IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 29, 2019, accepted October 18, 2019, date of publication October 24, 2019,
date of current version November 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2949271

A New Trajectory-Planning Beetle Swarm Optimization Algorithm for Trajectory Planning of Robot Manipulators

LEI WANG[1], QING WU[1], FEI LIN[1], SHUAI LI[2], (Senior Member, IEEE), AND DECHAO CHEN[1], (Member, IEEE)

[1]Department of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China
[2]Department of Computing, The Hong Kong Polytechnic University, Hong Kong

Corresponding authors: Shuai Li (shuaili@polyu.edu.hk) and Dechao Chen (chdchao@hdu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61906054, Grant 61401385, and Grant 61702146, in part by the Hong Kong Research Grants Council Early Career Scheme under Grant 25214015, in part by Departmental General Research Fund of Hong Kong Polytechnic University under Grant G.61.37.UA7L, and in part by PolyU Central Research under Grant G-YBMU.

ABSTRACT Based on a heuristic optimization algorithm, this paper proposes a new algorithm named trajectory-planning beetle swarm optimization (TPBSO) algorithm for solving trajectory planning of robots, especially robot manipulators. Firstly, two specific manipulator trajectory planning problems are presented as the practical application of the algorithm, which are point-to-point planning and fixed-geometric-path planning. Then, in order to verify the effectiveness of the algorithm, this paper develops a control model and conducts numerical experiments on two planning tasks. Moreover, it compares with existing algorithms to show the superiority of our proposed algorithm. Finally, the results of numerical comparisons show that algorithm has a relatively faster computational speed and better control performance without increasing computational complexity.

INDEX TERMS Beetle swarm optimization, robot manipulators, trajectory planning, optimization algorithms, control systems.

I. INTRODUCTION

The robotic arm, or the manipulator, has similar functions to a human arm and can be either a separate mechanism or part of a more complex robot [1]. It can be considered that the link of the robot arm forms a kinematic chain, and the end of the kinematic chain is called an end-effector for performing actual operations such as grasping, spraying, cutting, etc. The robotic arm is widely used in modern society. In the field of industrial manufacturing, it is used for assembly, spraying, welding etc [2]. In the medical field, it is used as a surgical aid [3], [4]. And in agriculture, for picking vegetables and fruits. Even in space exploration the robotic arm can also find its application.

In the research area of robot manipulator, trajectory planning has always been a hot spot, which is usually performed with constraints that may come from dynamic equations or from the inputs [5]. There are several important issues in the study of manipulator trajectory planning, one of which is the solution of inverse kinematics transformation [6]–[10]. Due to the high nonlinearity of inverse kinematics transformation, the solution process is difficult. Therefore, improving the efficiency and effectiveness of inverse kinematics is very important in manipulator trajectory planning, especially in redundant manipulator trajectory planning [11]–[18]. The second is obstacle avoidance. In many manufacturing applications today, robot manipulators must use their end-effector to pass through the desired curve while their fuselage avoids collisions with obstacles in the environment. This problem is very complicated, however, researchers have proposed some relatively effective solutions. In addition, on-line algorithms for continuous paths [19], [20], or tracking control [21]–[24], can be found in robotic applications such as welding, painting process, and part assembly.

The original study used traditional planning algorithms to solve such problems, however traditional optimization techniques have many shortcomings, such as couldn't find global solutions, requiring gradients, not for discontinuous functions, etc. Therefore, many heuristic and meta heuristic algorithms, such as simulated annealing (SA) [25], [26], genetic algorithm (GA), particle swarm optimization (PSO), ant colony optimization (ACO), Teaching-learning-based

The associate editor coordinating the review of this manuscript and approving it for publication was Emanuele Crisostomi.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

optimization (TLBO) [27] etc. have been applied to trajectory planning.

Bio-heuristic algorithms are the methods to solve problems learn from the survival skills of organisms. Like other heuristic algorithms, bio-heuristics are used to solve some of the most time-consuming problems [28]–[30] that traditional algorithms are too slow to solve, or to find an approximate solution when traditional algorithms cannot find any exact solution [31]. The evolution of organisms under natural selection reflects good adaptability, which makes the effectiveness of these algorithms guaranteed.

A. RELATED WORK

Some optimization indicators are usually set during trajectory planning. Generally, these Zanotto Zanotto indicators can be divided into the following three categories: minimum execution time, minimum energy [32], and minimum jerk [33], [34]. Chen and Desrochers [35] addressed the problem of the structure of minimum-time control of robotic manipulators along a specified geometric path. Faris *et al*. [36] proposed an energy minimization approach for a two-link flexible manipulator. Piazzi and Visioli [37] proposed a new approach based on interval analysis to find the global minimum-jerk trajectory of a robot manipulator within a joint space scheme using cubic splines. Of course, there are also many kind of researches use hybrid optimality criteria in practical planning [38]. Gasparetto and Zanotto [39] proposed a technique for time-jerk optimal planning of robot trajectories.

In recent years, more and more researchers began using bio-heuristic algorithms for trajectory planning. GA is used for trajectory planning of mobile robots and manipulators, including collision-free path planning, dynamic constrained multi-path planning and redundant robots planning [40]. ACO can be used to mobile and articulated robots as well as multi-robot systems in robot trajectory planning. As a broad and effective algorithm, PSO is also used in many obstacle avoidance trajectory planning, for multi-objective optimization [41], soccer robot path planning [42], and mobile robot smooth path planning [43]. In addition, ABC is used for multi-robot path planning [44] from the beginning to the target position. Savsani *et al.* [45] use artificial bee colony algorithm (ABC) and TLBO for trajectory optimization of a three-axis (3R) planar manipulator, where TLBO performed better than ABC.

B. ORGANIZATION AND CONTRIBUTIONS

The structure of this paper is as follows. In Section II we introduce the problem we are trying to solve as well as some related concepts. In Section III, we will present specific planning methodology, including the definition of cost function and the description of algorithm TPBSO we proposed. In Section IV, we apply the TPBSO algorithm to two types of manipulator trajectory planning problems and compare the results with other algorithms. Finally, in Section V, we summarize the content of the full paper. The main contributions of this paper are listed as follows.

- In this paper, a new trajectory planning algorithm termed TPBSO was proposed for the first time. Compared with traditional algorithms, it inherently has the characteristics of low computational complexity and fast optimization speed;
- The BSO algorithm is used to optimize the control parameters in trajectory planning problems of robot manipulators;
- The proposed TPBSO algorithm is successfully applied to two specific types of planning problem and compare with the existing algorithms to verify its superiority.

II. PRELIMINARIES

Path planning and trajectory planning are the focus of research in the field of robotics, and there is a difference between these two terms [1]. Path planning simply generates a geometric path without applying a specific motion rule, while trajectory planning requires a motion rule to be specified for the geometric path. Therefore, trajectory planning is usually done after path planning, one planning problem we will test in section III is just like that. But sometimes, these two steps can be combined, such as point-to-point planning which is another planning problem we will test in section III.

Next, we will introduce some important concepts that are used in trajectory planning:

- **Configuration**: Configuration is used to indicate the position and orientation of the robot relative to the fixed reference frame F_w. For mobile robots and end-effectors, the configuration is their position and orientation in a Cartesian coordinate system. But for manipulators, the configuration consists of their joint rotation angles.
- **Configuration Space** (C-space): C-space refers to the space formed by all possible configurations of the robot, and trajectory planning is performed in this space. It can be divided into the space of free configurations (C-free) and the space of obstacles (C-obs).
- **Degrees of Freedom** (DOF): For arm robots, the degrees of freedom refers to the number of joints it has. In three-dimensional space, the robot requires at least 6 degrees of freedom: 3 rotational degrees and 3 translational degrees. When the robot has more joints than it needs, we say that the extra degrees are redundant.
- **Kinematic transformation**: The kinematic transformation is divided into forward transformation and inverse transformation. The forward transformation problem is to find the position of the end-effector in Cartesian space when the joint angles are known, and inverse transformation is just the opposite. Because of its high nonlinearity, the inverse kinematic transformation is usually more difficult to resolve.

In this paper, we use the plane geometry to solve inverse kinematics transformation. The specific method in the case of the two-link planar manipulator is as follows:

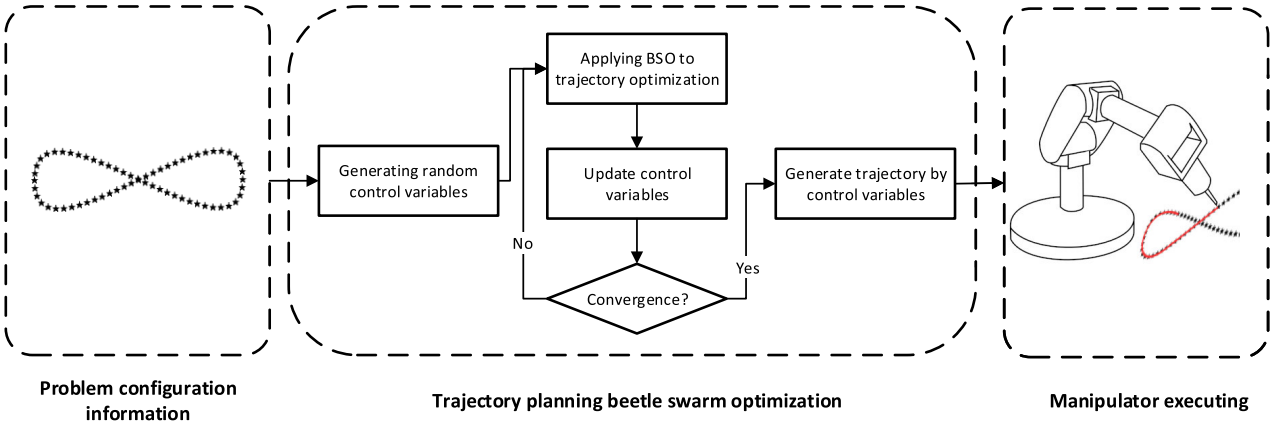


FIGURE 1. Flowchart of TPBSO algorithm applied to trajectory planning of robot manipulator.

