Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 19(1): 643–662.
DOI: 10.3934/mbe.2022029
Received: 28 August 2021
Accepted: 01 November 2021
Published: 18 November 2021

Research article

A Bio-inspired trajectory planning method for robotic manipulators based on improved bacteria foraging optimization algorithm and tau theory

Zhiqiang Wang, Jinzhu Peng*and Shuai Ding

School of Electrical Engineering, Zhengzhou University, No. 100 of Science Avenue, Zhengzhou 450001, China

* **Correspondence:** Email: jzpeng@zzu.edu.cn.

Abstract: In this paper, a novel bio-inspired trajectory planning method is proposed for robotic systems based on an improved bacteria foraging optimization algorithm (IBFOA) and an improved intrinsic Tau jerk (named Tau-J*) guidance strategy. Besides, the adaptive factor and elite-preservation strategy are employed to facilitate the IBFOA, and an improved Tau-J* with higher-order of intrinsic guidance movement is used to avoid the nonzero initial and final jerk, so as to overcome the computational burden and unsmooth trajectory problems existing in the optimization algorithm and traditional interpolation algorithm. The IBFOA is utilized to determine a small set of optimal control points, and Tau-J* is then invoked to generate smooth trajectories between the control points. Finally, the results of simulation tests demonstrate the eminent stability, optimality, and rapidity capability of the proposed bio-inspired trajectory planning method.

Keywords: robotic manipulator; trajectory planning; improved bacteria foraging optimization algorithm; Tau theory; interpolation algorithm; multi-objective optimization

1. Introduction

Robotic manipulators have been widely used in various fields such as electronics industries, automobile industries, military affairs, space exploration, and mining. Furthermore, the application fields of manipulators continue to expand, such as riot control, health industries, catering industries, construction industries, and explosion prevention. Trajectory planning, as one of the most imperative topics on manipulators, has been investigated since the manipulators appeared. A considerable number of approaches have been proposed to solve the robot planning problem, such as artificial potential field (APF) [1, 2], probabilistic roadmap [3], rapidly-exploring random tree (RRT) [4], configuration space method [5], optimization algorithm [6, 7], reinforcement learning [8, 9], metaheuristics [10–13], and Tau theory [14–21]. The goal of trajectory planning is to make robots move fast, accurately, and

stably. Besides, researchers also dedicate to make robots safer to the circumstances, more intelligent, and more compliant. In recent years, Tau theory has been utilized on robots to plan trajectories due to the compliant feature [17–21].

Since many objectives need to be achieved in robot planning, the optimization algorithms provide a promising way to solve the challenging issue. For example, the improved sparrow search algorithm was proposed to plan the shortest path for mobile robots [22]. The modified particle swarm optimization and the Tabu search algorithm were used to handle inverse kinematics and obstacle avoidance problems [23]. A novel multi-group particle swarm optimization was proposed to achieve motion planning [24]. The ant colony optimization algorithm was applied on unmanned air vehicles trajectory planning tasks [25]. Huang et al. [26] employed the elitist non-dominated sorting genetic algorithm — version II (NSGA-II) to optimize the 5th-order B-spline trajectory for an industrial robot manipulator. NSGA-II was also adopted to tune the proportional-integral-derivative (PID) controller of the robotic manipulator [27]. A hybrid differential evolution-based method was presented to address the 6-degree of freedom (DOF) manipulator trajectory planning problem [28]. Most optimization algorithms are imitated from the practical long-tested-by-nature behaviors of creatures. This inspired the design philosophy of the algorithms, which are also called bio-inspired optimization algorithms. Among the bio-inspired optimization algorithms, bacteria foraging optimization algorithm (BFOA) was successfully applied on liquid level control [29], which indicates the effectiveness of being employed on robot planning.

In this paper, a novel bio-inspired trajectory planning method is proposed for robotic systems based on the improved BFOA (IBFOA) and Tau-J*. The main contributions can be described as follows.

(1) The trajectory planning of the manipulator is considered as a multi-objective optimization problem. Then, the multi-objective optimization algorithm IBFOA is proposed to generate a set of optimal control points in the trajectory planning. Compared with the original BFOA, the IBFOA is presented with decreased reproduction size, decreased elimination & dispersal probability, and the elite-preservation strategy. In this way, the IBFOA can achieve better performance and more stable optimization in comparison to the original BFOA. Besides, the IBFOA performance improvement can be measured as the mean fitness change which is 6%. Similarly, the stability improvement can be measured as the fitness standard deviation change which is 7%. The improvement requires nearly no extra computation.

(2) A bio-inspired trajectory generation algorithm Tau-J* is proposed to interpolate the waypoints between the control points, enabling the calculation burden to be alleviated and the collision-free compliant trajectory of the manipulator to be obtained. Compared with other Tau-based guidance strategies, Tau-J* has the benefits of zero initial and final jerk due to the usage of the high-order Tau function. Next, a more compliant trajectory can be acquired using the proposed Tau-J* guidance strategy. Hence, a novel IBFOA-Tau-J* algorithm is designed to facilitate robotic manipulator planning.

The rest of the paper is structured as follows. In Section 2, the related work is introduced. In Section 3, the original BFOA, IBFOA, and the trajectory generation based on IBFOA are described. In Section 4, the interpolation algorithm Tau-J* is presented. Afterward, the simulation tests on a 6-DOF manipulator are conducted based on the proposed algorithm, and the test results are discussed in Section 5. and the discussion is also given. Finally, the conclusions and our future work are drawn in Section 6.

2. Related work

BFOA, as a bio-inspired optimization algorithm, was proposed to imitate the foraging behaviors of *Escherichia coli*, including chemotaxis, swarming, reproduction, and elimination & dispersal. Subsequently, the BFOA and its improved algorithms were presented to address the optimization problem. For instance, an adaptive BFOA was presented by adding a constant parameter in the denominator of the step size formula [30]. Similarly, Chen et al. [31] modified the chemotaxis behavior by introducing population diversity of the bacterial colony, universal iterations, mean fitness in two chemotaxis processes. Different from the above adaptive BFOAs, the nonlinear decreasing methods, the roulette gambling mechanism, and the linear decreasing adaptive regulation mechanism were introduced to chemotaxis behavior, reproduction behavior, and elimination & dispersal behavior, respectively [32]. BFOA was applied to optimize the controller parameters [33], and the better results were obtained compared to the conventional proportional-integral (PI) controller and genetic algorithm-based controller. Abd-Elazim et al. [34] built the hybrid particle swarm optimization-BFOA (BSO) to optimize the design of power system stabilizers and demonstrated the superiority of the proposed method to BFOA and PSO. Additionally, BFOA was used to optimize the nonlinear load frequency controller parameters [35]. Panda et al. [36] introduced the principle of swarming to hybrid BFOA and PSO, which was tested in automatic generation control. The BFOA was applied to real-time pose adjustment of the Hexa robot [37]. Besides, the multi-subpopulation bacterial foraging optimization algorithm [38] was proposed to facilitate unmanned surface vehicle path planning by constructing a new strategy that protects the elite member in the subpopulation. The above investigations suggest that the BFOA and its improved algorithms possess good performances in parameters optimization and searching extremum value. However, the application of BFOA for trajectory planning of robotic manipulators needs to be further investigated because it may fail to find the global optimum, and the current solution may become worse during iteration. Thus, the reproduction behavior and the elimination & dispersal behavior are modified by introducing the elite-preservation strategy and a decay factor d_k on the two behaviors. In this way, a novel improved BFOA (IBFOA) can be conducted to facilitate the trajectory planning of robotic manipulators.

Notably, the unexpected calculation burden and unsmooth trajectories would occur provided that the IBFOA (or BFOA) is implemented alone on the trajectory planning of robotic manipulators. One feasible way is to establish smoothness evaluation criteria in the cost function to obtain the smooth trajectories. However, the calculation burden problem would be further aggravated since more terms were introduced in the cost function. With the purpose of alleviating the above deficiencies, IBFOA is adopted to generate a small set of control points, and the interpolation algorithms (such as polynomial interpolation [39], cubic spline [40], linear segment with parabolic blend [41], and Tau theory) are then performed to generate the waypoints between the control points. In this way, both fast calculation and smooth trajectories can be achieved.

