International Journal of Innovative Computing, Information and Control
Volume **17**, Number **2**, April **2021**

 ISSN 1349-4198
pp. 595–611

OPTIMIZING THE OBSTACLE AVOIDANCE TRAJECTORY AND POSITIONING ERROR OF ROBOTIC MANIPULATORS USING MULTIGROUP ANT COLONY AND QUANTUM-BEHAVED PARTICLE SWARM OPTIMIZATION ALGORITHMS

Yan-Ting Chen and Wen-Jong Chen

Department of Industrial Education and Technology
National Changhua University of Education
No. 2, Baoshan Road, Changhua City, Changhua County 500, Taiwan
s901510011@gmail.com; wjong@cc.ncue.edu.tw

Received September 2020; revised January 2021

Abstract. *The primary objective of this study was to optimize the trajectory planning and positioning error of a five-degree-of-freedom robotic manipulator. First, a multigroup ant colony optimization (MACO) algorithm was used to determine the shortest trajectory with obstacle avoidance. A quantum-behaved particle swarm optimization (QPSO) algorithm was subsequently used to determine the optimal positioning error for each moving point along the shortest trajectory. The simulation results revealed that compared with the traditional ant colony optimization algorithm, the MACO algorithm reduced the number of iterations by 66% and the path length by approximately 5%-7%. Moreover, the QPSO algorithm ensured that the robotic manipulator performed the least possible number of joint movements. This algorithm also solved the inverse kinematics problem. The optimal configuration had a positioning error of 10^{-3} mm. The current results can be extensively used for designing and analyzing multiaxial robotic manipulators.*
Keywords: Robotic manipulator, Multigroup ant colony optimization, QPSO algorithm, Positioning error

1. **Introduction.** In recent years, robotic manipulators have replaced manual labor to ensure fast processing for improving the accuracy and stability of processes such as the machining of small parts, assembly of electronic components, painting, fabrication of integrated circuits, and welding. The accumulative errors of the operating mechanisms of robotic manipulators directly affect the accuracy of the end product. Therefore, to improve robotic manipulators' operating efficiency in various processes, their trajectory planning and positioning accuracy must be optimized. In general, robotic manipulator trajectory planning involves determining the optimal trajectory from a starting point to a target point while avoiding obstacles. During the movement of a robotic manipulator, the position of the gripper end must be accurately calculated such that the optimal trajectory is obtained from the predetermined motion and position of each motor-rotated joint angle.

Numerous researchers have reported on the optimization of the obstacle avoidance trajectory and positioning error of robotic manipulators. Cai et al. [1] presented a method for determining the structural and angular parameters in robot kinematics to compensate for an error model; the authors effectively reduced the angle errors by using the robot kinematics equation and Denavit-Hartenberg (D-H) algorithm for correcting and compensating for the errors in the joint angles. Liu et al. [2] used equal grids to optimize a robotic manipulator's trajectory by using an ant colony algorithm in a two-dimensional

DOI: 10.24507/ijicic.17.02.595

environment; the authors analyzed the optimum operating mechanism of the ant colony algorithm for alleviating the local optimization drawbacks and expediting the search. Shukla et al. [3] used a variational principle to optimize the trajectories of serial manipulators and presented modified boundary conditions to search for feasible trajectories under obstacle avoidance constraints. Finally, Fu et al. [4] developed an improved trajectory planning algorithm that considers the obstacle crossing of given nodes and provided a criterion for eliminating redundant nodes.

In recent research, the development of stochastic algorithms has been increasing, along with their use for optimization. For instance, numerous researchers have applied methods based on genetic algorithms (GAs), which are well-developed and bio-inspired, to handle the obstacle avoidance trajectory inverse kinematics, and positioning error optimization problem.

Jiang et al. [5] constructed a model of a four-degree-of-freedom (DOF) manipulator and used a GA to solve the trajectory planning problem for robotic manipulators. The fitness function adopted by the authors considers the maximum torque, path length, and total distance of each joint. Zacharia et al. [6] used a modified GA involving special encoding to resolve the multiplicity of the robot inverse kinematics problem and analyzed the collision-free motion plans and scheduled time-optimal routes along with a set of given task points. Menasri et al. [7] presented a bi-level GA for determining the optimal joint angles and configurations that prevent the problems associated with the obstacles and singularities of robotic manipulators. Kalra et al. [8] used a real-coded GA to solve the multimodal inverse kinematics problem associated with industrial robots and determined the robot configuration closest to an existing robot configuration in a joint space to achieve the required manipulator position. Tabandeh et al. [9] adopted a GA to solve the inverse kinematics problem of a serial robotic manipulator and added a modified filtering and clustering phase to the GA for obtaining the link parameters and joint variables of the manipulator. El-Sherbiny et al. [10] compared the performance of several methods, including algebraic, geometric, and iterative methods, and their adaptive neurofuzzy inference systems for solving the inverse kinematics problem. Their results indicated that a GA has superior positioning performance but requires a long time to achieve convergence. Thus, in general, a GA's convergence and performance depend on the shape of the solution space, and its mechanism is fully stochastic and discrete such that the solutions are difficult to converge to globally optimal solutions through its dynamics.

The particle swarm optimization (PSO) algorithm has a different evolutionary mechanism that gives an optimal solution with a convergence circle that depends on the parameters of the algorithm. This algorithm has been used in different scenarios [11] and its variant versions have also been proposed [12-14]. The use of a PSO algorithm for solving the robotic manipulator optimizing problem has been discussed previously. For instance, Huang et al. [15] presented a PSO algorithm for solving the redundant inverse kinematics problem of seven-DOF robotic manipulators. Zhou et al. [16] employed neural networks and a PSO algorithm to determine the absolute positioning accuracy of industrial robots used in flexible automated assembly for aircraft. The authors constructed a precise error model between neighborhoods of points to compensate for the precision error. Prasert-taweelap et al. [17] applied a PSO algorithm to aiding trajectory planning with obstacle avoidance for a real-life layout of hard disk manufacturer; the authors proposed a risk function to define safety zones and combined the risk function and the path length into a cost function. In their work, the PSO algorithm was used to construct a new data point in an interpolated trajectory and to minimize a defined cost function. However, the limitations of PSO include algorithm structure, whereby the particles follow certain deterministic mechanism and easily converge into a locally optimal solution.

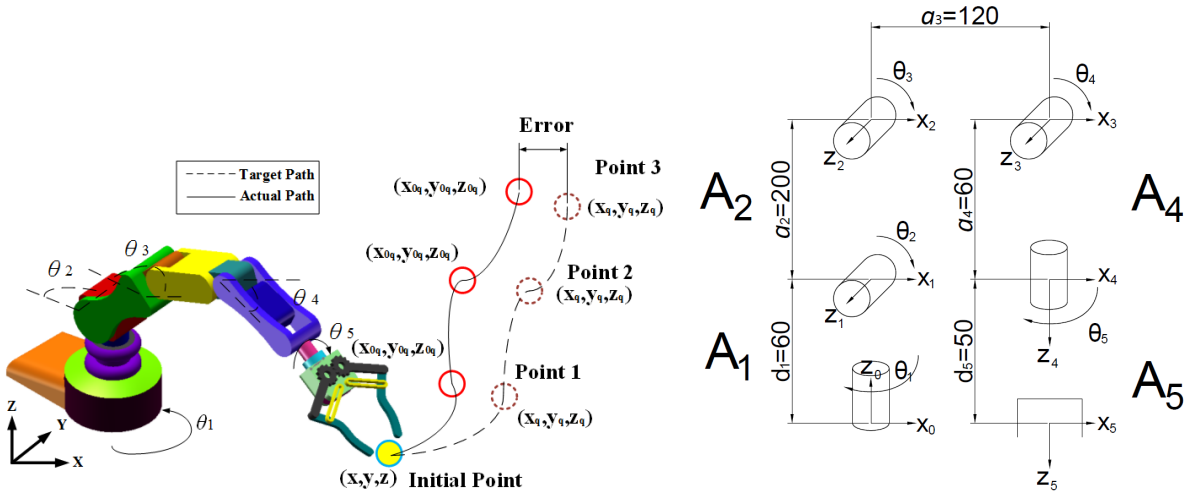
Other related works have considered the use of artificial neural networks (ANNs), which are composed of neurons, weightings and an optimization process; ANNs have been widely applied in nonlinear mapping problems and related works. For instance, Pham et al. [18] used the bees algorithm to train multilayer perceptron neural networks for modeling the inverse kinematics of an articulated robotic manipulator and established that the bees algorithm was robust in consistently training the neural networks to model kinematics data with high accuracy. Mayorga and Sanongboon [19] used an ANN for the rapid computation of the inverse kinematics problem and the effective prevention of the singularities of redundant manipulators. In particular, the authors analyzed constrained geometrical concepts to establish characterization matrices for preventing singularities and identifying a safe trajectory. Hasan et al. [20] used an ANN to control a robotic manipulator's motion; this ANN was trained to learn the desired set of joint angle positions from a given set of end-effector positions. Lin et al. [21] designed a convolutional neural network (CNN)-based obstacle avoidance algorithm for unmanned underwater vehicles; to avoid obstacles, this algorithm could adjust the dynamic parameter, yaw, and velocity.