Available from the cosine theorem, we have

$$\cos(\theta_2) = \frac{\|p_e\|^2 - l_1^2 - l_2^2}{2l_1 l_2}, \tag{1}$$

where θ_2 represents the joint angle of the second joint, p_e is the coordinate of the end-effector, and l_1 and l_2 are the lengths of the two links, respectively. If $\cos(\theta_2) > 1$, this triangle does not exist, and the coordinate of p_e is beyond the reach of the arm. This should be avoided.

Then we can obtain the joint angle of the first joint

$$\cos(\theta_1) = \frac{p_e \cdot p_e'}{\|p_e\|^2}, \tag{2}$$

where θ_1 represents the joint angle of the first joint, and p_e' is the coordinate of the end-effector when this joint is turned back to the θ_1 angle to make the first link horizontal.

The trajectory planning means to generate control signals that are passed to the control system to ensure that the robot performs the required motion. The conduction structure of the control signal can be either hierarchical [46] or distributed [47], [48]. Typically, the algorithm used for trajectory planning takes the path generated by path planner, as well as kinematic and dynamic constraints as input. Then it provides a set of position, velocity values to joints or end-effector as output.

In order to plan the trajectory in joint space, a set of points should be extracted from the path of end-effector that generated by the path planner. This set of points is then mapped to the joint space by inverse kinematic transformation. The joint space is used as the configuration space, and considering the kinematic and dynamic constraints on each joint, the trajectory planning is performed. A trajectory with sufficient precision is then generated by means of interpolation functions.

III. METHODOLOGY

As shown in FIGURE 1, we will present a method for manipulator trajectory planning in this section. First, for a specific problem, we find some representative variables as control parameters and use them as input to TPBSO algorithm. Then, the algorithm starts its iteration, and each time the cost function is used as a basis to generate better control parameters until a certain number of iterations is reached. Finally, we use the generated control parameters to build a complete trajectory.

The following content will be divided into three parts to illustrate the method. Subsection III-A introduces the overview method and control parameters of two robot trajectory planning problems. Subsection III-B describes the meaning and structure of the cost function. Subsection III-C gives the algorithm framework of TPBSO.

A. OVERVIEW

The TPBSO algorithm is used to deal with two types of trajectory planning problems: point-to-point planning and along-a-specified-path planning. Further, we classify the first type of problem into two situations with or without obstacle.

In the first case, the positions of initial point and final point are given, the joint speed and acceleration of them are both set to zero. We have two optimization indicators when without obstacle, the minimum joint rotation and the minimum execution time. And for the situation with obstacle, we use the minimum travel length in workspace as indicator. In order to find the optimal trajectory under this problem, we use the traditional polynomial time function to describe its motion behavior. We set a intermediate point of a path as a control point. During the movement, the manipulator starts from the initial point, passes through the intermediate point, and then reaches the final point.

This intermediate point divides the path into two. For the first segment, because there are five constraints: starting position $\boldsymbol{\theta_s}$, starting speed $\dot{\boldsymbol{\theta}}_s$, starting acceleration $\ddot{\boldsymbol{\theta}}_s$, intermediate point position $\boldsymbol{\theta_m}$ and intermediate point speed $\dot{\boldsymbol{\theta}}_m$, we have combined five equations and used a fourth-order polynomial as its motion equation.

The first segment, from $\boldsymbol{\theta_s}$ to $\boldsymbol{\theta_m}$, is expressed as a fourth-order polynomial time function:

$$\boldsymbol{\theta}(t) = \mathbf{a_0} + \mathbf{a_1}t + \mathbf{a_2}t^2 + \mathbf{a_3}t^3 + \mathbf{a_4}t^4. \tag{3}$$

Substituting the boundary conditions, and we can get the solution of coefficients as follows:

$$\begin{cases} \mathbf{a_0} = \boldsymbol{\theta}_s, \\ \mathbf{a_1} = \mathbf{0}, \\ \mathbf{a_2} = \mathbf{0}, \\ \mathbf{a_3} = \dfrac{4\boldsymbol{\theta}_m - \dot{\boldsymbol{\theta}}_m t_1 - 4\boldsymbol{\theta}_s}{t_1^3}, \\ \mathbf{a_4} = \dfrac{\dot{\boldsymbol{\theta}}_m t_1 - 3\boldsymbol{\theta}_m + 3\boldsymbol{\theta}_s}{t_1^4}. \end{cases} \tag{4}$$

After determining the time function for the first segment of trajectory, we can find the acceleration when manipulator passes the intermediate point. So in the second segment we have six constraints and a quintic polynomial is used as its equation of motion.

The time function from $\boldsymbol{\theta}_m$ to $\boldsymbol{\theta}_f$:

$$\boldsymbol{\theta}(t) = \mathbf{a_0} + \mathbf{a_1}t + \mathbf{a_2}t^2 + \mathbf{a_3}t^3 + \mathbf{a_4}t^4 + \mathbf{a_5}t^5. \tag{5}$$

The solution of coefficients:

$$\begin{cases} \mathbf{a_0} = \boldsymbol{\theta}_m, \\ \mathbf{a_1} = \dot{\boldsymbol{\theta}}_m, \\ \mathbf{a_2} = \dfrac{\ddot{\boldsymbol{\theta}}_m}{2}, \\ \mathbf{a_3} = \dfrac{20\boldsymbol{\theta}_f - 20\boldsymbol{\theta}_m - 12\dot{\boldsymbol{\theta}}_m t_2 - 3\ddot{\boldsymbol{\theta}}_m t_2^2}{2t_2^3}, \\ \mathbf{a_4} = \dfrac{30\boldsymbol{\theta}_m - 30\boldsymbol{\theta}_f + 16\dot{\boldsymbol{\theta}}_m t_2 + 3\ddot{\boldsymbol{\theta}}_m t_2^2}{2t_2^4}, \\ \mathbf{a_5} = \dfrac{12\boldsymbol{\theta}_f - 12\boldsymbol{\theta}_m - 6\dot{\boldsymbol{\theta}}_m t_2 - \ddot{\boldsymbol{\theta}}_m t_2^2}{2t_2^5}. \end{cases} \tag{6}$$

The whole trajectory planning cannot be completed if just the time function of motion is known. In order to relate the time parameter t in motion equation (3) and (5) to the actual time, the total time which the manipulator is executing in reality is also required. After obtaining the total execution time, the actual speed and acceleration of manipulator's each joint can be calculated at any time during the movement. It is necessary to limit the speed and acceleration to meet the practical significance, the detailed constraints will be discussed in section III-B.

In addition, in this problem, we originally used a planar two-link manipulator to plan the trajectory between two points on the plane, but it was too easy. In order to increase the difficulty of planning, we have increased the number of links. However, the coordinates of the two points on the plane are still only two parameters, so the manipulator has extra degrees of freedom.

In the second case, we gave the information about the entire path in advance, and then the end-effector of the manipulator needs to move along the given path. Here, we use a set of ordered points to represent the path, and the end-effector follows the path through these points in turn. In addition, we use a planar three-link manipulator to plan, so the manipulator has one redundant degree of freedom, which also makes one of the optimization indicators to minimize the joint rotation has a planning significance.

The core of solving this planning problem is the determination of the time function of the redundant parameter. Once the time function of the redundant parameter is determined, the rotation angle of each joint when the manipulator passes each point on the path can be obtained, according to the coordinates of each point on the given path and using the forward kinematics transformation. After that, we can get the rotation angle of each joint at any time by interpolation. Here, the redundant parameter has a value range of $[-\pi, \pi]$, but for geometric reasons, when the end-effector is in different positions, the range of its value is different. The time function of the redundant parameter's range can be obtained by the geometric method. The time function of the redundant parameter must be within this range.

To reduce the computational cost, we do not use all the points on the input path, but take part of them, and generate the trajectory between these points through polynomial interpolation in the joint space. Therefore, the generated trajectory does not strictly pass through all points on the input path. In order to reduce the difference between the planned trajectory and the original path, we need to add a penalty term named trajectory accuracy in the cost function. We will discuss this more formally in section III-B.

As with the first problem, we also need information about the real execution time of the manipulator. Since a given path can be viewed as multi-point-to-point planning, we can specify the execution time for every interval. However, this leads to an unnecessarily high dimension of the solution vector. Since the actual time of each movement is not much different, we still only use the total execution time t_T and the execution time of each segment is equal.

From two types of problems above, although the form is different, we can find two types of parameters:

One type is called the configuration parameter, given at the beginning of each problem, as an input variable. In the first case, the positions of initial point $\boldsymbol{\theta}_s$ and final point $\boldsymbol{\theta}_f$ are configuration parameters; in the second case, They are path points.

The other type, called the control parameter, determines the trajectory of the manipulator under the conditions given by the configuration parameters, and is also parameters that need to be optimized. In the first case, control parameters are the execution time t_1 and t_2, the position and speed of the intermediate point $\boldsymbol{\theta}_m$ and $\dot{\boldsymbol{\theta}}_m$ which including the rotation angle and rotation speed of each joint. In addition, when the manipulator has redundant degrees of freedom, redundant parameters also serves as control parameters; In the second case, control parameters are the total execution time t_T and redundant parameters when the end-effector passes through various points. Since there are too many redundant parameters corresponding to each point, we only take 5 variables and then generate the remaining values by interpolation.

We need to put all control parameters into a solution vector when optimizing. The structure of the solution vector in the first problem is as follows, where k is the number of links:

- k variables, the joint angles in joint space when passing the intermediate point $\boldsymbol{\theta_m}$;
- k variables, the joint speeds in joint space when passing the intermediate point $\dot{\boldsymbol{\theta}}_{\boldsymbol{m}}$;
- $2(k-2)$ variables, redundant parameters ϕ_s at the initial point and ϕ_f at the final point when $k > 2$;
- 2 variables, execution time t_1 in the first segment and t_2 in the second segment.