Tau theory, as a bio-inspired interpolation method, was applied to generate the trajectory of robotic systems in recent years owing to its fast calculation and smooth trajectory generation. In the early stage, researchers focused on the physical meaning of Tau [14, 15]. Then, Tau theory was employed to describe the law of creatures approaching objects [16]. A so-called intrinsic Tau jerk (Tau-J) guidance strategy was proposed and applied to quadrotors [17]. Then, Tau-J also implemented on manipulators [18], and the performance was thoroughly discussed [21]. Besides, Tau harmonic (Tau-H) was

proposed by introducing harmonic movement into Tau theory and used on unmanned aerial vehicles based on PSO to overcome the shortages of Tau-J [19]. Moreover, Lee modified the work of [16], making it possible to guide a movement to a departing destination [20]. However, Tau theory cannot be used to handle the global multi-objective trajectory planning of robotic manipulators.

3. Improved bacteria foraging optimization algorithm

In this section, the fundamentals of the bacteria foraging optimization algorithm (BFOA) are first presented. Then a novel improved bacteria foraging optimization algorithm (IBFOA) is proposed. Different from the existed BFOAs, the decay factors are used in the reproduction behavior and elimination & dispersal behavior. Furthermore, the elite-preservation strategy is introduced in the proposed algorithm. Moreover, the trajectory generation for the robotic manipulator is described based on the proposed IBFOA.

3.1. Fundamentals of BFOA

BFOA [29] is a bio-inspired optimization algorithm that imitates the foraging behavior of *Escherichia coli*. The *E. coli* will survive and breed in nutrient-rich places and will die in a food-lacking or noxious circumstance. As a result, *E. coli* tend to move and stay in a comfort zone. In BFOA, all the bacteria only need to chase the global minimum (or maximum) of the fitness function by taking the following four kinds of behaviors.

(a) **Chemotaxis**: It consists of tumbling and swimming. In short, the bacteria tumble to change direction and swim forward.

(b) **Swarming**: Each bacterium attracts and repels other bacteria to approach while all the bacteria cannot be too close.

(c) **Reproduction**: The bacteria in better places will live and reproduce asexually, and others will die. The population remains unchanged.

(d) **Elimination & dispersal**: Every bacterium will be eliminated and dispersed to another random place stochastically with a constant probability.

At the early stage of the BFOA optimization process, the potential optimal domain can be obtained by searching the whole domain field. As the optimization process proceeds, the step size remains fixed and has to be selected carefully in advance to determine a compromise between efficiency and optimal solutions. Given the drawback, different kinds of adaptive-step BFOA were proposed [30–32]. Moreover, the best bacteria may be eliminated and dispersed to another random place, resulting in a worse current solution.

3.2. Improved BFOA

A decay factor d_k is defined and multiplied by the reproduction size S_r and the elimination & dispersal probability P_{ed}. S_r and P_{ed} will decrease over time by the decay factor. Consequently, the reproduction and elimination & dispersal behaviors are improved. The advantage is that the bacteria will focus on exploring the searching space and then pay increasing attention to find better solutions around the current optimal solution, contributing to a quick achievement of the optimal solutions. Generally, the decay variables are linearly reduced or exponentially reduced. In this paper, the decay factor d_k is defined as,

$$d_k = e^{\frac{1-n}{N}} \tag{3.1}$$

where $N = N_c \cdot N_{re} \cdot N_{ed}$ is the total iteration number, N_c denotes the number of the chemotaxis steps, N_{re} refers to the number of reproduction steps, N_{ed} represents the number of elimination & dispersal steps, and n is the current iteration number (starts at 1). A decay variable can be obtained by multiplying d_k by a nonzero constant.

Different from the previous research [30–32], a novel IBFOA is proposed with decreased reproduction size S_r, decreased elimination & dispersal probability P_{ed}, and the elite-preservation strategy. By introducing the elite-preservation strategy to the elimination & dispersal behavior, the bacterium in the best place will always survive, protecting the current optimum. In this study, Θ is defined as the search agents and stores all the bacterium positions. Define the bacteria $\Theta(j,k,l) = [\theta^1(j,k,l), \ldots, \theta^S(j,k,l)]$, where S indicates the number of bacteria, $\theta^i(j,k,l)$ represents a p-dimensional position vector with $i = 1, \cdots, S$; $j = 1, \cdots, N_c$; $k = 1, \cdots, N_{re}$; and $l = 1, \cdots, N_{ed}$ that stores the i-th bacterium position at the j-th chemotactic step, the k-th reproduction step, and the l-th elimination & dispersal step. The IBFOA (compared with BFOA) can be described as Algorithm 1.

Define a fitness function $J(i, j, k, l)$ as,

$$J(i, j, k, l) = J_c(i, j, k, l) + J_{ar}(i, j, k, l) \tag{3.2}$$

where $J_c(i, j, k, l)$ represents the cost function that will be defined later, and $J_{ar}(i, j, k, l)$ denotes the combined attraction and repelling effect, defined as,

$$\begin{aligned} J_{ar}(i, j, k, l) \quad = \quad & \sum_{i=1}^{S} \left[-d_{attract} \exp\left(-w_{attract} \sum_{m=1}^{p} \left(\theta_m - \theta_m^i \right)^2 \right) \right] \\ & + \sum_{i=1}^{S} \left[h_{repellant} \exp\left(-w_{repellant} \sum_{m=1}^{p} \left(\theta_m - \theta_m^i \right)^2 \right) \right] \end{aligned} \tag{3.3}$$

where θ_m represents the m-th dimension of the bacterium position, θ_m^i is the m-th dimension of the i-th bacterium position, $d_{attract}$ denotes the depth of the attractant, $w_{attract}$ strands for the width of the attractant, $h_{repellant}$ indicates the height of the repellant, and $w_{repellant}$ refer to the width of the repellant.

The objective of IBFOA is to minimize the fitness function $J(i, j, k, l)$ so as to to determine the optimal solution stored by J_{last} according to Algorithm 1. Besides, the health function J_{health}^i is defined as follows to evaluate the health of the bacteria:

$$J_{health}^i = \sum_{j=1}^{N_c+1} J(i, j, k, l), i = 1, \cdots, S \tag{3.4}$$

The objective of the health function J_{health}^i is to examine whether the i-th bacterium obtains enough nutrients during its lifetime of foraging evaluate "healthy" [29].

3.3. Trajectory planning based on IBFOA

First, it is assumed that the trajectory planning for an n-DOF robotic manipulator is conducted in an environment with N_{obs} obstacles using the proposed IBFOA. Additionally, the obstacles are set to be spheres randomly in the workspace to simplify the calculations. The initial and final joint angles are also set to be random values if no collision happens.