Taken together, the reviewed literature indicates that the predictive effectiveness of an ANN depends on the number of effective training samples and parameter adjustment; moreover, the ANN training process is time-consuming, and collecting an effective amount of samples, even for sample size determination, can be difficult [22-24]. Moreover, a GA can easily lose favorable solutions during the optimization process. Furthermore, traditional PSO algorithms have a limitation in that they cannot cover the entire solution space in a feasible region. Nevertheless, some variants of aforementioned methods are proposed to improve the convergence and solution quality.

Therefore, in the current study, two algorithms were applied for solving two subproblems related to a robotic manipulator: 1) a multigroup ant colony optimization (MACO) algorithm to determine the shortest trajectory under static obstacle avoidance conditions and 2) the quantum-behaved PSO (QPSO) algorithm to calculate the angles of each joint of the robotic manipulator, along with the positioning errors, by solving the inverse kinematics problem.

The remainder of this paper is organized as follows. Section 2 describes the kinematics of the robotic manipulator. Section 3 presents the trajectory and positioning error problems. Section 4 describes the MACO algorithm-based solution for the optimal trajectory subproblem of obstacle avoidance. Section 5 delineates the QPSO algorithm-based solution of the inverse kinematics subproblem. Section 6 presents the study flow. Section 7 discusses the results for the two algorithms in detail. Finally, Section 8 concludes the paper.

2. **Manipulator Kinematics.** The topography and joint motion of the five-DOF robotic manipulator are displayed in Figures 1(a) and 1(b). In these figures, θ_1, θ_2, θ_3, θ_4, and θ_5 are the joint angles, which are the configuration parameters of the manipulator; x_q, y_q, and z_q are the spatial coordinates of the target position of the robotic manipulator; and x_{0q}, y_{0q}, and z_{0q} are the spatial coordinates of the determined position of the robotic manipulator. The positioning error is affected by the configuration parameters and the resolutions of the rotator encoder. Each joint of a five-DOF robotic manipulator with five servo motor components has a certain angle with a limited motor rotation range. The elementary homogeneous transform matrices (${}^iA_{i+1}, i = 0, 1, 2, 3, 4$) and the D-H coordinate transformation were used to compute the matrix ${}^0T_{i+1}$, which defines the transformation between the $X_0Y_0Z_0$ coordinate system attached to the base and the $X_{i+1}Y_{i+1}Z_{i+1}$ coordinate system attached to the ith link. Equation (1) indicates that ${}^iA_{i+1}$ is a homogeneous



(a) (b)

FIGURE 1. Schematics of a five-DOF robotic manipulator: (a) joint angle, target trajectory, and actual trajectory and (b) coordinates of the parameters and links

transformation matrix between the consecutive coordinate frames i and $i+1$. The following definitions of the link parameters are valid: 1) a_i is the distance from Z_i to Z_{i+1} measured along X_i; 2) α_i is the angle between Z_i and Z_{i+1} measured about X_i; 3) d_i is the distance from X_i to X_{i+1} measured along Z_i; 4) θ_i is the angle between X_i and X_{i+1} measured about Z_i.

$$
\begin{aligned}
{}^{i}A_{i+1} &= Rot(z,\theta_{i+1}) \times Trans(0,0,d_{i+1}) \times Trans(a_{i+1},0,0) \times Rot(x,a_{i+1}) \\
&= \begin{bmatrix} \cos\theta_{i+1} & -\sin\theta_{i+1} & 0 & 0 \\ \sin\theta_{i+1} & \cos\theta_{i+1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{i+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_{i+1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_{i+1} & -\sin\alpha_{i+1} & 0 \\ 0 & \sin\alpha_{i+1} & \cos\alpha_{i+1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta_{i+1} & -\sin\theta_{i+1}\cos\alpha_{i+1} & \sin\theta_{i+1}\sin\alpha_{i+1} & \alpha_{i+1}\cos\theta_{i+1} \\ \sin\theta_{i+1} & \cos\theta_{i+1}\cos\alpha_{i+1} & -\cos\theta_{i+1}\sin\alpha_{i+1} & \alpha_{i+1}\sin\theta_{i+1} \\ 0 & \sin\alpha_{i+1} & \cos\alpha_{i+1} & d_{i+1} \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{1}
$$

For an n-joint robotic manipulator, Equation (2) presents the transformation matrix between the joints and links associated with the D-H parameters. The aforementioned matrix is a 4×4 matrix.

$$T_5 = {}^{0}A_1 {}^{1}A_2 {}^{2}A_3 {}^{3}A_4 {}^{4}A_5 \tag{2}$$

3. **Problem Description.** The optimization problem is divided into two subproblems: 1) the temporary static trajectory problem of a robotic manipulator and 2) the inverse kinematics problem of a robotic manipulator in a least movement condition. Both these two subproblems are solved using MACO and QPSO, described in Section 4 and Section 5.

3.1. **Trajectory and path length.** The term "temporary static trajectory" indicates that the target objects have no movement over a particular period. In most manufacturing processes, semifinished products are processed station by station and then classified by robotic manipulators. In a station, target objects can be placed at predefined target position. These objects can then be picked up using a trajectory formed by nodes

connected in a certain permutation (Figure 1). The cost function is described as follows:

$$\arg\min_{x} \sum_{i=2}^{n} D(\boldsymbol{p}(x_i), \boldsymbol{p}(x_{i-1})) = \sum_{i=2}^{n} \sqrt{\sum (p_j(x_i) - p_j(x_{i-1}))^2} \tag{3}$$

$$\boldsymbol{x} = [x_1, x_2, \ldots, x_n] \in \mathbb{Z}$$

where $\boldsymbol{p}(x_i)$ is the position vector of the x_ith node (where $\boldsymbol{x}$ is the solution vector, which is the permutation of nodes) and $\sqrt{\sum (p_j(x_i) - p_j(x_{i-1}))^2}$ represents the distance between nodes x_i and x_{i-1}. In addition, the distance is set as infinity if the connection of the nodes between x_i and x_{i-1} intersects the obstacle. The summation of the distances between nodes is referred to as the path length. The cost function is used for optimizing a permutation such that the path length is minimized. Herein, the MACO algorithm is used to solve the permutation problem with the least distance objective and obtain the optimal trajectory.

3.2. **Positioning error and least movement.** The aims of manipulator control are to ensure the precision of positioning and to minimize manipulator movements. A cost function for determining an optimal positioning error is the function that minimizes the distance between the final real position and the expected position of the end effector. The joint angles θ_1, θ_2, θ_3, θ_4, and θ_5 are the design variables (i.e., the configuration parameters) of the five-DOF manipulator. The movements of these joint angles should be minimized optimally to reduce the operation time. The D-H parameters of the simulated robotic manipulator are presented in Table 1.