The structure of solution vector in the second problem is as follows:

- n variables, with any value in the interval [0,1], x_ϕ;
- 1 variable, total execution time t_T.

The n values within range [0,1] above are the normalized redundant sequence x_ϕ. As we discussed before, n should be as small as possible to reduce the dimension, then more precise sequences can be generated by interpolation. Since the values of the redundant sequence x_ϕ are normalized, we still need to restore them to real value in the end.

$$\phi = x_\phi(\phi_{\max} - \phi_{\min}) + \phi_{\min}. \tag{7}$$

Equation (7) restores a normalized x_ϕ value to the corresponding ϕ value.

B. COST FUNCTION

In this section, we present a planning model and cost function for the two problems. As mentioned in the previous section, we have defined two optimization indicators, the minimum joint rotation and the minimum execution time, for both two problems. Therefore, for the first problem, point-to-point planning, when minimum execution time is used as optimization indicator, the goal of the planning model is:

$$\min T = t_1 + t_2. \tag{8}$$

And when minimum joint rotation is used as optimization indicator, the goal of the planning model is:

$$\min F_q = \sum_{i=1}^{n}\sum_{j=1}^{m-1} \Delta\theta_{ij}, \tag{9}$$

where n is the joint number, m is the number of segments of the total execution time, and $\Delta\theta_{ij}$ is the angular displacement in the j^{th} time segment of the i^{th} joint. We discretize time and then sum up the angles of each joint for each small period of time.

When in obstacle avoidance situation, minimum travel length in workspace is used as optimization indicator, the goal of the planning model is:

$$\min F_d = \sum_{i=1}^{n}\sum_{j=1}^{m-1} \Delta\kappa(\theta_{ij}), \tag{10}$$

where $\kappa(\cdot)$ represents the forward kinematics transformation of the manipulator. We discretize time and then sum up the displacement of end-effector in workspace for each small period of time.

Despite the speed and acceleration limits of joints, the limitation of redundant parameters is also included in the constraints of this planning model when manipulator has redundant degrees of freedom. This is because some values of the redundant parameters are not legal, and it is impossible to use them for inverse kinematic transformation to calculate joint angles.

Joint speed limits can be set as follows:

$$\begin{cases} \forall t \in [0, t_1 + t_2], \\ |\dot{\theta}_i(t)| \leqslant \dot{\theta}_{\max}. \end{cases} \tag{11}$$

Joint acceleration limits can be set as follows:

$$\begin{cases} \forall t \in [0, t_1 + t_2], \\ |\ddot{\theta}_i(t)| \leqslant \ddot{\theta}_{\max}. \end{cases} \tag{12}$$

Restrictions on the initial point redundant parameters ϕ_s and final point redundant parameters ϕ_f:

$$\begin{cases} k_s \leqslant 1, \\ k_f \leqslant 1, \end{cases} \tag{13}$$

where k_s, k_f represent the value of $\cos(\theta_2)$ calculated by ϕ_s, ϕ_f from (1) and (2).

When obstacles appear, restrictions of obstacle avoidance can be set as follows:

$$\begin{cases} \forall t \in [0, t_1 + t_2], \\ l_i(t) \bigcap \text{Obs} = \emptyset, \end{cases} \tag{14}$$

where $l_i(t)$ represents the point set of i^{th} manipulator link at time t, and Obs represents the point set of Obstacle's surface.

Thus, according to the planning model we propose the following cost function:

$$F(x) = \begin{cases} f(x) + \lambda_1 \sum_{i=1}^{3} h_i(x) g_i(x), & k_s, k_f \leqslant 1, \\ \lambda_2(k_s + k_f), & \text{otherwise}, \end{cases} \tag{15}$$

where $f(\cdot)$ is the objective function of the equation (8), (9) or (10), $g_1(\cdot)$ and $g_2(\cdot)$ respectively calculate the sum of the speed and the acceleration of each joint at each time interval according to (11) and (12), $g_3(\cdot)$ calculate the number of lapped points according to (14). And functions $h_1(\cdot)$ to $h_3(\cdot)$ denote respectively as

$$h_1(x) = \begin{cases} 1, & |\dot{\theta}_i(t)| > \dot{\theta}_{\max}, \\ 0, & |\dot{\theta}_i(t)| \leqslant \dot{\theta}_{\max}, \end{cases} \tag{16}$$

$$h_2(x) = \begin{cases} 1, & |\ddot{\theta}_i(t)| > \ddot{\theta}_{\max}, \\ 0, & |\ddot{\theta}_i(t)| \leqslant \ddot{\theta}_{\max}. \end{cases} \tag{17}$$

$$h_3(x) = \begin{cases} 1, & l_i(t) \bigcap \text{Obs} \neq \emptyset, \\ 0, & \text{otherwise}. \end{cases} \tag{18}$$

For the second problem, trajectory planning in different specified path, when minimum execution time is used as optimization indicator, the goal of the planning model is:

$$\min T = t_T. \tag{19}$$

And the goal for minimum joint rotation indicator, just the same as we proposed in the first problem:

$$\min F_q = \sum_{i=1}^{n} \sum_{j=1}^{m-1} \Delta\theta_{ij}. \tag{20}$$

This planning model is constrained by the redundant parameter limit, the joint speed limit, and the joint acceleration limit.

Joint speed limit can be set as follows:

$$\begin{cases} \forall t \in [0, t_T], \\ |\dot{\theta}_i(t)| \leqslant \dot{\theta}_{\max}. \end{cases} \tag{21}$$

Joint acceleration limit can be set as follows:

$$\begin{cases} \forall t \in [0, t_T], \\ |\ddot{\theta}_i(t)| \leqslant \ddot{\theta}_{\max}. \end{cases} \tag{22}$$

Redundant parameter limit can be set as follows:

$$\begin{cases} \forall t \in [0, t_T], \\ \phi(t)_{\min} \leqslant \phi(t) \leqslant \phi(t)_{\max}, \end{cases} \tag{23}$$

where $\phi(t)$ represents the value of redundant parameter at time t, $\phi(t)_{\min}$ represents the lower bound of redundant parameter and $\phi(t)_{\max}$ is the upper bound.

Path accuracy limit can be set as follows:

$$\begin{cases} \forall i \in [1, n], \\ \|P^*(i) - P(i)\| \leqslant \text{accuracy}, \end{cases} \tag{24}$$

where P^* represents the point set of target path and P is the point set of our planned trajectory, accuracy represents the maximum distance of two compared point. The point set P^* and P both have n points and generated by normalizing the original trajectory points, so that they are comparable.

Again, we propose the following cost function where $f(\cdot)$, $g_1(\cdot)$ and $g_2(\cdot)$ is the same meaning in the first problem, $g_3(\cdot)$ calculate the sum of the distance of each pair of points in planned trajectory and target path according to (24):

$$F(x) = f(x) + \sum_{i=1}^{3} h_i(x) g_i(x). \tag{25}$$

Notice that the constraint of redundant parameter is not applied to the cost function, that's because we normalized the redundant parameter sequence in advance, and the restored redundant parameter values must be within their range. Through this way, the constraint in (23) is already satisfied.

C. ALGORITHM FRAMEWORK

In this paper, a new heuristic optimization algorithm called BSO is used for trajectory planning. This algorithm was first proposed in [49]. It combines the PSO algorithm with another new heuristic algorithm BAS, which is being applied to various practical problems [50], [51]. In this algorithm, the optimization process is considered as a process of exploring a group of beetles in the solution space. The position of each beetle represents a solution to the problem. The BSO algorithm has good robustness and speed. In addition, it also exhibits higher performance when dealing with nonlinear constraints. Therefore, we use it for trajectory planning problem optimization and name the algorithm TPBSO.

Next we will introduce the specific process of the TPBSO algorithm. First, we randomly generate the initial position x of each beetle which represents the solution vector and calculate its fitness value $F(x)$. The fitness value is calculated according to the cost function we mentioned in section III-B, the first planning problem corresponds to the equation (15), and the second planning problem corresponds to the equation (25).

Then we generate the next exploration position for each beetle according to the following formula

$$x^{t+1} = x^t + \lambda V^t + (1 - \lambda)\xi^t, \tag{26}$$

where t is the number of iterations, V^t derived from the PSO algorithm is the speed of the beetle at the t-th iteration, and ξ^t derived from the BAS algorithm represents the increment of the beetle's displacement along the velocity direction at the t-th iteration, λ controls the ratio of the former two.

$$V^{t+1} = V^t + c_1 r_1 (P^t_{\text{is}} - x^t) + c_2 r_2 (P^t_{\text{gs}} - x^t). \tag{27}$$

Equation (27) denotes the content of V^t in (26). Where c_1 and c_2 are constants called learning factor, r_1 and r_2 are random variables within the range of [0,1], P_{is} is the individual optimal position, P_{gs} is the global optimal position.

$$\xi^{t+1} = \delta^t V^t \text{sign}(F(x^t_{\text{rt}}) - F(x^t_{\text{lt}})). \tag{28}$$

Equation (28) denotes the content of ξ^t in (26). Where δ^t indicates step size at the t-th iteration, x^t_{rt} and x^t_{lt} can be expressed as

$$x^t_{\text{rt}} = x^t + \frac{V^t d^t}{2}, \tag{29}$$

and

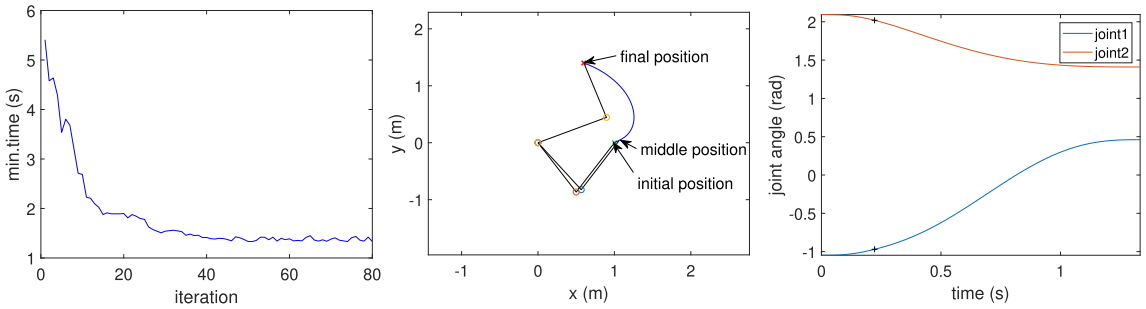
$$x^t_{\text{lt}} = x^t - \frac{V^t d^t}{2}, \tag{30}$$

where

$$d^t = \frac{\delta^t}{c}. \tag{31}$$

The step size of the beetle is proportional to the length of the antenna, c is its scale factor, and the length of each step is reduced in each iteration:

$$\delta^t = \eta \delta^{t-1}. \tag{32}$$



(a) Execution time varies with iteration (b) The planned path with manipulator (c) Joint angles varies with time

FIGURE 2. Minimum execution time trajectory planning in 2-link situation.