Algorithm 1: IBFOA (compared with BFOA)

Initialize the parameters S, p, $d_{attract}$, $w_{attract}$, $h_{repellant}$, $w_{repellant}$, N_c, swimming steps N_s, N_{re}, S_r, N_{ed}, P_{ed}, and tumbling steps N_t. Initialize d_k for IBFOA (no decay factor in BFOA);
for $l = 1 : N_{ed}$ **do**
  for $k = 1 : N_{re}$ **do**
    for $j = 1 : N_c$ **do**
      for $i = 1 : S$ **do**
        Fitness Eq (2);
        $J_{last} = J(i, j, k, l)$;
        Tumble: generate a normalized random vector $\Delta(i) \in R^p$;
        Move: let $\theta^i(j + 1, k, l) = \theta^i(j, k, l) + N_t(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$;
        Fitness $J(i, j + 1, k, l) = J_c(i, j + 1, k, l) + J_{ar}(i, j + 1, k, l)$;
        Swim: **for** $m = 1 : N_s$ **do**
          if $J(i, j + 1, k, l) < J_{last}$ **then**
            $J_{last} = J(i, j + 1, k, l)$;
            Move;
          else
            Break;
          end
        end
      end
    end
    for $i = 1 : S$ **do**
      Eq (4);
    end
    Sort the bacteria by J_{health};
    The better $S_r \cdot d_k$ (S_r for BFOA) bacteria survive and reproduce, the other bacteria die;
  end
  for $i = 2 : S$ ($i = 1 : S$ for BFOA) **do**
    $\theta^i(j, k, l)$ is eliminated and dispersed to another random place with probability $P_{ed} \cdot d_k$ (P_{ed} for BFOA);
  end
end

Second, trajectory planning is regarded as a multi-objective optimization problem. Then, the proposed IBFOA can be employed to obtain the optimal trajectory. With the purpose of avoiding the unstable and cumbersome inverse kinematics, all control points and waypoints are calculated in the joint space of the robotic manipulator, and forward kinematics is applied in collision detection. The robot links are modeled as the cylinders with hemispheres for each joint, and the obstacles are modeled as the spheres. Afterward, a collision model composed of the cylinders with hemispheres and the spheres is established. In the collision model, the cylinder with hemispheres is abstracted as the line segment, while the radius of the cylinder, the modeling error compensation, and the safe stopping distance are all modeled in the spheres. In this way, only the distance between the line segment and the center of the sphere needs to be calculated in the collision model for collision detection. The cost function is defined as,

$$J_c = \omega_1 L + \omega_2 L_q + \omega_3 CP \tag{3.5}$$

where ω_1, ω_2, and ω_3 are weights, L denotes the path length of the end effector, L_q represents the sum of angles of all joints, C refers to the collision flag (0 and 1 indicates no collision and collision, respectively), and P suggests the penalty that will increase when the manipulator approaches and collides the obstacles. The path length L can be defined as,

$$L = \sum_{k=1}^{N_w-1} \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2 + (z_{k+1} - z_k)^2} \tag{3.6}$$

where N_w denotes the overall number of waypoints, k is the number of current waypoint, x_k, y_k, and z_k describe the current position of the end effector. The sum of angles L_q is expressed as,

$$L_q = \sum_{k=1}^{N_w-1} \sum_{i=1}^{N_{DOF}} \left| q_{k+1}^i - q_k^i \right| \tag{3.7}$$

where N_{DOF} indicates the degree of freedom of the manipulator, and q^i denotes the i-th joint of the manipulator. The penalty P can be defined as,

$$P = \sum_{k=1}^{N_w} \sum_{j=1}^{N_{obs}} \max\left(1 - \frac{d_{link}^j}{r_{obs}^j + r_{expansion}^j}, 0\right) \tag{3.8}$$

where N_{obs} is the number of obstacles, r_{obs}^j denotes the radius of the j-th obstacle, $r_{expansion}^j$ denotes the expansion radius of the j-th obstacle, and

$$d_{link}^j = \sum_{i=1}^{N_{DOF}} dist\left(l^i, obs^j\right) \tag{3.9}$$

represents the distance between the j-th obstacle and all links of the manipulator at k-th waypoint, where l^i indicates the i-th link, and obs^j represents the j-th obstacle, respectively.

Generally, the weight parameters ω_1, ω_2, and ω_3 can be manually selected to minimize the moving path of the manipulator in both Cartesian space and joint space. Notably, ω_3 is chosen to be large, allowing the fitness to be significantly increased provided that the manipulator collides with obstacles.

Then, the planned trajectory can be obtained by using the proposed IBFOA. However, it is inefficient to obtain every single waypoint of a trajectory through IBFOA since the calculation burden will be aggravated and the unsmooth trajectories will be generated. One feasible way to handle these problems is that a small set of control points can be planned using IBFOA, and then the plenty of waypoints between two adjacent control points can be generated by an interpolation algorithm. It suggests that only the positions of control points for all the joints of the manipulator need to be optimized using the proposed IBFOA. In this way, the calculation burden and the unsmooth trajectories will be reduced since the interpolation algorithms can generate the trajectories between the two adjacent control points as smoothly as possible with simple computation.

4. Tau-J*: an interpolation algorithm

Several interpolation algorithms are used to generate the trajectories of industrial robotic manipulators, so as to improve the implementation efficiency. However, the common interpolation algorithms generally suffer from the nonzero initial jerk and final jerk, which would damage the actuators. Thus, a bio-inspired interpolation algorithm, Tau theory, is utilized to generate a smooth trajectory and achieve the compliant motion. Based on the Tau theory, an improved Tau-J (Tau-J*) is presented in this section to generate the trajectory with zero initial and final jerk, contributing to achieving the compliant trajectory.

4.1. Tau theory

Lee [14] investigated how creatures utilize visual information when approaching objects and then proposed the Tau theory in 1976. Lee et al. [15] explicitly described the Tau function $\tau(x)$ as,

$$\tau(t) = \frac{x(t)}{\dot{x}(t)} \tag{4.1}$$

where $x(t)$ denotes the gap between the current position and the object, and $\dot{x}(t)$ indicates the approaching velocity.

If there are two movements reaching the object simultaneously, the relationship of the two movements can be described as,

$$\tau(t) = k_A \tau_A(t) = k_A \frac{x_A(t)}{\dot{x}_A(t)} \tag{4.2}$$

where k_A represents the coupling coefficient [16], and $x_A(t)$ refers to the gap of the intrinsic guidance movement (IGM), which represents a virtual movement guiding the actual gap $x(t)$. Given the IGM, one can obtain the IGM-guided actual movement by Eq (4.2).

Intrinsic Tau jerk (Tau-J) guidance strategy [19] was proposed by applying constant jerk movement as IGM $x_A(t) = \frac{1}{6} j \left(T^3 - t^3\right)$. By solving Eq (4.2), the actual movement can be obtained as,

$$\begin{cases} x(t) = \frac{x_0}{T^{3/k}} \left(T^3 - t^3\right)^{1/k} \\ \dot{x}(t) = \frac{-3x_0 t^2}{kT^{3/k}} \left(T^3 - t^3\right)^{1/k-1} \\ \ddot{x}(t) = \frac{3x_0 t}{kT^{3/k}} \left(\frac{3-k}{k} t^3 - 2T^3\right) \left(T^3 - t^3\right)^{1/k-2} \end{cases} \tag{4.3}$$

where x_0 denotes the initial motion gap, T indicates the reach time, and k represents the coupling coefficient. Tau-J can guide a movement with zero initial acceleration and zero final acceleration. The IGM of Tau-J is a cubic polynomial, causing the second derivative of gap (acceleration) to start and end at zero. However, the initial jerk and final jerk are nonzero, resulting in uncompliant movement, even damage to the actuators.

4.2. Improved Tau-J (Tau-J)*

To make the movement more complaint, we propose an improved intrinsic Tau jerk (Tau-J*) guidance strategy. With a quartic polynomial as the IGM, $x_A(t) = \frac{1}{24} j^* \left(T^4 - t^4\right)$, the third derivative of gap (jerk) will start and end at zero. It can be obtained that,

$$\begin{cases} x(t) = \dfrac{x_0}{\left(\frac{1}{T^4-t^4}\right)^{1/k}\left(T^4\right)^{1/k}} \\ \dot{x}(t) = -\dfrac{4t^3 x_0}{k\left(T^4-t^4\right)^2\left(\frac{1}{T^4-t^4}\right)^{\frac{1}{k}+1}\left(T^4\right)^{1/k}} \\ \ddot{x}(t) = -\dfrac{4t^2 x_0\left(3T^4k+kt^4-4t^4\right)}{k^2\left(T^4-t^4\right)^2\left(\frac{1}{T^4-t^4}\right)^{1/k}\left(T^4\right)^{1/k}} \end{cases} \tag{4.4}$$

Since the higher-order IGM is used, Eq (4.4) can guide the movement with zero initial jerk and zero final jerk. Tau-J, Tau-J*, quintic polynomial, and cubic spline algorithms are compared to demonstrate the performances of the interpolation algorithms. It is assumed that the gaps are 0, 300 m, 200 m, −300 m, −700 m at 0, 100 s, 200 s, 300 s, 400 s, respectively, which can be considered one start point, three control points, and one endpoint. Then, the four interpolation algorithms are used to generate trajectories, respectively. Figure 1 illustrates the interpolation results of the gap, velocity, acceleration, and jerk curves, with the parameters $k = 0.1$, and $T = 100$ chosen for Tau-J and Tau-J*, respectively. The results demonstrate that the Tau-J, Tau-J*, and quintic polynomial can stop at every control point while the cubic spline algorithm can generate a nonstop trajectory. As revealed from Figure 1(c), the accelerations of the cubic spline are nonzero values at the start and end of the movement, and this would cause instability or damage the actuators. Figure 1(d) indicates that the jerks of Tau-J, quintic polynomial, and cubic spline are discontinuous and nonzero at the start and end of the movement, while the jerk of the proposed Tau-J* is continuous and zero at the start and end of the movement. It indicates that Tau-J* can guide more compliant movements.