TABLE 1. Constraints of each joint of the five-DOF robotic manipulator

Joint	θ_i	d_i (mm)	a_i (mm)	α_i (deg)
1	θ_1	60	0	90
2	θ_2	0	200	0
3	θ_3	0	120	0
4	θ_4	0	60	0
5	θ_5	0	50	90

The transformation T_5 between the base and end-effector frames is expressed as follows:

$$\begin{aligned} T_5 = &\begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & 0 \\ \sin\theta_1 & 0 & \cos\theta_1 & 0 \\ 0 & 1 & 0 & 60 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 200\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & 200\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &\begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 120\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & 120\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & 60\cos\theta_4 \\ \sin\theta_4 & \cos\theta_4 & 0 & 60\sin\theta_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &\begin{bmatrix} \cos\theta_5 & 0 & \sin\theta_5 & 50\cos\theta_5 \\ \sin\theta_5 & 0 & \cos\theta_5 & 50\sin\theta_5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{4}$$

The constraints of the configuration parameters are as follows: $-180^\circ < \theta_1 < 180^\circ$, $0^\circ < \theta_2 < 150^\circ$, $-90^\circ < \theta_3 < 90^\circ$, $-135^\circ < \theta_4 < 135^\circ$, and $-80^\circ < \theta_5 < 80^\circ$. These initial configuration parameters are denoted in the vector form as ${}^0\theta = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]^T$, and

the determined parameters for the jth position are denoted as ${}^{j}\theta$. Therefore, the cost function is defined as follows:

$$\arg\min Y(X) = \sqrt{(x_q - x_{0q})^2 + (y_q - y_{0q})^2 + (z_q - z_{0q})^2} + 10 \times \sum |\Delta\theta| \tag{5}$$
$$\Delta\theta = {}^{i}\theta - {}^{i-1}\theta$$

where $[\ x_q \quad y_q \quad z_q\ ]^T$ represents the target position, whose coordinates are obtained using the MACO algorithm. The position of the end effector relative to the base frame is $[\ x_{0q} \quad y_{0q} \quad z_{0q} \quad 1\ ] = \mathbf{T_5}[\ 0 \quad 0 \quad 0 \quad 1\ ]^T$. The configuration parameters $[\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]^T$ are present in the D-H matrix $\mathbf{T_5}$ in Equation (2). The cost function $Y(X)$ ensures the positioning accuracy, and $\Delta\theta$ is the angle change between the previous and present configurations that ensures the least manipulator movements.

4. MACO Algorithm.

4.1. **Ant colony system.** The ant colony system (ACS) algorithm has the local and global pheromone updating mechanisms used by ants [25]. The ACS algorithm is a population-based algorithm in which several artificial ants search for optimal solutions. This algorithm is based on the natural foraging behavior of ants, which use only pheromone information for determining the shortest path between their nest and the food source. Thus, a greater pheromone concentration on a path indicates a shorter path length. Ants following a foraging path mainly depend on the pheromone concentration to select the shortest path. Therefore, the pheromone update plays a crucial role in selecting the shortest path. The updating formula is expressed as follows:

$$\tau_{t+1}(i,j) \leftarrow (1-\rho)\cdot\tau_t(i,j) + \sum_{k=1}^{|T_{ant}|} \Delta\tau_m(i,j)$$
$$\Delta\tau_m \triangleq \frac{Q}{L_m} \tag{6}$$

In Equation (6), a pheromone can be defined as a discrete pheromone field between nodes i, j and denoted as $\tau(i,j)$. For simplicity, the update operations are applied to the same node index (i,j), which is omitted from τ. T_{ant} and $|T_{ant}|$ represent a set of ants and the total number of ants, respectively. Moreover, $\rho \in (0,1]$ is the evaporation rate, $\Delta\tau_m$ is the quantity of pheromone laid on the node (i,j) by the mth ant, Q is an arbitrarily selected number, and L_m is the tour length of the mth ant.

When searching for optimal solutions, the ants in the ACS algorithm construct several path candidates that follow a probabilistic decision rule. The transition probability of the mth ant moving from the previously visited node a to the next candidate node k is described as follows:

$$p_{m,t}(k) \triangleq \begin{cases} \dfrac{\tau_t^{\zeta}(a,k)\cdot\eta_t^{\beta}(a,k)}{\sum \tau_t^{\zeta}(a,k)\eta_t^{\beta}(a,k)}, & \text{if } k \in \text{not visited node} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$$F_k(k) = \sum_{k=1}^{k} p_{m,t}(k) \tag{8}$$

$$\eta_t^{\beta}(i,j) \triangleq \left(\frac{1}{d(i,j)}\right)^{\beta} \tag{9}$$

Here, $p_{m,t}(k)$ represents the probability for the kth node selection and that the mth ant has not yet visited at the tth iteration; $F_k(k)$ is the cumulative probability of $p_{m,t}(k)$; $d(i,j)$ is the path length between nodes i and j; ζ and β determine the influence of the pheromone and heuristic information, respectively. Moreover, the next visiting city is determined by the roulette wheel selection:

$$\arg\max_{l} F_k(l) - q \tag{10}$$

where q is an auxiliary random variable uniformly distributed within the range $[0,1]$.

Because low pheromone values exist in the local pheromone update mechanism of the ACS algorithm, the global pheromone update is executed at the end of each iteration. The global pheromone update mechanism is executed only by the best ant. In particular, only paths that were visited by the best ant are updated in the search process. The global pheromone update mechanism is expressed as follows:

$$\tau_t \leftarrow (1-\rho)\cdot\tau_t + \rho\cdot\Delta\tau_{best} \tag{11}$$

$$\Delta\tau_{best} = \begin{cases} \left(\dfrac{Q}{L_{best}}\right), & (i,j) \in \text{global best solution at iteration } t \\ 0, & \text{otherwise} \end{cases}$$

where the parameter L_{best} can be set to the length of the optimal path found in the current iteration. Each ant in the ACS applies a probabilistic action choice rule, which is called the pseudorandom proportional rule, to determine which of the neighboring nodes is to be visited next.

4.2. **MACO algorithm.** The drawbacks of the traditional ant colony optimization (ACO) algorithm are its tendency to be easily trapped into a local optimum and its unnecessary computations. These drawbacks increase the computation time and complexity of the algorithm and cause premature convergence. Therefore, an MACO algorithm [26] is presented to overcome the disadvantages of the traditional ACO algorithm. The ants in the MACO algorithm are divided equally into several partitions, which are also called "groups". The ants belonging to each group pass each node along their paths; thus, the pheromone concentrations of each path between nodes are constrained in all the groups. The MACO mechanism involves splitting the number of ants, forming their local pheromones, and finally, sharing these pheromones with the global pheromone pool at the end of ACS.

$$|G_d| = \frac{|T_{ant}|}{g} \tag{12}$$

$$\tau_{t+1} \leftarrow (1-\rho)\tau_t + \rho\sqrt[g]{\Delta\tau_{G_1}\times\Delta\tau_{G_2}\times\cdots\times\Delta\tau_{G_g}} \tag{13}$$

$$\Delta\tau_{G_d} = \begin{cases} \displaystyle\sum_{m=1}^{|T_{ant}|}\frac{Q}{L_{ij}^m}, & \text{if } (i,j) \text{ belong to the best tour} \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

where $\Delta\tau_{G_d}$ represents the pheromone increments for G_d, g is the number of groups (set as 3 in this study), G_d represents the dth ant group subset, 1-norm $|G_d|$ represents the number of ants in the dth group, and L_{ij}^m represents the path length between the nodes i and j visited by the mth ant of each group. The nodes of the optimal trajectory of each group at the tth iteration perform a random exchange during the information exchange process.

In addition, the number of groups is an empirical parameter, which depends on the complexity of the solution space, and the ant size of each group determines the quality of

convergence and optimal solution in a limited iteration. In this study, the related results of convergence along with parameters ζ, ρ and g are discussed in Section 7.