Algorithm 1 TPBSO Algorithm for Trajectory Planning

input: configuration parameters mentioned in section III-A
output: $t_{\min}$ or $\theta_{\min}$, $\vec{j}_{\text{out}}$

Initialize $m, n, x_i, V_i, \delta_0, c, \eta$, velocity range $v_{\min}$ and $v_{\max}$
Update the fitness of each beetle
Update x_{pbest} and x_{gbest}
for $p = 1$ to n **do**
  for all x such that $x \in X$ **do**
    Update V according to (27)
    Calculate $F(x_{\text{rs}})$ and $F(x_{\text{ls}})$ according to (29) and (30)
    Update ξ according to (28)
    Update x according to (26)
  end for
  Update the fitness of each beetle
  Update x_{pbest} and x_{gbest}
  for $i = 1$ to m **do**
    if $F(x_i) < F_{\text{pbest}}$ **then**
      $F_{\text{pbest}} \leftarrow F(x_i)$
    end if
    if $F(x_i) < F_{\text{gbest}}$ **then**
      $F_{\text{gbest}} \leftarrow F(x_i)$
    end if
  end for
  Update step size δ according to equation (32)
end for
Generate $\vec{j}_{\text{out}}$, $t_{\min}$, $\theta_{\min}$ according to x_{gbest}

In order to describe the TPBSO algorithm more clearly, we write the detailed steps as pseudo-code in Algorithm 1. The variables and steps are explained as follows: $t_{\min}$ and $\theta_{\min}$ represent the result of two optimization indicator, minimum execution time and minimum joint rotation. $\vec{j}_{\text{out}}$ represents the planning trajectory. m and n represent the population size and the number of iterations respectively. Initial beetle group position $x_i(i = 1, 2, ..., m)$ and velocity $V_i(i = 1, 2, ..., m)$ are random. δ_0 represents initial step size, x_{pbest} and x_{gbest} represent the individual optimal solutions and global optimal solution. The remainder of the variables is the same as mentioned above.

IV. SIMULATION STUDIES AND COMPARISONS

In this section, we will apply the proposed TPBSO algorithm in some robot trajectory planning applications. In Subsection IV-A, we apply TPBSO algorithm to the point-to-point trajectory planning which is the simplest scenario in the real world. In Subsection IV-B, we apply TPBSO algorithm to a more complex scenario of trajectory planning in different specified path. In Subsection IV-C, we apply the proposed TPBSO algorithm, the conventional GA and PSO algorithm in the same environments and conduct comparative simulations.

A. POINT-TO-POINT TRAJECTORY PLANNING

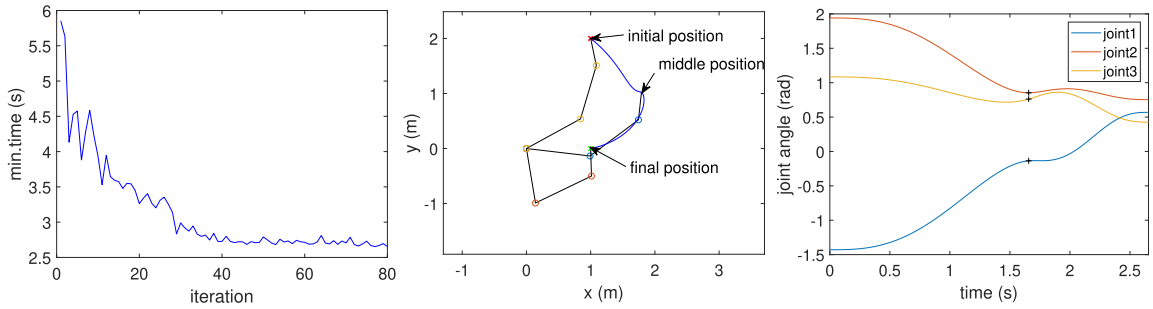
In this subsection, we divide the point-to-point trajectory planning into two different optimality criteria: minimum execution time and minimum joint rotation. Since this experiment is performed in an obstacle-free environment, to increase its work's difficulty, we have increased the number of links more than the original two links.

1) MINIMUM EXECUTION TIME

In this subsection, we take minimum execution time as our optimality criteria, and we set up four situations for simulation, from 2-link to 5-link differed from the number of links.

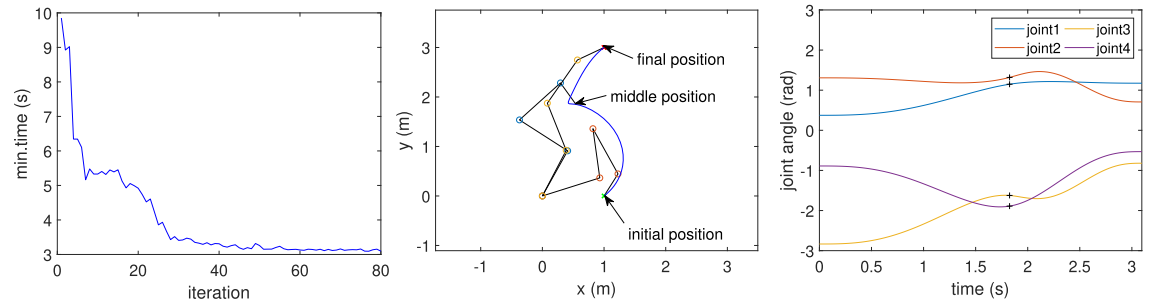
FIGURE 2 to FIGURE 9 have the same organization: a trend of optimal fitness value and a path plan figure.

Figures such as FIGURE (2b) are a simple presentation of the manipulator and the planned path. Links of the manipulator are represented by the straight line in figure, and joints are represented by ''o'' in the junction of two links. This figure shows three snapshots of the manipulator when it's at initial position, middle position and final position. We give two positions: initial position in the coordinate of (1,0) and final position in the coordinate of (0.5,1.5). The middle point that to be optimized is the control point of the path. The curve with blue colour represent the planned path which from initial position to final position.



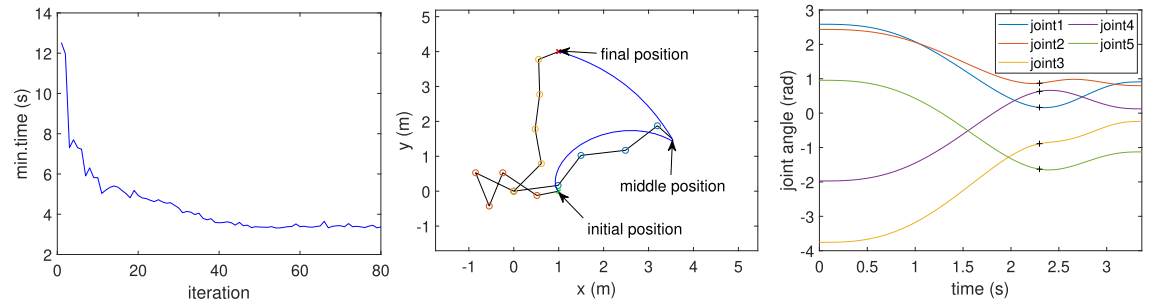
(a) Execution time varies with iteration (b) The planned path with manipulator (c) Joint angles varies with time

FIGURE 3. Minimum execution time trajectory planning in 3-link situation.



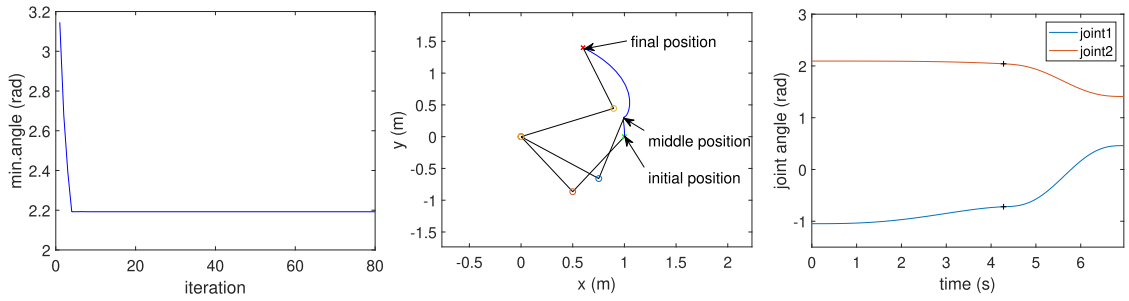
(a) Execution time varies with iteration (b) The planned path with manipulator (c) Joint angles varies with time

FIGURE 4. Minimum execution time trajectory planning in 4-link situation.