5. Simulation

In this section, the simulation tests are conducted for a robotic manipulator on the robot simulator CoppeliaSim (also known as V-REP) to validate the performances of the proposed bio-inspired trajectory planning method.

5.1. Simulation Procedure

We set up a 6-DOF robotic manipulator in the CoppeliaSim, where three obstacles are assumed to exist in the workspace. The initial and final joint angles of the robotic manipulator are set to be random, and the positions of the obstacles are randomly chosen in the workspace. The multi-objective collision-free compliant trajectories are obtained with the proposed bio-inspired trajectory planning

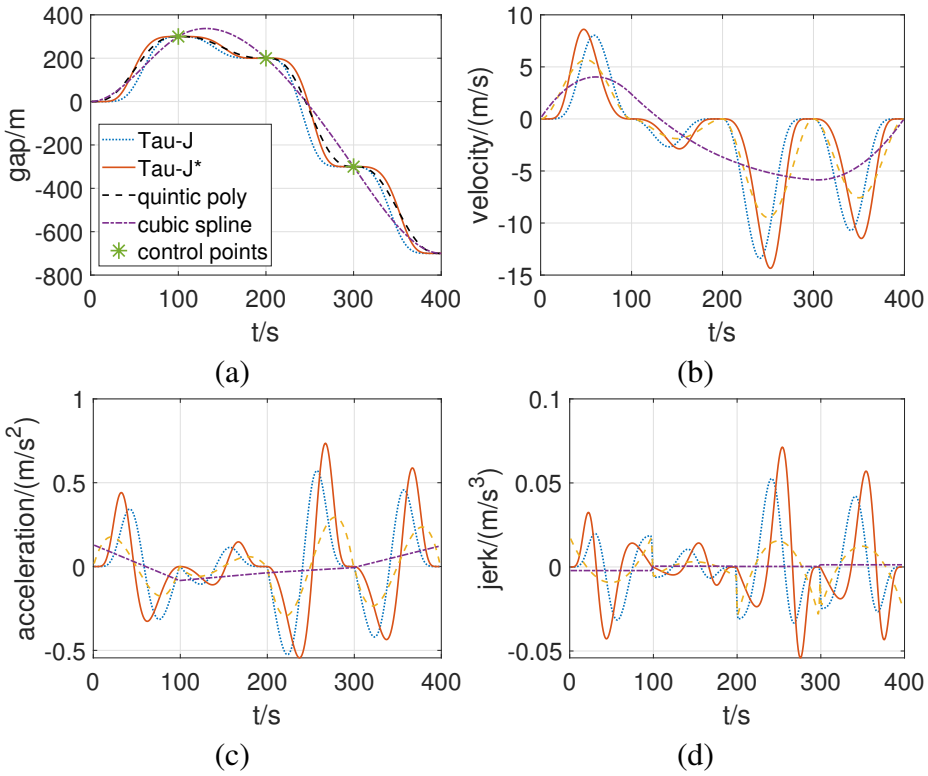

Figure 1. The curves of Tau-J, Tau-J*, quintic polynomial, and cubic spline.

method. The control points are updated every iteration through IBFOA in the joint space, and the waypoints between the control points are generated by Tau-J*. Within such an algorithm design, the joint angle, velocity, acceleration, jerk, and torque are reasonable values, and there is no need to evaluate kinematical and dynamical constraints. Particularly, a larger number of waypoints is desired since collision detection is used for every waypoint. Nonetheless, more waypoints would lead to more computational consumption. Therefore, one can make a trade-off between optimality and computational efficiency by choosing an adequate number of waypoints. Following the analyses of Section 3.3 and above, the numbers of the control points and waypoints are both selected by the trial-and-error method.

The whole procedure is described in Algorithm 2, where the function $CptF(i, j, k, l)$ that computes the fitness is invoked as Algorithm 3.

Algorithm 2: Bio-inspired Trajectory Planning for Robotic Manipulator

Create the manipulator model;
Create the obstacle model;
do
    `// Set random values when colliding`
    Set the random initial and final joint angles of manipulator;
    Set the random obstacle position;
while *collide*;
Initialize the parameters of IBFOA;
Generate the initial solution randomly;
Run the main body of IBFOA (Algorithm 1), but use `CptF`(i, j, k, l) to calculate the fitness;

Algorithm 3: Compute the fitness

Function `CptF`(i, j, k, l)**:**
    Interpolate the control points by Tau-J* (Eq (4.4));
    Compute the path length of end effector (Eq (3.6));
    Compute the sum of angles of all joints (Eq (3.7));
    Compute the the penalty (Eqs (3.8 and 3.9));
    Compute the the collision flag for all waypoints;
    Compute the fitness (Eqs (3.5) and (3.3));
    return fitness;
end

5.2. Simulation results

Two cases are conducted under the same condition to compare the proposed method with other similar methods. For simulation, it is assumed that there are 3 obstacles in the circumstance. By the trial-and-error method, 3 control points, 1 initial point and 1 endpoint can be chosen, followed by 120 waypoints between the adjacent control points. Case 1 is implemented to compare the performances of IBFOA, BFOA, and BSO; Case 2 is conducted to demonstrate the convergence of the solutions by using IBFOA BFOA, and BSO, respectively. Notably, IBFOA, BFOA, and BSO are implemented with Tau-J*. The parameters of the two cases are exhibited in Table 1, where some parameters are selected according to the literature [29, 35] and the others are tuned by the trial-and-error method. BFOA and BSO share the same parameters. The tests are performed on the desktop with a 3.4 GHz central processing unit (CPU) and 16 GB DDR4 memory.

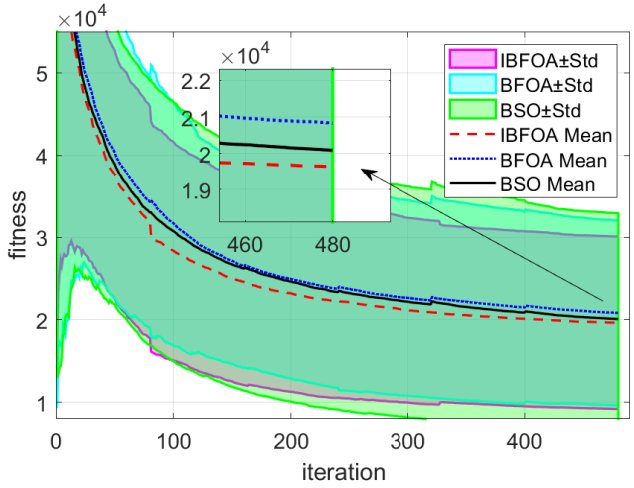

Figure 2. Mean fitness and standard deviation of Case 1.