5. QPSO Algorithm.

5.1. **Overview of the traditional PSO algorithm.** The standard PSO algorithm [27] is an optimization technique that involves population-based stochastic optimization. This algorithm is inspired by the movement and intelligence of swarms. Each particle symbolizes a potential solution to the optimization problem in a search space. In the search process, the mth particle possesses a position vector $\boldsymbol{X_{m,t}}$ and velocity vector $\boldsymbol{V_{m,t}}$ in the current iteration t. The parameters $\boldsymbol{pbest_m}$ and $\boldsymbol{gbest}$ represent the optimal positions of the mth particle and all the particles, respectively. The mth particle updates its velocity $\boldsymbol{V_{m,t+1}}$ and position $\boldsymbol{X_{m,t+1}}$ according to the following equations:

$$\boldsymbol{V_{m,t+1}} = \omega \boldsymbol{V_{m,t}} + c_1 r_1 (\boldsymbol{pbest_m} - \boldsymbol{X_{m,t}}) + c_2 r_2 (\boldsymbol{gbest} - \boldsymbol{X_{m,t}}) \tag{15}$$

$$\boldsymbol{X_{m,t+1}} = \boldsymbol{X_{m,t}} + \boldsymbol{V_{m,t+1}} \tag{16}$$

where ω is the inertia weighting; r_1 and r_2 are uniformly distributed random numbers within the range $[0, 1]$; c_1 and c_2 are the control parameters to balance the effects of the velocity when updating $\boldsymbol{pbest_m}$ and $\boldsymbol{gbest}$.

5.2. **QPSO algorithm.** In the standard PSO algorithm, the current trajectory of each particle in search space is always constrained in an area according to the previous movement trajectories of the particle. Therefore, the standard PSO algorithm cannot ensure convergence to a globally optimal solution. This study used a QPSO algorithm to overcome the disadvantage of the standard PSO algorithm without losing its advantages. All the particles in the QPSO algorithm exhibit quantum behavior during the search process [28]. Only a control parameter α and the position vector for each particle are determined. The probability for each particle at position $\boldsymbol{x}$ and time t is presented using the probability density function $|\Psi(x,t)|^2$. Thus, the state of particle aggregation in the QPSO algorithm is completely different from that in the standard PSO algorithm. Consequently, the QPSO algorithm can search the entire feasible solution space. The particle movement and position update are conducted according to the following iterative equations:

$$\boldsymbol{p_m} = \varnothing \times \boldsymbol{pbest_m} + (1 - \varnothing) \times \boldsymbol{gbest_m} \tag{17}$$

$$\boldsymbol{mbest} = \frac{1}{N} \sum_{i=1}^{N} \frac{\boldsymbol{pbest_m}}{N} \tag{18}$$

$$\boldsymbol{X_{m,t+1}} = \boldsymbol{p_m} + \alpha \times |\boldsymbol{mbest} - \boldsymbol{X_{m,t}}| \cdot \ln\left(\frac{1}{u}\right), \text{ if } k \geqq 0.5 \tag{19}$$

$$\boldsymbol{X_{m,t+1}} = \boldsymbol{p_m} - \alpha \times |\boldsymbol{mbest} - \boldsymbol{X_{m,t}}| \cdot \ln\left(\frac{1}{u}\right), \text{ if } k < 0.5 \tag{20}$$

where $\boldsymbol{mbest}$ is the mean of the optimal positions of all the particles, α is a contraction-expansion coefficient that controls the convergent speed, $\boldsymbol{X_{m,t}}$ is the position of the mth particle at the tth iteration, N is the total number of particles, and $\boldsymbol{X_{m,t+1}}$ is the updated position vector of the mth particle. Moreover, u, k, and $\varnothing$ are random numbers that are distributed uniformly in the range $[0, 1]$.

6. Research Method Framework.

The framework of this study, displayed in Figure 2, illustrates the two algorithms being applied to solving the two subproblems of obstacle avoidance. The first subproblem is the trajectory problem, where the sequence of target

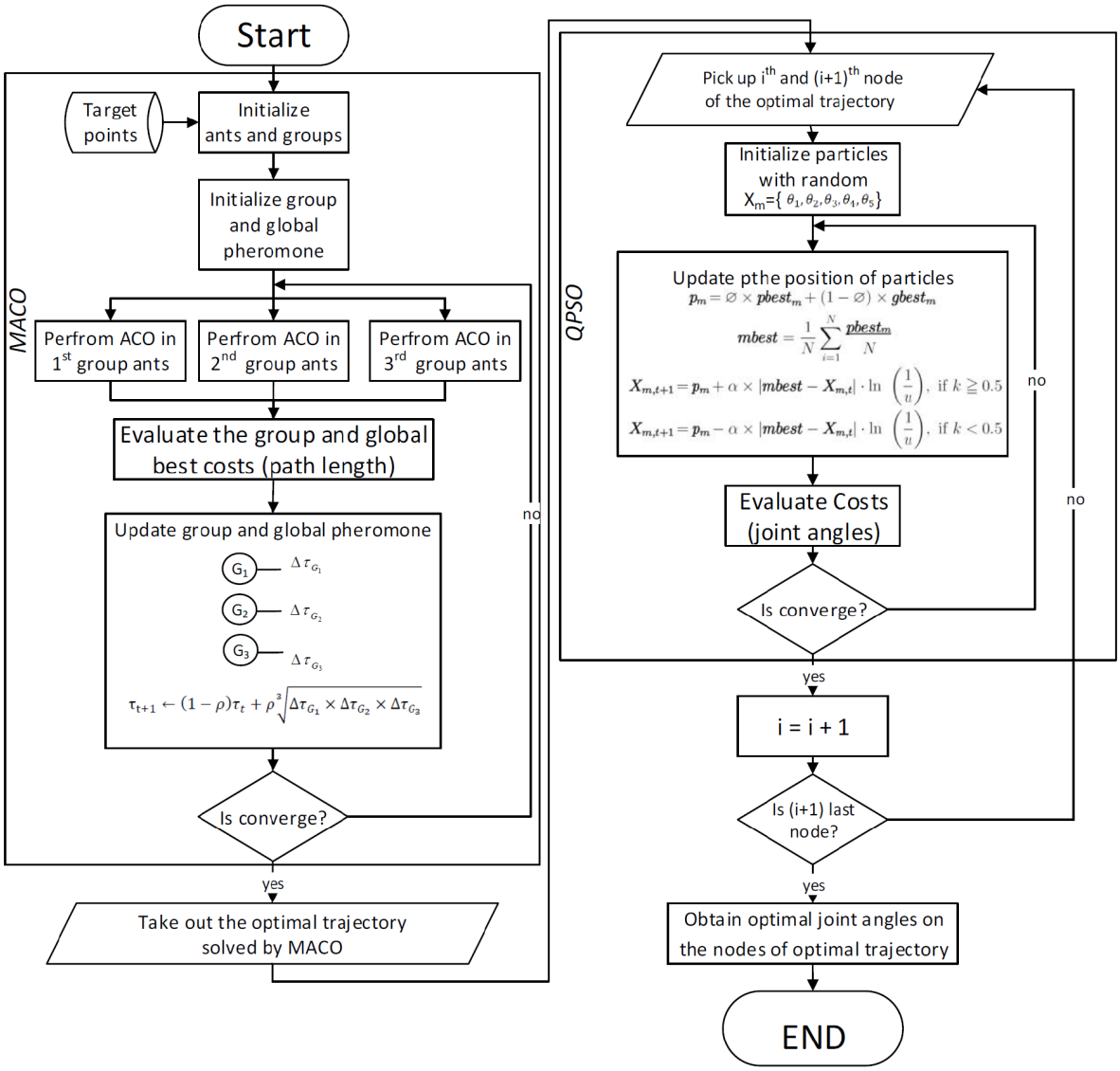


FIGURE 2. Research flowchart

positions is optimally ordered using the MACO algorithm. The second subproblem is the least movements problem, where the QPSO algorithm is used to solve the defined cost function which removes unnecessary movement of the manipulator's joints. The subsequent section discusses and compares the results of this framework.