(a) Execution time varies with iteration (b) The planned path with manipulator (c) Joint angles varies with time

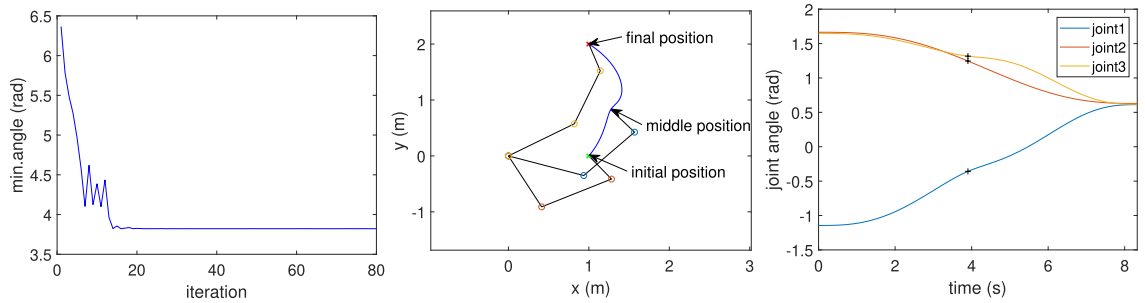
FIGURE 5. Minimum execution time trajectory planning in 5-link situation.



(a) Joint rotation varies with iteration (b) The planned path with manipulator (c) Joint angles varies with time

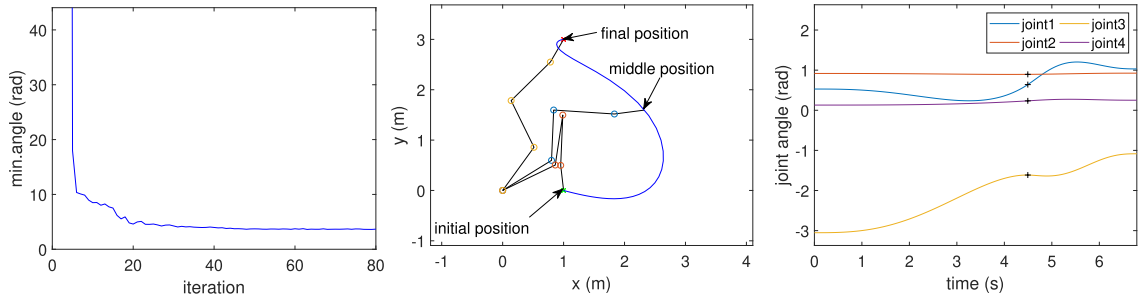
FIGURE 6. Minimum joint rotation trajectory planning in 2-link situation.

Figures like FIGURE (2a) present the execution time of the manipulator varies with the number of iterations, and it shows that the number of iterations required for the algorithm to converge. The simulation result in FIGURE (2c) present the movement status of each joint.



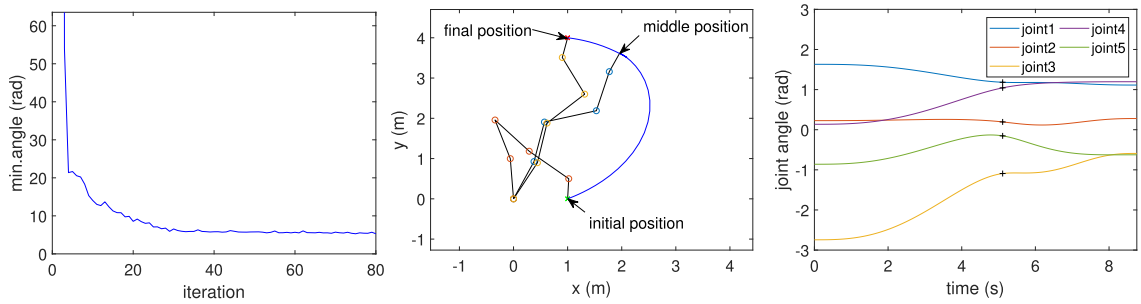
(a) Joint rotation varies with iteration (b) The planned path with manipulator (c) Joint angles varies with time

FIGURE 7. Minimum joint rotation trajectory planning in 3-link situation.



(a) Joint rotation varies with iteration (b) The planned path with manipulator (c) Joint angles varies with time

FIGURE 8. Minimum joint rotation trajectory planning in 4-link situation.



(a) Joint rotation varies with iteration (b) The planned path with manipulator (c) Joint angles varies with time

FIGURE 9. Minimum joint rotation trajectory planning in 5-link situation.

Through this series of experiments, we can find that as the number of links grows, the TPBSO can converge well.

2) MINIMUM JOINT ROTATION

In this subsection, we take minimum joint rotation as our optimality criteria. Like we mentioned in section IV-A.1, we set four situations from 2-link to 5-link to test our algorithm's performance, and the organization of experiment figures are the same.

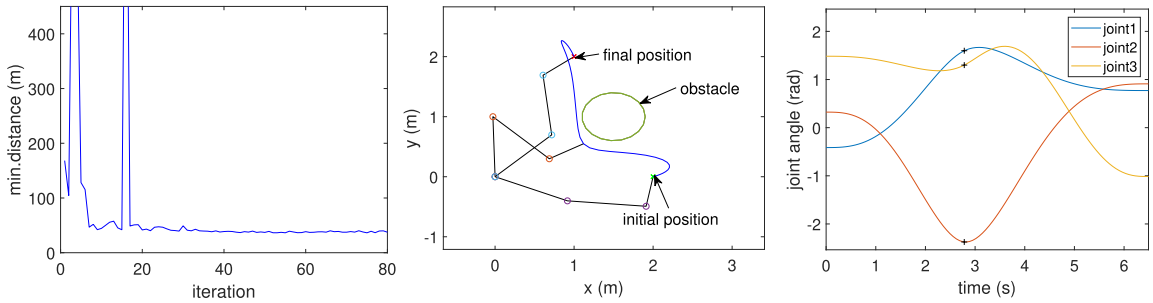
As shown in FIGURE 6(a) which is under the situation of 2-link, the number of iterations required for algorithm convergence is very small. This is because this task in 2-link situation is pretty simple and the algorithm can easily find the optimal solution in the solution space.

B. POINT-TO-POINT TRAJECTORY PLANNING WITH OBSTACLE AVOIDANCE

This subsection is an extension of the previous point-to-point trajectory planning. To make the simulation more complex, we added an oval obstacle between the two points in the workspace. Unlike the situation of obstacle-free, we chose the minimum travel distance in workspace as its optimality criteria. Like the previous task, manipulators with different link number also used for simulation.

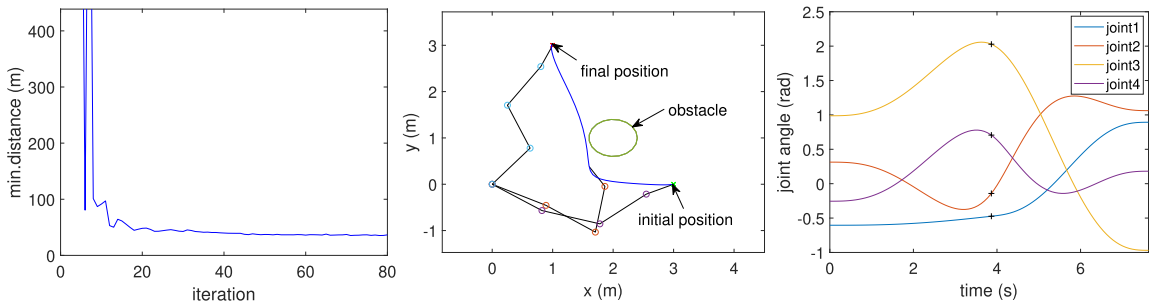
C. TRAJECTORY PLANNING IN DIFFERENT SPECIFIED PATH

In this subsection, we test the TPBSO algorithm for the scenario with two different paths: pentagram path and closed



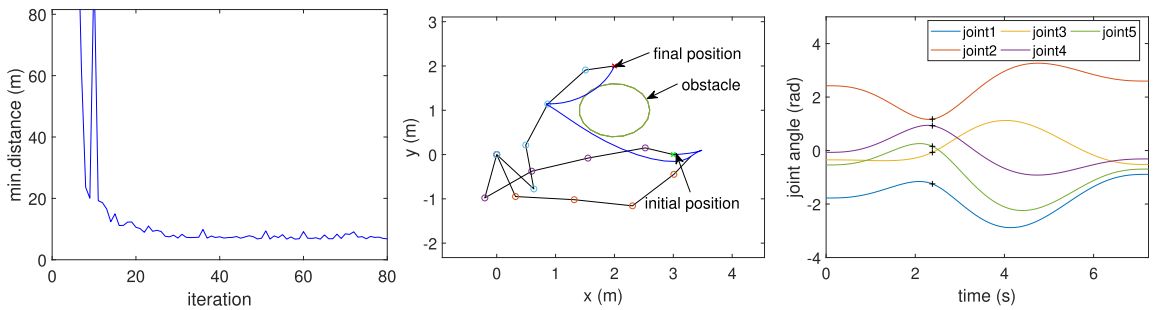
(a) Execution time varies with iteration (b) The planned path with manipulator (c) Joint angles varies with time

FIGURE 10. Obstacle avoidance trajectory planning in 3-link situation.



(a) Execution time varies with iteration (b) The planned path with manipulator (c) Joint angles varies with time

FIGURE 11. Obstacle avoidance trajectory planning in 4-link situation.



(a) Execution time varies with iteration (b) The planned path with manipulator (c) Joint angles varies with time

FIGURE 12. Obstacle avoidance trajectory planning in 5-link situation.

curve path. These two paths represent two types of and smooth curve. To represent the two paths, we sampled 50 points of each and stored the point set by order.

1) IN PENTAGRAM

In this subsection, the pentagram path is chosen for test purpose. We also made two sets of experiments with minimum execution time criteria and minimum joint rotation criteria.

Figures shown in this section has the same organization: a trend of optimal fitness value, a path plan figure and a planning figure of the redundant parameter ϕ.