Table 1. Parameters of the simulation.

symbol	Case 1		Case 2	
	IBFOA	BFOA & BSO	IBFOA	BFOA & BSO
S	12	12	30	30
N_{re}	8	8	20	20
S_r	6	6	15	15
N_{ed}	6	6	15	15
N_c	10	10	30	30
N_t	$0.2\text{ver}_S{}^*$	0.2ver_S	0.2ver_S	0.2ver_S
p	$18^\dagger$	18	18	18
$d_{attract}$	0.1	0.1	0.1	0.1
$w_{attract}$	20	20	20	20
$h_{repellant}$	0.1	0.1	0.1	0.1
$w_{repellant}$	100	100	100	100
N_s	4	4	4	4
P_{ed}	0.2	0.2	0.2	0.2
T	6	6	6	6
k	0.1	0.1	0.1	0.1
r_{obs}	0.18	0.18	0.18	0.18
$r_{expansion}$	0.1	0.1	0.1	0.1
ω_1	7	7	7	7
ω_2	300	300	300	300
ω_3	1.2×10^7	1.2×10^7	1.2×10^7	1.2×10^7

Note: $*$ ver_S represents S-dimensional vector that all elements are 1.

† 3 control points for each joint of the 6-DOF manipulator, and each dimension of p is limited on the interval $(-\pi, \pi)$ as the joint limit of the robotic manipulator.

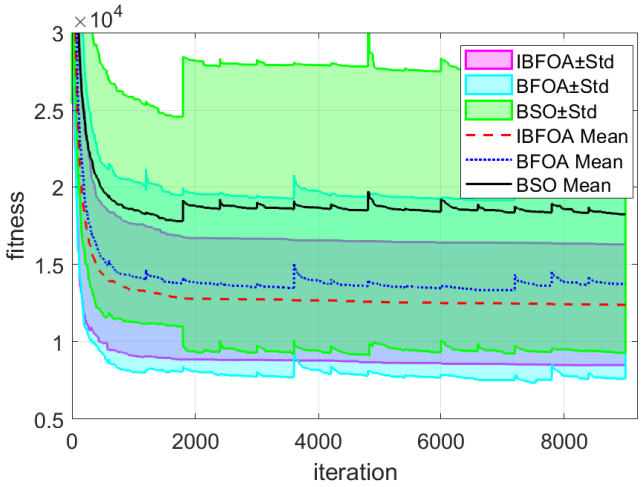

Figure 3. Mean fitness and standard deviation of Case 2.

Case 1: Three hundred results of IBFOA and BFOA are collected respectively, with each result containing 480 iterations. Figure 2 illustrates the mean fitness and standard deviation, where ±std denotes the standard deviation. As indicated in Figure 2, IBFOA has faster convergence with lower fitness compared to BFOA and BSO. This demonstrates that IBFOA can achieve better solutions and be more stable.

Case 2: Twenty results of IBFOA, BFOA, and BSO are collected respectively, with each result containing 9000 iterations. Figure 3 shows the mean fitness and standard deviation, revealing that IBFOA is superior to BFOA and BSO in the stability, speed of convergence, and quality of solutions with the larger iteration numbers.

Table 2 provides the final fitness (mean ± std) of Case 1 and Case 2, suggesting that IBFOA can achieve smaller final fitness. Specifically, the smaller mean value generally indicates the better solution, and the smaller standard deviation value reflects the more stable optimization algorithm.

Table 2. Final fitness (Mean ± Std).

	IBFOA	BFOA	BSO
Case 1	$1.96 \times 10^4 \pm 1.04 \times 10^4$	$2.08 \times 10^4 \pm 1.12 \times 10^4$	$2.01 \times 10^4 \pm 1.28 \times 10^4$
Case 2	$1.24 \times 10^4 \pm 3.89 \times 10^3$	$1.37 \times 10^4 \pm 6.13 \times 10^3$	$1.82 \times 10^4 \pm 8.95 \times 10^3$

Besides, a set of optimal solutions of IBFOA (with Tau-J*) and BFOA (with Tau-J) is chosen respectively to be visualized on CoppeliaSim, so as to further intuitively demonstrate the performances of the proposed trajectory planning method. Figures 4 and 5 present the visualized scenes with the same positions of the obstacles and initial and final joint angles obtained using IBFOA-Tau-J* and BFOA-Tau-J, respectively. The three obstacles are colored in yellow, blue, and red, respectively; the path of the end-effector is drawn as black lines; the coordinates at the bottom right in each subfigure indicate the angle of view; the endpoint and control points of the end-effector in Cartesian space are exhibited with white dots.

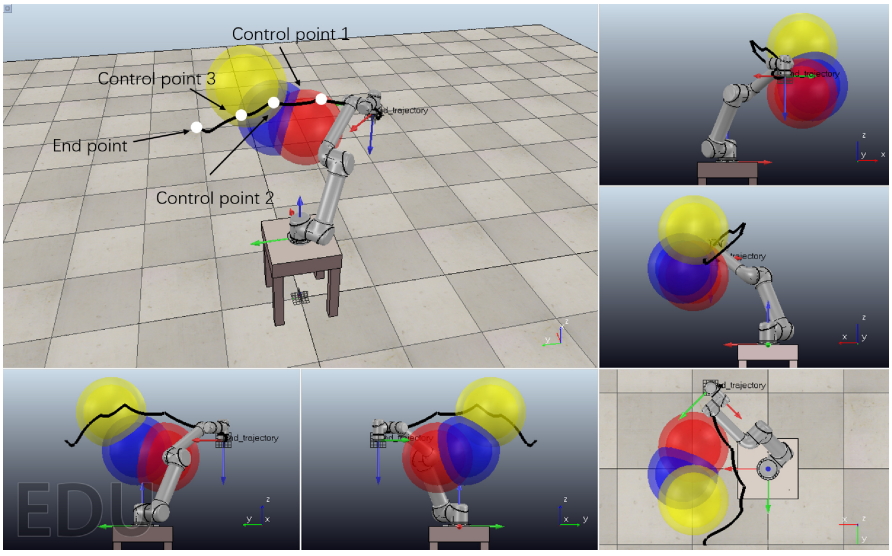

Figure 4. The visualized scene by IBFOA.

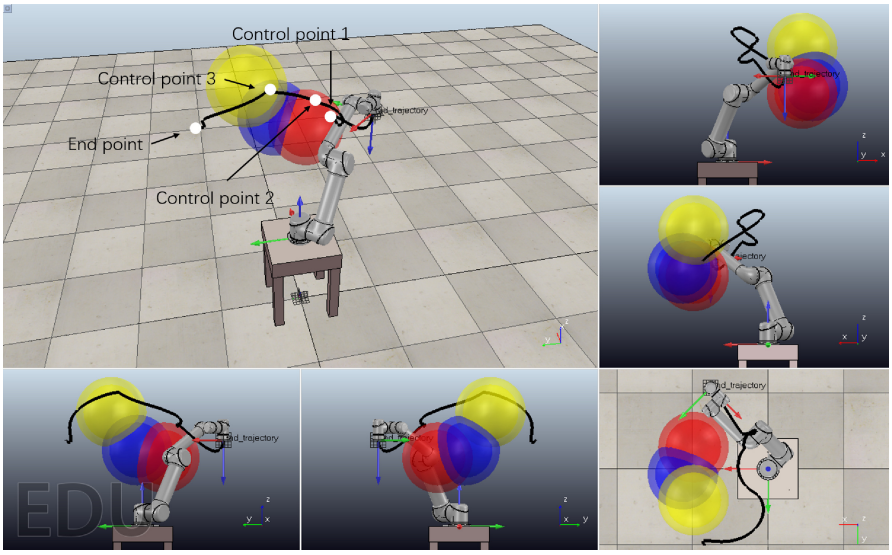

Figure 5. The visualized scene by BFOA.

Figures 6 and 7 exhibit the angle, angular velocity, angular acceleration, angular jerk, torque, and end-effector position obtained by IBFOA and BFOA, respectively. The control points at 6s, 12s and 18s of planned trajectories are indicated by the black vertical lines.

5.3. Discussion

As illustrated in Figures 2 and 3, the faster the fitness decreases, the smaller the mean and standard deviation of the observable fitness. This implies that the proposed IBFOA can achieve faster converges, better solutions, and more stable performance compared to BFOA and BSO. Moreover, there are the stepped increases in the mean BFOA and BSO fitness (Figure 3), which worsen the optimization process. The reason is that the elimination & dispersal behavior forces several bacteria "rebirth" in a random place, making the best bacterium "rebirth" in a bad place. According to the parameters in Table 1, there are $N_c \times N_{re} = 30 \times 20 = 600$ iterations in the chemotaxis and reproduction behaviors, and the stepped increases would occur every 600 iterations. In Figure 2, the stepped mean fitness increase would also appear every $N_c \times N_{re} = 10 \times 8 = 80$ iterations. The mean fitness of IBFOA does not increase in Figures 2 and 3 because of the elite-preservation strategy.