7. **Results and Analysis.**

7.1. **Optimal trajectory for the constraint conditions.** For robotic manipulator trajectory optimization in the real world, obstacles in the trajectory represent the practical constraint conditions.

The main purpose of this study, therefore, was to determine the optimal trajectory of the end effector of a robotic manipulator. In this study, the obstacles were assumed to be cylinders for simulating a real-world situation. As displayed in Figure 3(a), two obstacle models were constructed: 1) a cylinder centered at (150 mm, 210 mm) with a radius of 20 mm and height of 250 mm and 2) a cylinder centered at (250 mm, 100 mm) with a radius of 20 mm and height of 200 mm. These two simulated obstacles block some

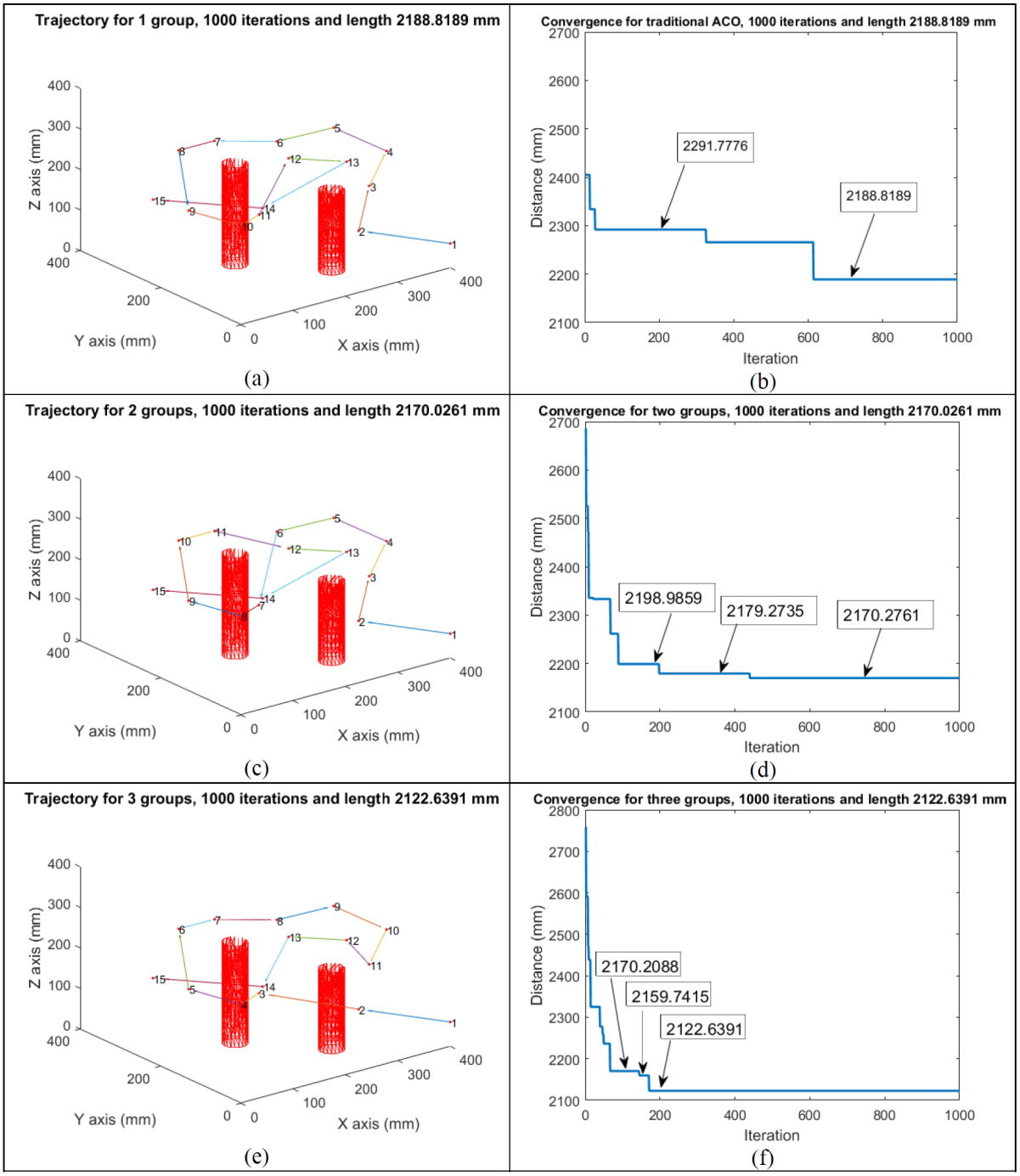


FIGURE 3. Trajectory simulation results: (a) trajectory and (b) convergence of the traditional ACO algorithm, (c) trajectory and (d) convergence of the MACO algorithm for two groups, and (e) trajectory and (f) convergence of the MACO algorithm for three groups

possible routing paths if the paths cross them. We randomly used 16 nodes to determine the optimal path according to Equations (11)-(13).

The initial degenerating parameter of the pheromone ρ was set as 0.4, and ζ was randomly set in the $[0, 1]$ range per iteration. The parameter β was set as $(1 - \zeta)$. For verification, the number of ants $|T_{ant}|$ was set as 120 and the number of iterations was set to 1000. Figures 3(a), 3(c) and 3(e) display the trajectories of the traditional ACO and MACO algorithms for two and three groups. These figures indicate that the ACO and

MACO algorithms allow all the obstacles to be avoided and provide optimal trajectories. Thus, the aforementioned algorithms provide routing and obstacle avoidance capabilities. Figures 3(b), 3(d) and 3(f) display the convergence of the traditional ACO and MACO algorithms (two and three groups). The ACO algorithm, MACO algorithm for two groups, and MACO algorithm for three groups converged at 2188.8189, 2170.0261, and 2122.6391 mm, respectively. The results of the MACO algorithm were superior to those of the traditional ACO algorithm after two obstacles were avoided.

In most practical applications, a reasonable number of iterations is 100-200. Therefore, the slope of convergence and the optimal cost should be considered in the early iterations. The optimal solution of the MACO algorithm was reached in the early iterations and was superior to that of the ACO algorithm because the MACO algorithm involved pheromone information sharing. At the 200th iteration, the optimal costs of the traditional ACO algorithm, MACO algorithm for two groups, and MACO algorithm for three groups were 2291.7776, 2198.9859, and 2122.6391 mm, respectively. Compared with the traditional ACO algorithm, the MACO algorithm (for two and three groups) required fewer iterations, exhibited a 5%-7% smaller path length, and achieved its optimal solution earlier.

Furthermore, the stochastic setup was applied to verifying the MACO algorithm. The values of MACO algorithm parameters such as ζ and ρ were randomly selected in each execution. For verification, the number of groups was set as 1 and 3, and the MACO algorithm was executed 1000 times to record the convergence tendency.

Figures 4(a) and 4(b) present the convergence tendencies of the MACO algorithms for one group (equivalent to the traditional ACO algorithm) and three groups, respectively. These tendencies are evaluated at each iteration based on the average values (μ) and their standard deviation (σ), calculated from the cost values among 1000 executions. The solid line denotes the average values, and the dash lines are the three standard deviation interval ($\mu \pm 3\sigma$), which are considered as the convergence boundaries. As shown in Figure 4, the traditional ACO algorithm continuously converges to optimal solutions until the iteration limitation is reached, whereas the MACO algorithm for three groups has a steeper converging tendency, where the optimal costs are converged at almost 200 iterations. The iteration limitation of the MACO algorithm when reduced to approximately 200 can provide high performance without time-consuming.