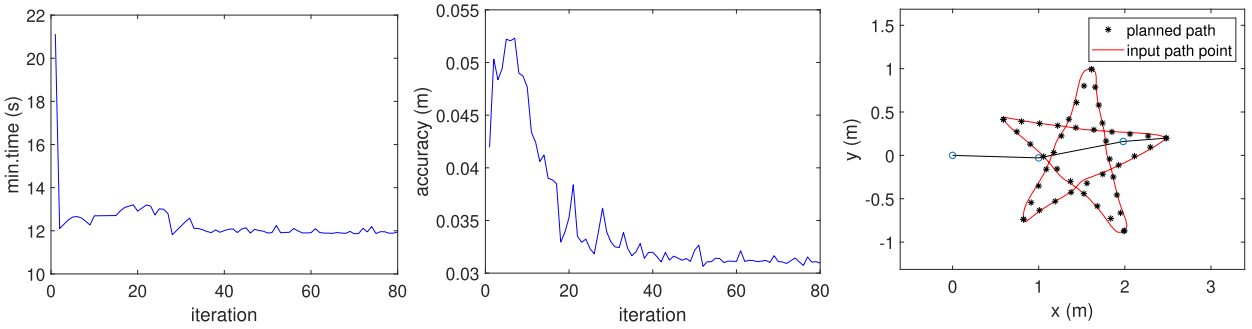
The simulation result in FIGURE 13(b) present the trajectory accuracy varies with iteration times. The trajectory accuracy, as mentioned in III, is a indicator to compare the similarity of two path. The accuracy of planned trajectory is higher if its value is smaller.

Figures such as FIGURE 13(c) show the presentation of the manipulator and the planned path. The green asterisk represents the input point of the specified path, and the blue curve represents the planned path which manipulator actually passed by. As the result shown in this figure, the manipulator well followed the given point.

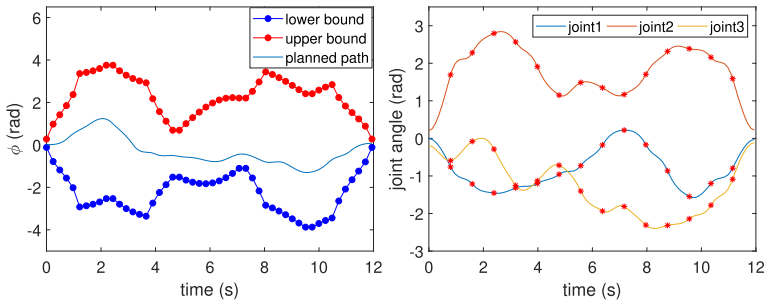
The simulation result in FIGURE 13(d) present the planned path of the redundant parameter ϕ. The x-axis represents the passed time, it shows the value of ϕ varies with time. The red points above are the maximum of ϕ when manipulator passed given points and the purple points below are minimum.

2) IN closed curve

Another path, the closed curve is chosen in this subsection.

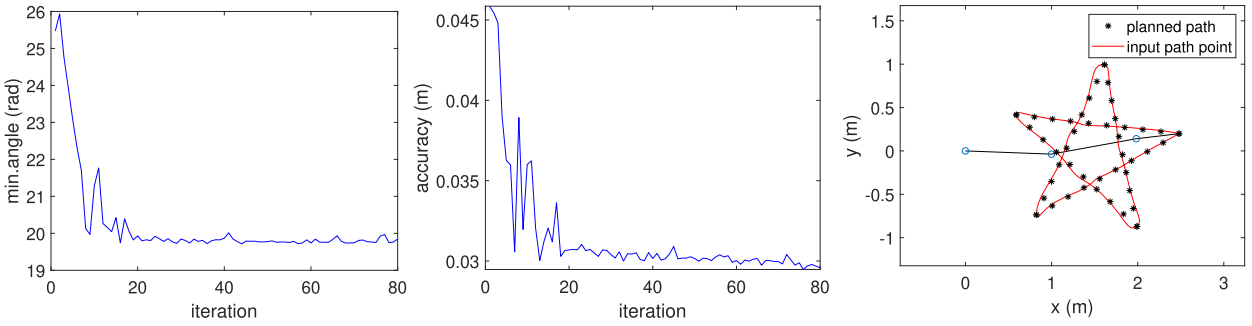


(a) Execution time varies with iteration (b) Trajectory accuracy varies with iteration (c) The planned path with manipulator

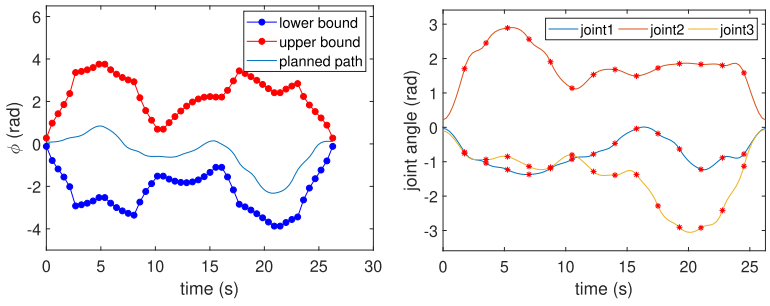


(d) The planned path of redundant parameter ϕ with its upper and lower bound (e) Joint angles varies with time

FIGURE 13. **Minimum execution time trajectory planning while giving a path of pentagram.**



(a) Joint rotation varies with iteration (b) Trajectory accuracy varies with iteration (c) The planned path with manipulator



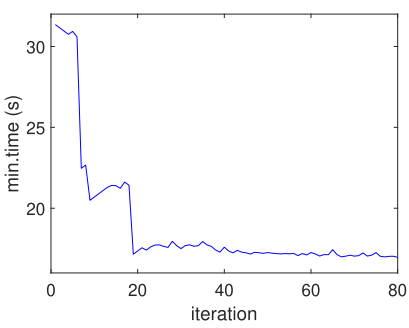
(d) The planned path of redundant parameter ϕ with its upper and lower bound (e) Joint angles varies with time

FIGURE 14. **Minimum joint rotation trajectory planning while giving a path of pentagram.**

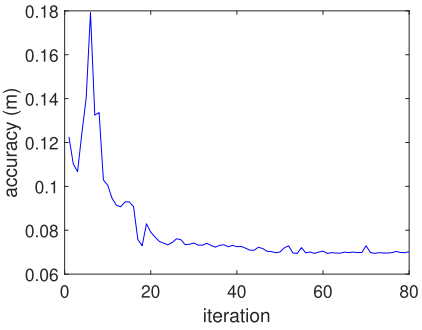
As we can see in FIGURE 15(a) and FIGURE 16(a), the number of iterations that reach convergence does not differ too much compare to FIGURE 13(a) and FIGURE 14(a) in Subsection IV-C.1.

D. COMPARISONS WITH PSO AND GA

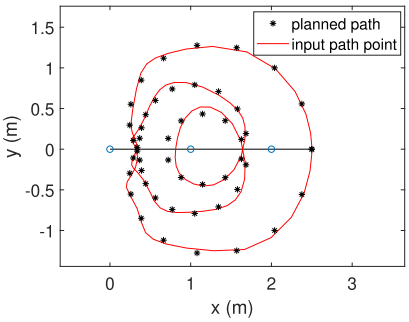
In this subsection, we compared our TPBSO algorithm to traditional bio-heuristic algorithms GA and PSO. To compare the performance between them, we start in three parts.



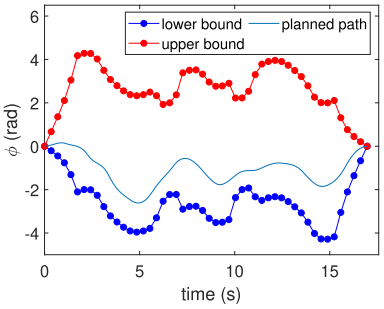
(a) Execution time varies with iteration



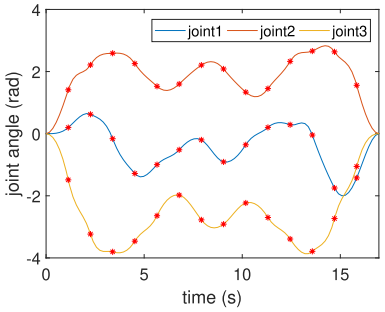
(b) Trajectory accuracy varies with iteration



(c) The planned path with manipulator

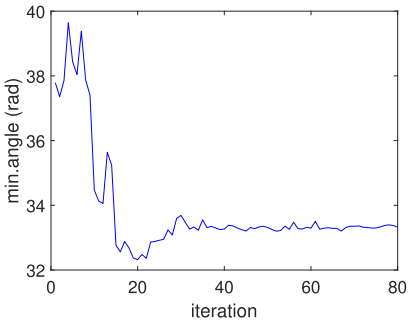


(d) The planned path of redundant parameter ϕ with its upper and lower bound

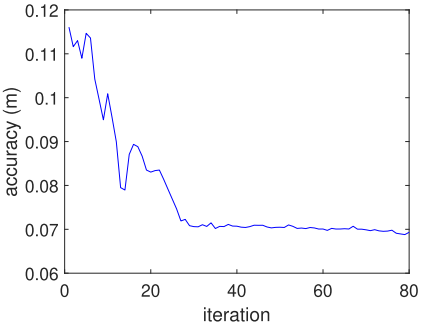


(e) Joint angles varies with time

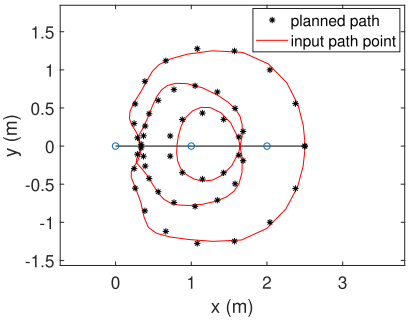
FIGURE 15. Minimum execution time trajectory planning while giving a path of pentagram.