Multiple parameters should be tuned in IBFOA to achieve good solutions. Larger S and N_c boost the chance to find global optimal while raising the computational burden. N_t can be considered the search step length. If N_t is too large the bacteria may swim through local optimal; if N_t is too small the bacteria may take too many steps to find local minimal and be unable to jump out. In Case 1 and Case 2, N_t is adequately chosen under the consideration of the manipulator joint limit and the desired solution coarseness. The swimming behavior is controlled by N_s to determine how long the bacteria should swim to the better solution. This behavior leads to the random evaluation times of the cost function and causes inconvenience when IBFOA is compared with other optimization algorithms. N_s and S_r control the "rebirth" in good places; N_{ed} and P_{ed} control the "rebirth" in random places. Since the "rebirth" behaviors are improved by d_k, large values can be selected and will decay in the optimization progress.

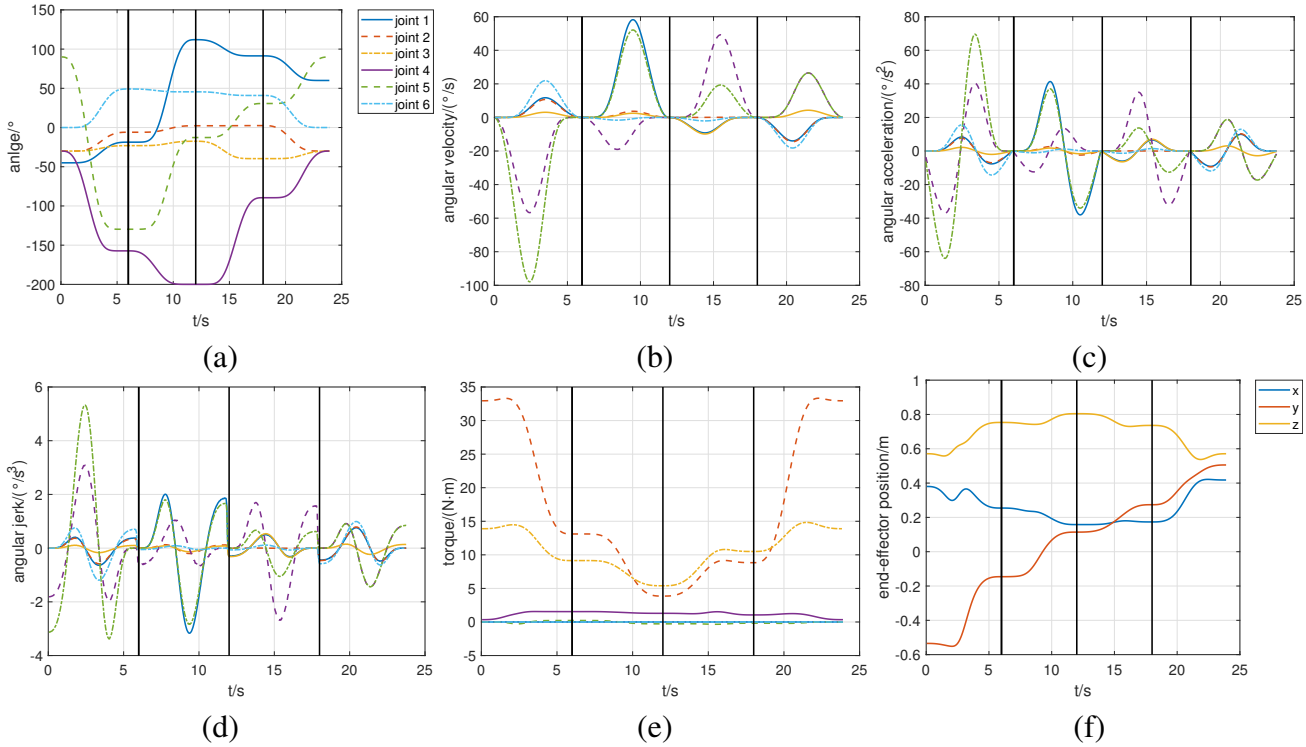

Figure 6. The joint angle, velocity, acceleration, jerk, torque and end-effector postion by IBFOA.

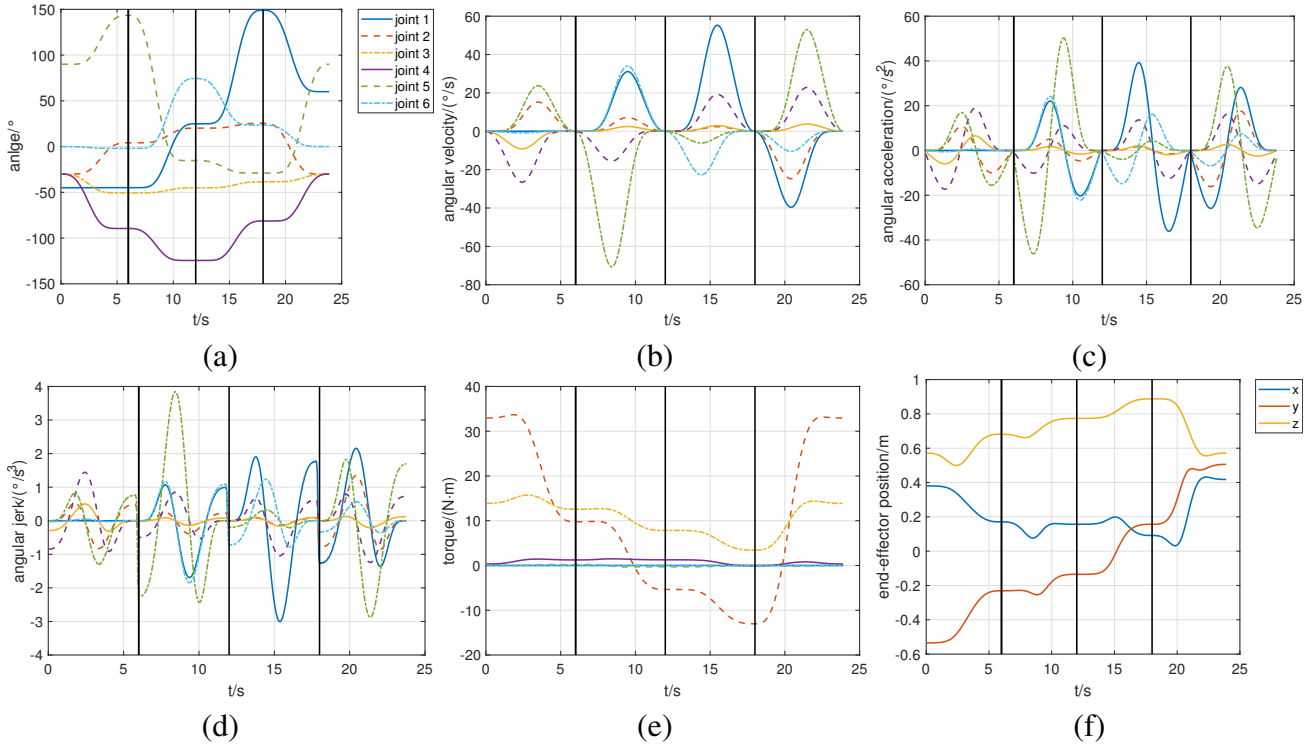

Figure 7. The joint angle, velocity, acceleration, jerk, torque and end-effector postion by BFOA.

In Figures 4 and 5, both of the algorithms can avoid obstacles. However, the end-effector position curve of BOFA is intuitively longer than that of IBFOA, reflecting the smaller L in the cost function obtained by IBFOA.