As shown in Figure 5, the converged results of the traditional ACO algorithm are all converged in a narrow range of 2100-2300 mm and reflected in a small standard deviation

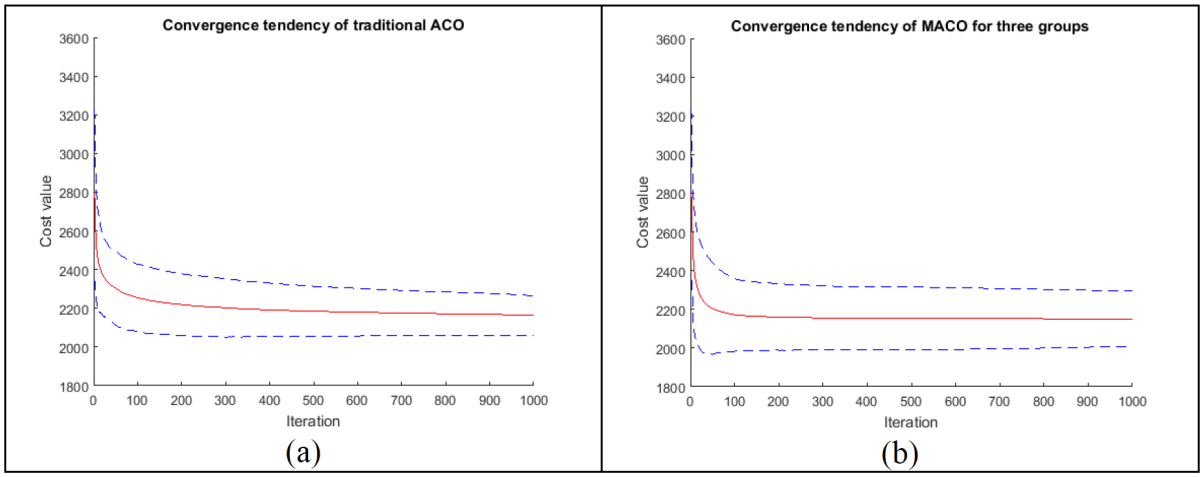


FIGURE 4. Convergence tendency of (a) the traditional ACO algorithm and (b) the MACO algorithm for three groups

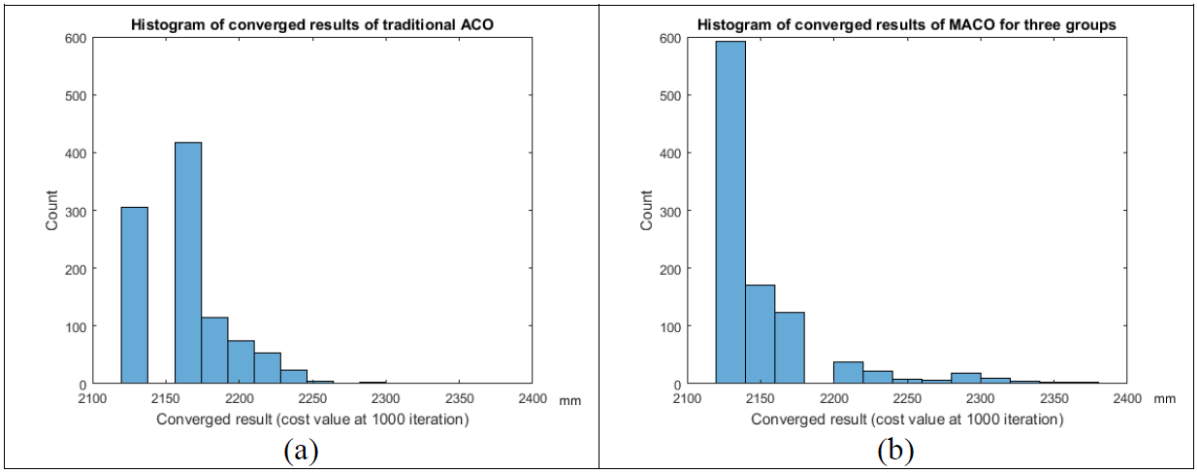


FIGURE 5. Histogram of converged results for (a) the traditional ACO algorithm and (b) the MACO algorithm for three groups

TABLE 2. Joint angle solutions obtained with the QPSO algorithm

No.	$\boldsymbol{\theta}_1$ (deg)	$\boldsymbol{\theta}_2$ (deg)	$\boldsymbol{\theta}_3$ (deg)	$\boldsymbol{\theta}_4$ (deg)	$\boldsymbol{\theta}_5$ (deg)
Initial	−80	30	−30	−45	−80
1	0.000	13.492	−11.141	−11.045	−49.557
Δθ	11.102	43.349	−56.290	−20.698	0.000
2	11.102	56.841	−67.431	−31.743	−49.557
Δθ	33.898	−19.697	56.290	−95.366	0.000
3	45.000	37.144	−11.141	−127.109	−49.557
Δθ	7.400	42.109	−61.970	25.428	0.000
4	52.400	79.253	−73.111	−101.681	−49.557
Δθ	21.225	−65.889	63.665	236.681	80.101
5	73.625	13.364	−9.446	135.000	30.544
Δθ	−1.610	9.870	5.347	−32.906	−80.101
6	72.015	23.234	−4.099	102.094	−49.557
Δθ	−12.979	19.249	−7.067	−113.570	−0.038
7	59.036	42.483	−11.166	−11.476	−49.595
Δθ	−18.027	17.699	0.025	−30.946	0.038
8	41.009	60.182	−11.141	−42.422	−49.557
Δθ	−13.744	−16.203	7.322	44.983	3.425
9	27.265	43.979	−3.819	2.561	−46.132
Δθ	−11.533	−7.812	−2.987	−4.398	11.026
10	15.732	36.167	−6.806	−1.837	−35.106
Δθ	−0.151	−25.995	−4.335	111.130	−14.451
11	15.581	10.172	−11.141	109.293	−49.557
Δθ	13.053	13.449	11.008	−111.730	9.219
12	28.634	23.621	−0.133	−2.437	−40.338
Δθ	13.134	4.688	−8.387	1.871	−9.183
13	41.768	28.309	−8.520	−0.566	−49.521
Δθ	5.310	7.725	−37.810	−11.375	−0.041
14	47.078	36.034	−46.330	−11.941	−49.562
Δθ	26.282	−11.194	35.189	−20.274	0.005
15	73.360	24.840	−11.141	−32.215	−49.557

due to the large population in the group. By contrast, for the MACO algorithm for three groups, most of the converged results are in the range of 2100-2200 mm and reflected in a

smaller modal value of 2122 mm. This means the MACO algorithm obtains a lower cost value in most of the executions than does the traditional ACO algorithm. In general, only a few unstable converged results were noted in the range of 2300-2400 mm. However, the unstable converging may have been due to grouping; it can be improved by simply increasing population size. Nevertheless, the verification results indicate that the converged results are not impacted largely by these randomly selected parameters ζ and ρ, but the number of groups g. The convergence results indicated the superior performance of the MACO algorithm. The optimal solutions of the MACO algorithm were used to operate the manipulator's joints.