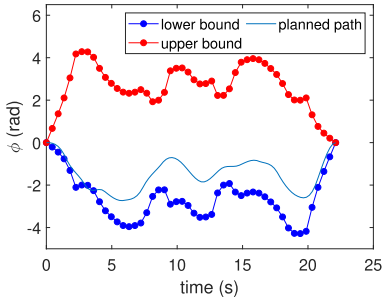
(a) Joint rotation varies with iteration



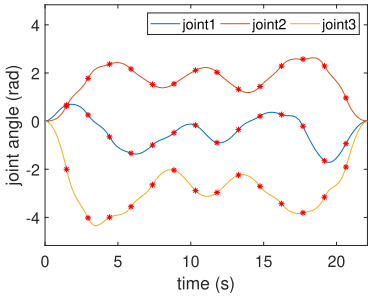
(b) Trajectory accuracy varies with iteration



(c) The planned path with manipulator



(d) The planned path of redundant parameter ϕ with its upper and lower bound



(e) Joint angles varies with time

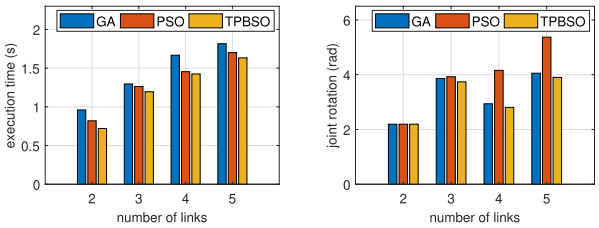
FIGURE 16. Minimum joint rotation trajectory planning while giving a path of pentagram.

The first and second part is the comparison of their performance in two simulation experiments we mentioned in Subsection IV-A and Subsection IV-C. The third part is the comparison of their computational complexity, both in time and space. In each part, we performed multiple simulations and took the average of the results.

1) POINT-TO-POINT TRAJECTORY PLANNING COMPARISON

In this subsection, we performed two sets of experiments with the goal of minimum execution time and minimum joint rotation respectively, and in each set of experiments, we set up four situations from 2-link to 5-link differed from the number of links. In each situation, we tested different algorithms each 20 times and takes the average value.

From FIGURE 17, we can see that whether in the comparison of execution time or joint rotation, TPBSO shows the best performance among three algorithms. PSO has a close performance with TPBSO in minimum execution time planning but has a poor performance in minimum joint rotation planning, and GA is just the opposite.



(a) Comparison of execution time under different number of links

(b) Comparison of joint rotation under different number of links

FIGURE 17. Performance comparisons of different algorithms in point-to-point trajectory planning.

2) TRAJECTORY PLANNING IN DIFFERENT SPECIFIED PATH COMPARISON

The comparison of this subsection corresponds to the second simulation, we also tested each algorithm 20 times in each specified path with the goal of minimum execution time and minimum joint rotation.

TABLE 1 shows the results, we compared the average performance and the stability of each algorithm. $\bar{t}$ represents the average value of execution time, t_σ represents the standard deviation of execution time. Similarly, $\bar{\theta}$ represents the average value of joint rotation and θ_σ represents its standard deviation. Both in execution time and joint rotation, TPBSO has lower variance and better performance compare to PSO and GA.

TABLE 1. Comparisons of execution time and joint rotation of different algorithms under different paths.

Path	Algorithm	$\bar{t}$(s)	t_σ(s)	$\bar{\theta}$(rad)	θ_σ(rad)
Pentagram	GA	0.7982	0.0757	25.4070	1.3351
	PSO	0.7432	0.0603	22.8910	1.9832
	TPBSO	**0.7139**	**0.0420**	**22.3850**	**1.0973**
Closed curve	GA	1.4088	0.1982	24.3354	0.2205
	PSO	1.2569	0.0370	24.3594	0.2050
	TPBSO	**1.1863**	**0.0043**	**24.1952**	**0.2365**

3) COMPUTATIONAL COMPLEXITY COMPARISON

In this subsection, we compare the computational complexity between these algorithm. In order to compare the performance, we need to make a more specific evaluation. Therefore, we divide the comparison of computational complexity into two parts: time and space.

a: TIME COMPLEXITY

The time complexity of the algorithm is a function, which qualitatively describes how the running time of the algorithm changes with the increase of the problem scale. We chose some parameters as an indicator, and analysed the growth of time when these parameters grow. In the case of time complexity, we set four parameters: population size, parameter dimension, number of points of the output trajectory, number of iterations. The comparison result of time complexity is shown in Table 2, where n_p represents the population size, n_d represents the parameter dimension, n_o represents the number of points of the output trajectory, and n_i represents the number of iterations.

TABLE 2. A time complexity comparison table for two experiments.

Algorithm	Complexity in one iteration	Complexity in Experiment I	Complexity in Experiment II
TPBSO	$O(n_p n_d)$	$O(n_i n_p n_d n_o)$	$O(n_i n_p n_o)$
PSO	$O(n_p n_d)$	$O(n_i n_p n_d n_o)$	$O(n_i n_p n_o)$
GA	$O(n_p n_d)$	$O(n_i n_p n_d n_o)$	$O(n_i n_p n_o)$

b: SPACE COMPLEXITY

The space complexity of the algorithm is a function, which qualitatively describes how the memory use of the algorithm changes with the increase of the problem scale. In the case of space complexity, the number of iterations does not affect, and we chose the number of input points as an alternative. The comparison result of space complexity is shown in TABLE 3, where n_{in} is the number of input points.

TABLE 3. A space complexity comparison table for two experiments.

Algorithm	Complexity of algorithm it self	Complexity in Experiment I	Complexity in Experiment II
TPBSO	$O(n_p n_d)$	$O((n_p+n_o)n_d)$	$O(n_o+n_{in}+n_p n_d)$
PSO	$O(n_p n_d)$	$O((n_p+n_o)n_d)$	$O(n_o+n_{in}+n_p n_d)$
GA	$O(n_p n_d)$	$O((n_p+n_o)n_d)$	$O(n_o+n_{in}+n_p n_d)$

From TABLE 2 and TABLE 3 we can find that these three algorithms are not significantly different from the perspective of either time complexity and space complexity. In other words, our algorithm achieves better planning performance without increasing the computational complexity of the algorithm.

V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a new trajectory planning algorithm called TPBSO. Based on a heuristic optimization algorithm called BSO, TPBSO is applied to the manipulator trajectory planning. In addition, we have discussed two types of manipulator trajectory planning problems in this paper,

and establish corresponding planning models and cost functions. Moreover, we use TPBSO to simulate these two kinds of problems, and compare it with two other heuristic optimization algorithms PSO and GA.

We mainly compare them from two aspects. One is task performance, which is measured by the optimization indicators of the planning model we built; the other is computational performance, which is measured by computational complexity. Through the comparison of task performance, we can find that our algorithm can find a relatively better trajectory which consume less execution time and joint rotation. And through the comparison of computational performance, our algorithm has the same level computational complexity with PSO and GA in both time complexity and space complexity.

Our current work is only a preliminary combination of heuristic optimization algorithm and manipulator trajectory planning. Especially, our simulation is only in two-dimensional space, and the motion of the manipulator in reality is more complex. Also, we haven't taken the dynamics of manipulator into account in our planning model, which seemed simply. Therefore, to improve our algorithm, our next research direction is to combine robotic dynamics and apply our algorithm to a more realistic environment. In this way, more complex robot trajectory planning problems like dynamic obstacle avoidance can also be added in.

REFERENCES

[1] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," in *Motion and Operation Planning of Robotic Systems*. Cham, Switzerland: Springer, 2015, pp. 3–27.

[2] M. Stenmark and J. Malec, "Knowledge-based instruction of manipulation tasks for industrial robotics," *Robot. Comput.-Integr. Manuf.*, vol. 33, pp. 56–67, Jun. 2015.

[3] R. S. Penning, J. Jung, J. A. Borgstadt, N. J. Ferrier, and M. R. Zinn, "Towards closed loop control of a continuum robotic manipulator for medical applications," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2011, pp. 4822–4827.

[4] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, "Continuum robots for medical applications: A survey," *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1261–1280, Dec. 2015.

[5] A. A. Ata, "Optimal trajectory planning of manipulators: A review," *J. Eng. Sci. Technol.*, vol. 2, no. 1, pp. 32–54, 2007.

[6] Z. Li, F. Xu, Q. Feng, J. Cai, and D. Guo, "The application of ZFD formula to kinematic control of redundant robot manipulators with guaranteed motion precision," *IEEE Access*, vol. 6, pp. 64777–64783, 2018.

[7] Y. Lei, B. Liao, and Q. Yin, "A noise-acceptable ZNN for computing complex-valued time-dependent matrix pseudoinverse," *IEEE Access*, vol. 7, pp. 13832–13841, 2019.

[8] Z. Tan, Y. Hu, L. Xiao, and K. Chen, "Robustness analysis and robotic application of combined function activated RNN for time-varying matrix pseudo inversion," *IEEE Access*, vol. 7, pp. 33434–33440, 2019.

[9] W. Li, L. Xiao, and B. Liao, "A finite-time convergent and noise-rejection recurrent neural network and its discretization for dynamic nonlinear equations solving," *IEEE Trans. Cybern.*, to be published.

[10] Y. Zhang, S. Li, S. Kadry, and B. Liao, "Recurrent neural network for kinematic control of redundant manipulators with periodic input disturbance and physical constraints," *IEEE Trans. Cybern.*, vol. 49, no. 12, pp. 4194–4205, Dec. 2019.

[11] D. Chen and Y. Zhang, "Robust zeroing neural-dynamics and its time-varying disturbances suppression model applied to mobile robot manipulators," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4385–4397, Sep. 2018.

[12] S. Li, M. Zhou, and X. Luo, "Modified primal-dual neural networks for motion control of redundant manipulators with dynamic rejection of harmonic noises," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4791–4801, Oct. 2018.

[13] Z. Zhang and Z. Yan, "Hybrid-level joint-drift-free scheme of redundant robot manipulators synthesized by a varying-parameter recurrent neural network," *IEEE Access*, vol. 6, pp. 34967–34975, 2018.

[14] D. Chen, S. Li, Q. Wu, and X. Luo, "New disturbance rejection constraint for redundant robot manipulators: An optimization perspective," *IEEE Trans. Ind. Informat.*, to be published.

