-based manipulator system based on quantum genetic algorithm," *Journal of Robotics*, vol. 2017, Article ID 3207950, 10 pages, 2017.

[14] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, Springer Science & Business Media, Berlin, Germany, 2010.

[15] C. Lin, P. Chang, and J. Luh, "Formulation and optimization of cubic polynomial joint trajectories for industrial robots," *IEEE Transactions on Automatic Control*, vol. 28, no. 12, pp. 1066–1074, 1983.

[16] J. rey Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," in *Proceedings of the first IEEE conference on evolutionary computation, IEEE World Congress on Computational Intelligence*, vol. 1, pp. 82–87, Citeseer, Orlando, FL, USA, June 1994.

[17] S. W. Mahfoud, "Crossover interactions among niches," in *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pp. 188–193, IEEE, Orlando, FL, USA, June 1994.

[18] A. Pétrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 798–803, IEEE, Nagoya, Japan, May 1996.

[19] B. Sareni and L. Krahenbuhl, "Fitness sharing and niching methods revisited," *IEEE Transactions on Evolutionary Computation*, vol. 2, no. 3, pp. 97–106, 1998.

[20] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB® Second, Completely Revised*, Vol. 118, Springer, Berlin, Germany, 2017.