By comparing Figure 6 with Figure 7, the subfigures (a), (b), and (c) in both figures demonstrate that the compliant angle, velocity, and acceleration curves can be generated by both IBFOA and BFOA. However, in subfigures (d), the jerk of IBFOA is zero at every control point while the jerk of BFOA may suddenly change at control points leading to uncompliant movements. Meanwhile, more compliant trajectories can be obtained by using Tau-J*. In subfigures (f), the end-effector position of IBFOA is quite more steady than that of BFOA at $6 \sim 18$ s.

Case 1 takes 54.924 seconds on average for 480 iterations to generate a feasible trajectory according to 300 tests. Case 2, which is used for testing the performance of IBFOA in a long-running process, takes 2520.544 seconds on average for 9000 iterations to test the converge performance of IBFOA according to 20 tests. Moreover, the number of iterations can be chosen adequately. The feasible trajectory can be obtained as shown in Figures 4 and 5 in Case 1. Based on the average runtime of Case 1, the computation time is acceptable for the trajectory planning task since the robotic manipulator moves in a static circumstance. According to the experimental tests, 93% runtime is used to calculate the cost function as the collision detection has to be calculated at all the waypoints and control points to guarantee a collision-free trajectory; 2% runtime and 3% runtime are consumed by IBFOA and Tau-J*, respectively; the remaining 2% runtime is occupied by other codes.

Additionally, the experimental tests are conducted for the end-effector trajectory planning using IBFOA-Tau-J*, BFOA-Tau-J, and RRT methods to demonstrate the effective performance of the proposed method. The experimental results are presented in Figure 8, where the results of each method are obtained for running around 60 seconds. Compared with BFOA-Tau-J and IBFOA-Tau-J* methods, RRT can be used to discover feasible solutions by sampling the domain field in high-dimensional spaces without prior knowledge. However, the RRT method may be inefficient considering that there are no potential optimal areas in the whole domain field for finding the solutions due to the absence of priori knowledge. As suggested in Figure 8, all the planned trajectories are achieved to be collision-free with the above methods. Nevertheless, the shortest and most compliant collision-free trajectory can be achieved by the proposed IBFOA-Tau-J* method compared with BFOA-Tau-J and RRT methods.

6. Conclusions

In this study, an IBFOA has been proposed for the trajectory planning of robotic manipulators by introducing adaptive factor and elite-preservation strategy. A, bio-inspired interpolation algorithm Tau-J* with high-order IGM has been designed to generate trajectories between the control points obtained by IBFOA. The advantages of the proposed method can be concluded that the calculation burden would be alleviated, facilitating the compliant collision-free trajectory planning to be obtained. The simulation results demonstrate that the proposed bio-inspired trajectory planning method has superiority in stability, speed of convergence, and quality of solutions. However, it possesses some disadvantages. First, some parameters in the proposed method should be manually chosen. Second, the computation time is too long for trajectory planning in a dynamic environment. Regarding the future work, a criterion will be developed to determine the best number of control points, the fast collision detection algorithm will be designed to promote the cost function calculation using the MoveIt collision checking

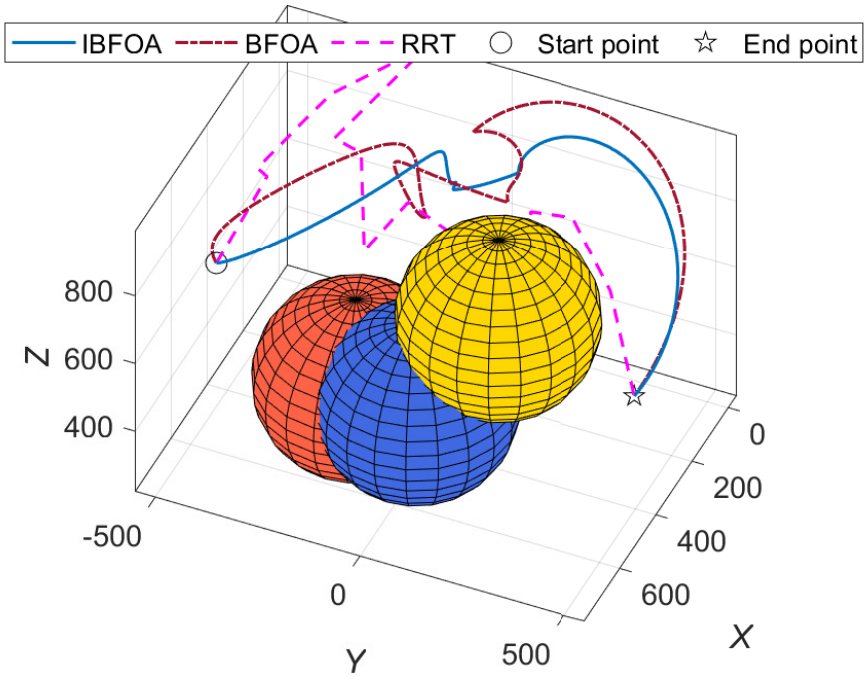

Figure 8. The comparison of the end-effector trajectory planning.

module, the other behaviors of BFOA will be further improved, and the effectiveness will be analyzed by performing statistical tests [42–45].

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 61773351), and the Program for Science & Technology Innovation Talents in Universities of Henan Province (Grant No. 20HASTIT031).

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, in *Autonomous Robot Vehicles*, Springer, (1986), 396–404.
2. R. Rodrigues, M. Basiri, A. Aguiar, P. Miraldo, Low-level active visual navigation: Increasing robustness of vision-based localization using potential fields, *IEEE Robot. Autom. Lett.*, **3** (2018), 2079–2086. doi: 10.1109/LRA.2018.2809628.
3. L. Kavraki, P. Svestka, J. Latombe, M. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE Trans. Robot. Automat.*, **12** (1996), 566–580. doi: 10.1109/70.508439.

4. S. LaValle, Rapidly-exploring random trees: A new tool for path planning, *Computer Sci. Dept.*, **1** (1998), 1–4.
5. D. Roy, Study on the configuration space based algorithmic path planning of industrial robots in an unstructured congested three-dimensional space: An approach using visibility map, *J. Intell. Robot. Syst.*, **43** (2005), 111–145. doi: 10.1007/s10846-005-9011-7.
6. J. López, P. Sanchez-Vilariño, M. D. Cacho, E. L. Guillén, Obstacle avoidance in dynamic environments based on velocity space optimization, *Robot. Auton. Syst.*, **131** (2020), 1–21. doi: 10.1016/j.robot.2020.103569.
7. B. Morrel, R. Thakker, G. Merewether, R. Reid, M. Rigter, et al., Comparison of trajectory optimization algorithms for high-speed quadrotor flight near obstacles, *IEEE Robot. Autom. Lett.*, **3** (2018), 4399–4406. doi: 10.1109/LRA.2018.2868866.
8. C. R. Gil, H. Calvo, H. Sossa, Learning an efficient gait cycle of a biped robot based on reinforcement learning and artificial neural networks, *Appl. Sci.*, **9** (2019), 1–22. doi: 10.3390/app9030502.
9. E. López-Lozada, E. Rubio-Espino, J. H. Sossa-Azuela, V. H. Ponce-Ponce, Reactive navigation under a fuzzy rules-based scheme and reinforcement learning for mobile robots, *PeerJ Comput. Sci.*, **7** (2021), 1–25. doi: 10.7717/peerj-cs.556.
10. S. Fong, S. Deb, A. Chaudhary, A review of metaheuristics in robotics, *Comput. Electr. Eng.*, **43** (2015), 278–291. doi: 10.1016/j.compeleceng.2015.01.009.
11. A. K. Kashyap, D. R. Parhi, Multi-objective trajectory planning of humanoid robot using hybrid controller for multi-target problem in complex terrain, *Expert. Syst. Appl.*, **179** (2021), 1–23. doi: 10.1016/j.eswa.2021.115110.
12. J. Pierezan, R. Z. Freire, L. Weihmann, G. Reynoso-Meza, L. D. Coelho, Static force capability optimization of humanoids robots based on modified self-adaptive differential evolution, *Comput. Oper. Res.*, **84** (2017), 205–215. doi: 10.1016/j.cor.2016.10.011.
13. A. K. Kashyap, D. R. Parhi, Particle swarm optimization aided pid gait controller design for a humanoid robot, *ISA Trans.*, **114** (2021), 306–330. doi: 10.1016/j.isatra.2020.12.033.
14. D. Lee, A theory of visual control of braking based on information about time-to-collision, *Perception*, **5** (1976), 437–459. doi: 10.1068/p050437.
15. D. Lee, M. N. O. Davies, P. R. Green, F. R. Weel, Visual control of velocity of approach by pigeons when landing, *J. Exp. Biol.*, **180** (1993), 85–104. doi: 10.1242/jeb.180.1.85.
16. D. Lee, General tau theory: evolution to date, *Perception*, **38** (2009), 837–858.
17. Z. Zhang, S. Zhang, P. Xie, O. Ma, Bioinspired 4D trajectory generation for a UAS rapid point-to-point movement, *J. Bionic. Eng.*, **11** (2014), 72–81. doi: 10.1016/S1672-6529(14)60021-4.
18. S. Zhang, Z. Zhang., J. Qian, Bio-inspired trajectory planning for robot catching movements based on tau theory, *J. Mech. Eng.*, **50** (2014), 1–8.
19. Z. Yang, Z. Fang, P. Li, Bio-inspired collision-free 4D trajectory generation for UAVs using tau strategy, *J. Bionic Eng.*, **13** (2016), 84–97. doi: 10.1016/S1672-6529(14)60162-1.
20. Z. Yang, Z. Fang, P. Li, Decentralized 4D trajectory generation for UAVs based on improved intrinsic tau guidance strategy, *Int. J. Adv. Robot Sys.*, **13** (2016), 1–13. doi: 10.5772/63431.