[15] D. Guo, F. Xu, and L. Yan, "New pseudoinverse-based path-planning scheme with PID characteristic for redundant robot manipulators in the presence of noise," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 6, pp. 2008–2019, Nov. 2018.

[16] Z. Li, B. Liao, F. Xu, and D. Guo, "A new repetitive motion planning scheme with noise suppression capability for redundant robot manipulators," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.

[17] L. Jin and S. Li, "Distributed task allocation of multiple robots: A control perspective," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 48, no. 5, pp. 693–701, May 2018.

[18] Y. Zhang, S. Li, and X. Zhou, "Recurrent-neural-network-based velocity-level redundancy resolution for manipulators subject to a joint acceleration limit," *IEEE Trans. Ind. Electron.*, vol. 66, no. 5, pp. 3573–3582, May 2019.

[19] D. Chen, Y. Zhang, and S. Li, "Zeroing neural-dynamics approach and its robust and rapid solution for parallel robot manipulators against superposition of multiple disturbances," *Neurocomputing*, vol. 275, pp. 845–858, Jan. 2018.

[20] U. Pattacini, F. Nori, L. Natale, G. Metta, and G. Sandini, "An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2010, pp. 1668–1674.

[21] D. Chen, Y. Zhang, and S. Li, "Tracking control of robot manipulators with unknown models: A jacobian-matrix-adaption method," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3044–3053, Jul. 2018.

[22] J. Jin, L. Xiao, M. Lu, and J. Li, "Design and analysis of two FTRNN models with application to time-varying Sylvester equation," *IEEE Access*, vol. 7, pp. 58945–58950, 2019.

[23] L. Jin, S. Li, H. M. La, X. Zhang, and B. Hu, "Dynamic task allocation in multi-robot coordination for moving target tracking: A distributed approach," *Automatica*, vol. 100, pp. 75–81, Feb. 2019.

[24] D. Chen, S. Li, F.-J. Lin, and Q. Wu, "New super-twisting zeroing neural-dynamics model for tracking control of parallel robots: A finite-time and robust solution," *IEEE Trans. Cybern.*, to be published.

[25] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[26] W. F. Carriker, P. K. Khosla, and B. H. Krogh, "The use of simulated annealing to solve the mobile manipulator path planning problem," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2002, pp. 204–209.

[27] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching–learning-based optimization: An optimization method for continuous non-linear large scale problems," *Inf. Sci.*, vol. 183, no. 1, pp. 1–15, Jan. 2012.

[28] H. A. Eiselt, M. Gendreau, and G. Laporte, "Arc routing problems, part I: The Chinese postman problem," *Oper. Res.*, vol. 43, pp. 231–242, Apr. 2008.

[29] S. Gass and C. M. Harris, "Encyclopedia of operations research and management science," *J. Oper. Res. Soc.*, vol. 48, no. 7, p. 759, 2006.

[30] D. Chen, S. Li, and Q. Wu, "Rejecting chaotic disturbances using a super-exponential-zeroing neurodynamic approach for synchronization of chaotic sensor systems," *Sensors*, vol. 19, no. 1, P. 74, Dec. 2018.

[31] M. V. Weghe, D. Ferguson, and S. S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in *Proc. 7th IEEE-RAS Int. Conf. Humanoid Robots (HUMANOIDS)*, Nov./Dec. 2008, pp. 477–482.

[32] A. Abe, "Minimum energy trajectory planning method for robot manipulator mounted on flexible base," in *Proc. 9th Asian Control Conf. (ASCC)*, Jun. 2013, pp. 1–7.

[33] D. Chen and Y. Zhang, "Minimum jerk norm scheme applied to obstacle avoidance of redundant robot arm with jerk bounded and feedback control," *IET Control Theory Appl.*, vol. 10, no. 15, pp. 1896–1903, Jun. 2016.

[34] D. Chen, S. Li, W. Li, and Q. Wu, "A multi-level simultaneous minimization scheme applied to jerk-bounded redundant robot manipulators," *IEEE Trans. Autom. Sci. Eng.*, to be published.

[35] Y. Chen and A. A. Desrochers, "Structure of minimum-time control law for robotic manipulators with constrained paths," in *Proc. Int. Conf. Robot. Automat.*, May 2003, pp. 971–976.
[36] W. F. Faris, A. A. Ata, and M. Y. Sa'Adeh, "Energy minimization approach for a two-link flexible manipulator," *J. Vib. Control*, vol. 15, pp. 497–526, Apr. 2009.
[37] A. Piazzi and A. Visioli, "Global minimum-jerk trajectory planning of robot manipulators," *IEEE Trans. Ind. Electron.*, vol. 47, no. 1, pp. 140–149, Feb. 2000.
[38] D. Chen and Y. Zhang, "A hybrid multi-objective scheme applied to redundant robot manipulators," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 3, pp. 1337–1350, Jul. 2017.
[39] A. Gasparetto and V. Zanotto, "A technique for time-jerk optimal planning of robot trajectories," *Robot. Comput.-Integr. Manuf.*, vol. 24, no. 3, pp. 415–426, 2008.
[40] R. Menasri, A. Nakib, B. Daachi, H. Oulhadj, and P. Siarry, "A trajectory planning of redundant manipulators based on bilevel optimization," *Appl. Math. Comput.*, vol. 250, pp. 934–947, Jan. 2015.
[41] H.-Q. Min, J.-H. Zhu, and X.-J. Zheng, "Obstacle avoidance with multi-objective optimization by PSO in dynamic environment," in *Proc. Int. Conf. Mach. Learn. Cybern.*, vol. 5, Aug. 2005, pp. 2950–2956.
[42] L. Wang, Y. Liu, H. Deng, and Y. Xu, "Obstacle-avoidance path planning for soccer robots using particle swarm optimization," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2006, pp. 1233–1238.
[43] Y.-Q. Qin, D.-B. Sun, N. Li, and Y.-G. Cen, "Path planning for mobile robot using the particle swarm optimization with mutation operator," in *Proc. Int. Conf. Mach. Learn. Cybern.*, vol. 4, Aug. 2005, pp. 2473–2478.
[44] P. Bhattacharjee, P. Rakshit, I. Goswami, A. Konar, and A. K. Nagar, "Multi-robot path-planning using artificial bee colony optimization algorithm," in *Proc. 3rd World Congr. Nature Biol. Inspired Comput. (NaBIC)*, Oct. 2011, pp. 219–224.
[45] P. Savsani, R. L. Jhala, and V. J. Savsani, "Optimized trajectory planning of a robotic arm using teaching learning based optimization (TLBO) and artificial bee colony (ABC) optimization techniques," in *Proc. 7th IEEE Int. Syst. Conf. (SysCon)*, Apr. 2013, pp. 381–386.
[46] P. Allgeuer and S. Behnke, "Hierarchical and state-based architectures for robot behavior planning and control," in *Proc. 8th Workshop Hum. Soccer Robots, Int. Conf. Hum. Robots (Humanoids)*, Sep. 2018, pp. 1–6.
[47] S. Li, M. Zhou, X. Luo, and Z.-H. You, "Distributed winner-take-all in dynamic networks," *IEEE Trans. Autom. Control*, vol. 62, no. 2, pp. 577–589, Feb. 2017.
[48] S. Li, R. Kong, and Y. Guo, "Cooperative distributed source seeking by multiple robots: Algorithms and experiments," *IEEE/ASME Trans. Mechatron.*, vol. 19, no. 6, pp. 1810–1820, Dec. 2014.
[49] T. Wang, L. Yang, and Q. Liu, "Beetle swarm optimization algorithm: Theory and application," 2018, *arXiv:1808.00206*. [Online]. Available: https://arxiv.org/abs/1808.00206
[50] Q. Wu, Z. Ma, G. Xu, S. Li, and D. Chen, "A novel neural network classifier using beetle antennae search algorithm for pattern classification," *IEEE Access*, vol. 7, pp. 64686–64696, 2019.
[51] Q. Wu, Z. Ma, J. Fan, G. Xu, and Y. Shen, "A feature selection method based on hybrid improved binary quantum particle swarm optimization," *IEEE Access*, vol. 7, pp. 80588–80601, 2019.

LEI WANG received the B.S. degree in computer science and technology from Zhejiang University City College, Hangzhou, China, in 2018. He is currently pursuing the M.S. degree in software engineering with Hangzhou Dianzi University, Hangzhou. His research interests include path planning, intelligent optimization algorithms, and machine learning.

QING WU received the Ph.D. degree in computer science from Zhejiang University, in 2006. He is a currently a Professor with the School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China. His current research interests include machine learning, data mining, adaptive software, and ubiquitous computing.

FEI LIN received the degree from the CAD Institute, Hangzhou Dianzi University, in 2002. She has a wealth of experience in software research and development, project management, and product planning.

Her current research interests include data mining, block chain, intelligent information system, and software engineering.

SHUAI LI received the B.E. degree in precision mechanical engineering from the Hefei University of Technology, China, in 2005, the M.E. degree in automatic control engineering from the University of Science and Technology of China, China, in 2008, and the Ph.D. degree in electrical and computer engineering from the Stevens Institute of Technology, USA, in 2014. He is currently a Research Assistant Professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. His current research interests include dynamic neural networks, wireless sensor networks, robotic networks, machine learning, and other dynamic problems defined on a graph. He is on the Editorial Board of the *International Journal of Distributed Sensor Networks*. He is currently on the Editorial Board of the *Neural Computing and Applications* and the *International Journal of Distributed Sensor Networks*.

DECHAO CHEN received the B.S. degree in electronic information science and technology from the Guangdong University of Technology, Guangzhou, China, in 2013, and the Ph.D. degree in information and communication engineering from Sun Yat-sen University, Guangzhou, in 2018. He is currently a Postdoctoral Fellow of the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. He is also an Associate Professor with the School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China. His research interests include robotics, neural networks, dynamics systems, control systems, optimization, and machine learning.