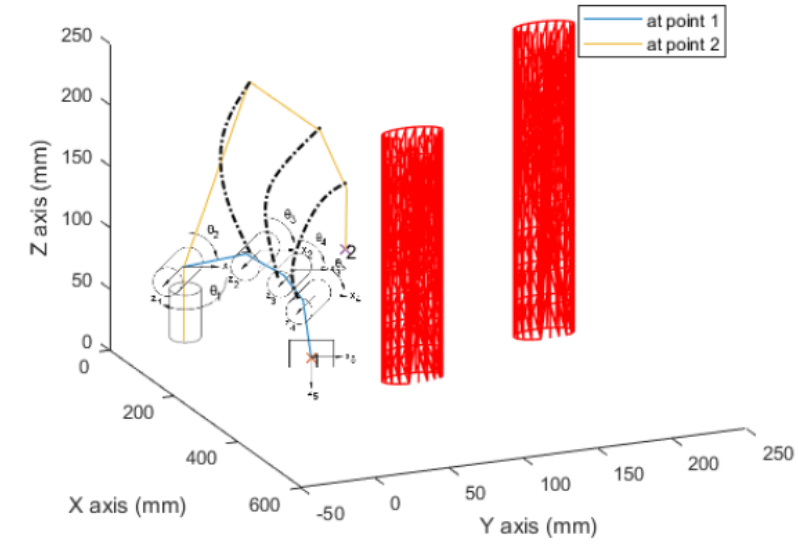
7.2. **Optimal positioning error.** After the optimal path of the end effector was determined using the MACO algorithm, the QPSO algorithm was executed to find the optimal positioning error and joint angles of the manipulator for a given optimal trajectory. Based on Equations (16)-(19), the number of iterations was set as 1000, the total number of particles was set as 100, and the contraction-expansion coefficient α was set as 0.5. Each particle represents the combination of the joint angles θ_1, θ_2, θ_3, θ_4, and θ_5. Figure 4 illustrates the cost function from a predefined start point to the end point (15th point) calculated using the QPSO algorithm. Table 2 presents the joint angle solutions calculated using the QPSO algorithm. Table 3 presents a comparison of the target and actual positions of the 15 points in Table 2.

TABLE 3. Comparison between the target and actual positions

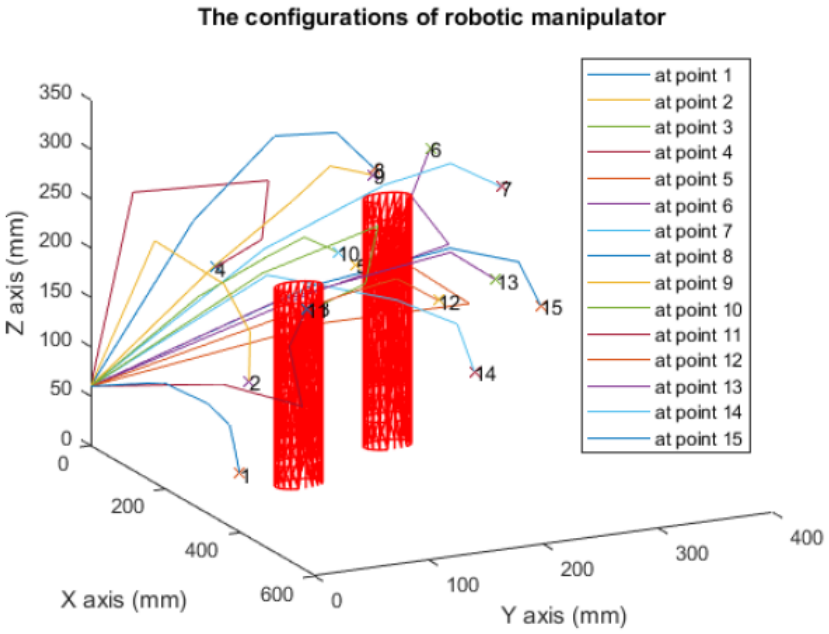
No.	Target position (mm)	Actual position (mm)	Err (mm)
Initial	–	–	
1	400, 0, 60	400.0000, 0.0000, 59.9970	0.0030
2	265, 52, 115	265.0007, 52.0007, 114.9985	0.0018
3	150, 150, 150	149.9996, 149.0007, 150.0006	0.0014
4	67, 87, 181	66.9992, 87.0003, 181.0012	0.0014
5	62, 211, 163	62.0007, 211.0003, 162.9995	0.00095
6	87, 268, 277	87.0001, 267.9976, 277.0026	0.0035
7	180, 300, 253	180.0020, 300.0005, 252.9978	0.0030
8	207, 180, 297	207.0008, 180.0001, 296.9990	0.0013
9	293, 151, 314	292.9982, 151.0010, 314.0007	0.0022
10	355, 100, 258	355.0020, 100.0005, 257.9954	0.0050
11	312, 87, 193	311.9990, 87.0002, 193.0023	0.0025
12	348, 190, 193	347.9987, 190.0030, 193.0007	0.0034
13	290, 259, 190	290.0002, 258.9998, 190.0020	0.0024
14	239, 257, 86	239.0021, 256.9989, 86.0004	0.0024
15	107, 358, 107	106.9970, 358.0013, 106.9985	0.0036

The resolution of the commercial rotary encoder was specified as approximately 0.001°. Therefore, the QPSO algorithm was set to be terminated when the rotation angle difference between two iterations $(\theta_{t+1} - \theta_t)$ was smaller than the resolution or when a predefined number of maximum iterations was reached. The actual positions corresponding to the joint angles calculated with Equation (2) were determined using T_5. Most of the optimal positioning errors were within 10^{-3} mm. The results in Table 3 validate that high-precision positioning control can be achieved for the five-DOF robotic manipulator by using the MACO algorithm.

Figure 6(a) depicts the simulated manipulator and joint movements, and Figure 6(b) displays the complete configurations of each target point. Figures 6(a) and 6(b) present



(a)



(b)

FIGURE 6. Robot configuration: (a) joint movement and (b) complete configurations of each target point

the optimal configuration solution at each target point and the simulated manipulator behavior for reference.

Obstacle avoidance was achieved using the MACO algorithm. The cost function, Equation (5), was minimized; the cost of each joint is presented in Figure 7.

The QPSO algorithm executed on each permutated target positions, and obtained the optimal solutions. The plots in Figure 7 indicate the stability of the convergence. The optimal solutions are almost converged within 200 iterations except for the 5th point. The algorithm includes the stochastic mechanism, and therefore, the optimal solution may not always be within a deterministic iteration; nevertheless, the optimal solutions