21. Z. Zhang, X. Yang, Bio-inspired motion planning for reaching movement of a manipulator based on intrinsic tau jerk guidance, *Adv. Manuf.*, **7** (2019), 315–325. doi: 10.1007/s40436-019-00268-z.

22. Z. Zhang, R. He, K. Yang, A bioinspired path planning approach for mobile robots based on improved sparrow search algorithm, *Adv. Manuf.*, **1** (2021), 1–17. doi: 10.1007/s40436-021-00366-x.

23. K. Abainia, Y. M. B. Ali, Bio-inspired approach for inverse kinematics of 6-dof robot manipulator with obstacle avoidance, in *2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, (2018), 1–8. doi: 10.1109/PAIS.2018.8598489.

24. Z. Feng, L. Chen, C. Chen, M. Liu, M. Yuan, Motion planning for redundant robotic manipulators using a novel multi-group particle swarm optimization, *Evol. Intell.*, **13** (2020), 677–686. doi: 10.1007/s12065-020-00382-z.

25. S. Perez-Carabaza, E. Besada-Portas, J. A. Lopez-Orozco, M. Jesus, Ant colony optimization for multi-uav minimum time search in uncertain domains, *Appl. Soft Comput.*, **62** (2018), 789–806. doi: 10.1016/j.asoc.2017.09.009.

26. J. Huang, P. Hu, K. Wu, M. Zeng, Optimal time-jerk trajectory planning for industrial robots, *Mech. Mach. Theory*, **121** (2018), 530–544. doi: 10.1016/j.mechmachtheory.2017.11.006.

27. H. V. H. Ayala, L. D. S. Coelho, Tuning of pid controller based on a multiobjective genetic algorithm applied to a robotic manipulator, *Expert Syst. Appl.*, **39** (2012), 8968–8974. doi: 10.1016/j.eswa.2012.02.027.

28. S. Bureerat, N. Pholdee, T. Radpukdee, P. Jaroenapibal, Self-adaptive MRPBIL-DE for 6D robot multi-objective trajectory planning, *Expert Syst. Appl.*, **136** (2019), 133–144. doi: 10.1016/j.eswa.2019.06.033.

29. K. Passino, Biomimicry of bacterial foraging for distributed optimization and control, *IEEE Control Syst. Mag.*, **22** (2002), 52–67. doi: 10.1109/MCS.2002.1004010.

30. S. Dasgupta, S. Das, A. Abraham, A. Biswas, Adaptive computational chemotaxis in bacterial foraging optimization: An analysis, *IEEE Trans. Evolut. Comput.*, **13** (2009), 919–941. doi: 10.1109/TEVC.2009.2021982.

31. H. Chen, L. Wang, J. Di, S. Ping, Bacterial foraging optimization based on self-adaptive chemotaxis strategy, *Comput. Intel. Neurosc.*, **1** (2020), 1–15. doi: 10.1155/2020/2630104.

32. W. Cao, Y. Tian, M. Huang, Y. Luo, Adaptive bacterial foraging optimization based on roulette strategy, in *International Conference on Swarm Intelligence (ICSI)*, (2020). 1–8. doi: 10.1007/978-3-030-53956-6_27.

33. E. Ali, S. Abd-Elazim, Bacteria foraging optimization algorithm based load frequency controller for interconnected power system, *Int. J. Elec. Power*, **33** (2011), 633–638. doi: 10.1016/j.ijepes.2010.12.022.

34. S. Abd-Elazim, E. Ali, A hybrid particle swarm optimization and bacterial foraging for optimal power system stabilizers design, *Int. J. Elec. Power*, **46** (2013), 334–341. doi: 10.1016/j.ijepes.2012.10.047.

35. E. Ali, S. Abd-Elazim, BFOA based design of PID controller for two area load frequency control with nonlinearities, *Int. J. Elec. Power*, **51** (2013), 224–231. doi: 10.1016/j.ijepes.2013.02.030.

36. S. Panda, B. Mohanty, P. Hota, Hybrid BFOA–PSO algorithm for automatic generation control of linear and nonlinear interconnected power systems, *Appl. Soft Comput.*, **13** (2013), 4718–4730. doi: 10.1016/j.asoc.2013.07.021.

37. B. Huynh, S. Su, Y. Kuo, Vision/position hybrid control for a hexa robot using bacterial foraging optimization in real-time pose adjustment, *Symmetry*, **12** (2020), 1–20. doi: 10.3390/sym12040564.

38. Y. Long, Y. Su, B. Shi, Z. Zuo, J. Li, A multi-subpopulation bacterial foraging optimisation algorithm with deletion and immigration strategies for unmanned surface vehicle path planning, *Intel. Serv. Robot.*, **14** (2021), 303–312. doi: 10.1007/s11370-021-00361-y.

39. Y. Guan, K. Yokoi, O. Stasse, A. Kheddar, On robotic trajectory planning using polynomial interpolations, in *2005 IEEE International Conference on Robotics and Biomimetics - ROBIO*, (2005), 111–116. doi: 10.1109/ROBIO.2005.246411.

40. C. Lin, P. Chang, J. Luh, Formulation and optimization of cubic polynomial joint trajectories for industrial robots, *IEEE Trans. Automat. Control*, **28** (1983), 1066–1074. doi: 10.1109/TAC.1983.1103181.

41. S. Macfarlane, E. Croft, Design of jerk bounded trajectories for online industrial robot applications, in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (ICRA)*, (2001), 979–984. doi: 10.1109/ROBOT.2001.932677.

42. J. Carrasco, S. García, M. Rueda, S. Das, F. Herrera, Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: practical guidelines and a critical review, *Swarm Evol. Comput.*, **54** (2020), 1–20. doi: 10.1016/j.swevo.2020.100665.

43. J. Derrac, S. García, S. Hui, P. N. Suganthan, F. Herrera, Analyzing convergence performance of evolutionary algorithms: A statistical approach, *Inf. Sci.*, **289** (2014), 41–58. doi: 10.1016/j.ins.2014.06.009.

44. J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, *Swarm Evol. Comput.*, **1** (2011), 3–18. doi: 10.1016/j.swevo.2011.02.002.

45. S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization, *J. Heuristics*, **15** (2009), 617–644. doi: 10.1007/s10732-008-9080-4.

AIMS Press