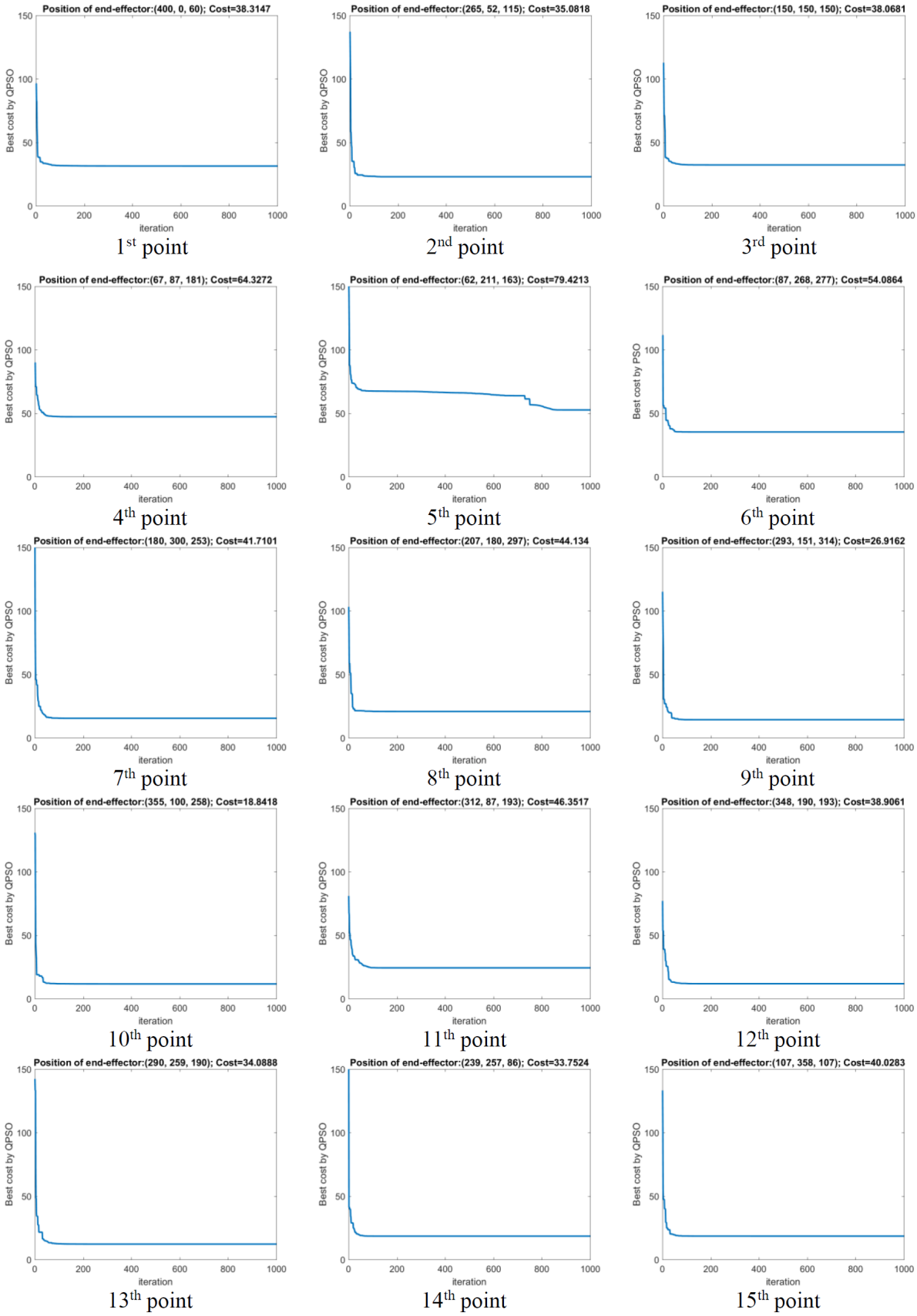


FIGURE 7. Iterative convergent diagrams of the QPSO algorithm for each position of the end effector

of these 15 points are obtained before reaching the maximum iterations. In summary, the QPSO algorithm optimizes the positioning error without time-consuming due to the outperformed convergence.

8. **Conclusion.** This study achieved obstacle avoidance for a five-DOF robotic manipulator in two stages. First, the MACO algorithm was used to search the shortest route around the given points. The MACO algorithm (with two and three groups) could achieve convergence in approximately 200 iterations. Compared with the traditional ACO algorithm, the MACO algorithm required approximately 66% fewer iterations and it reduced the optimal path length by approximately 5%-7%. Obstacle avoidance was considered in the MACO algorithm, and a route was blocked if it crossed any obstacle. Second, the configurations of the manipulator were determined using the QPSO algorithm, which ensured that the end effector reached the target points. A cost function was adopted in the aforementioned algorithm, which minimized the positioning error and movements of the manipulator. The optimal configurations obtained using the QPSO algorithm had a positioning error of 10^{-3} mm, which is suitable for industrial applications. The movement minimization term of the aforementioned algorithm ensured the stability of the trajectory. Finally, the manipulator was simulated to verify the optimal routes and configurations.

The results revealed that the predictive method used in this study effectively enhanced the precision of the robotic manipulator. The proposed algorithm can be widely used to predict the position of multi-axis robotic manipulators in the future.

REFERENCES

[1] J. Cai, J. Zhang and F. Sun, Method of error compensation of 6-axis industrial robot, *J. Mech. Transm.*, vol.7, 2013.

[2] G. Liu, T. Li, Y. Peng and X. Hou, The ant algorithm for solving robot path planning problem, *The 3rd International Conference on Information Technology and Applications (ICITA'05)*, vol.2, pp.25-27, DOI: 10.1109/ICITA.2005.268, 2005.

[3] A. Shukla, E. Singla, P. Wahi and B. Dasgupta, A direct variational method for planning monotonically optimal paths for redundant manipulators in constrained workspaces, *Robot. Auton. Syst.*, vol.61, no.2, pp.209-220, DOI: 10.1016/j.robot.2012.08.012, 2013.

[4] B. Fu et al., An improved A* algorithm for the industrial robot path planning with high success rate and short length, *Robot. Auton. Syst.*, vol.106, pp.26-37, DOI: 10.1016/j.robot.2018.04.007, 2018.

[5] A. Jiang, X. Yao and J. Zhou, Research on path planning of real-time obstacle avoidance of mechanical arm based on genetic algorithm, *The Journal of Engineering*, no.16, pp.1579-1586, DOI: 10.1049/joe.2018.8266, 2018.

[6] P. T. Zacharia, E. K. Xidias and N. A. Aspragathos, Task scheduling and motion planning for an industrial manipulator, *Robotics and Computer-Integrated Manufacturing*, vol.29, no.6, pp.449-462, DOI: 10.1016/j.rcim.2013.05.002, 2013.

[7] 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, DOI: 10.1016/j.amc.2014.10.101, 2015.

[8] P. Kalra, P. B. Mahapatra and D. K. Aggarwal, An evolutionary approach for solving the multimodal inverse kinematics problem of industrial robots, *Mech. Mach. Theory*, vol.41, no.10, pp.1213-1229, DOI: 10.1016/j.mechmachtheory.2005.11.005, 2006.

[9] S. Tabandeh, C. Clark and W. Melek, A genetic algorithm approach to solve for multiple solutions of inverse kinematics using adaptive niching and clustering, *IEEE International Conference on Evolutionary Computation*, pp.1815-1822, DOI: 10.1109/CEC.2006.1688527, 2006.

[10] A. El-Sherbiny, M. A. Elhosseini and A. Y. Haikal, A comparative study of soft computing methods to solve inverse kinematics problem, *Ain Shams Eng. J.*, DOI: 10.1016/j.asej.2017.08.001, 2017.

[11] M. Wei, M. Tong, B. Zi and S. Tang, Pipeline robot localization system based on PSO-BFGS hybrid algorithm, *Chin. J. Sci. Instrum.*, vol.11, 2012.

[12] S. Chiaradonna, F. Di Giandomenico and N. Murru, On enhancing efficiency and accuracy of particle swarm optimization algorithms, *International Journal of Innovative Computing, Information and Control*, vol.11, no.4, pp.1165-1189, 2015.

[13] T. L. Dang, T. Cao and Y. Hoshino, A proposed PSOseed2 algorithm for training hardware-based and software-based neural networks, *International Journal of Innovative Computing, Information and Control*, vol.13, no.4, pp.1205-1219, 2017.

[14] Q. Zhang, Y. Wei and W. Song, Two strategy cooperative particle swarm optimization algorithm with independent parameter adjustment and its application, *International Journal of Innovative Computing, Information and Control*, vol.16, no.4, pp.1203-1223, 2020.

[15] H. Huang, C. Chen and P. Wang, Particle swarm optimization for solving the inverse kinematics of 7-DOF robotic manipulators, *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp.3105-3110, DOI: 10.1109/ICSMC.2012.6378268, 2012.

[16] W. Zhou, W. Liao, W. Tian, S. Wan and Y. Liu, Method of industrial robot accuracy compensation based on particle swarm optimization neural network, *Zhong Guo Ji Xie Gong Cheng China Mech. Eng.*, vol.24, pp.174-179, DOI: 10.3969/j.issn.1004-132X.2013.02.007, 2013.

[17] R. Praserttaweelap, S. Kaitwanidvilai and H. Aoyama, Safety path planning with obstacle avoidance using particle swarm optimization for AGV in manufacturing layout, *International Journal of Innovative Computing, Information and Control*, vol.15, no.1, pp.351-368, 2019.

[18] D. T. Pham, M. Castellani and A. A. Fahmy, Learning the inverse kinematics of a robot manipulator using the bees algorithm, *The 6th IEEE International Conference on Industrial Informatics*, pp.493-498, DOI: 10.1109/INDIN.2008.4618151, 2008.

[19] R. V. Mayorga and P. Sanongboon, Inverse kinematics and geometrically bounded singularities prevention of redundant manipulators: An artificial neural network approach, *Robot. Auton. Syst.*, vol.53, no.3, pp.164-176, DOI: 10.1016/j.robot.2005.09.011, 2005.

[20] A. T. Hasan, A. M. S. Hamouda, N. Ismail and H. M. A. A. Al-Assadi, An adaptive-learning algorithm to solve the inverse kinematics problem of a 6 D.O.F serial robot manipulator, *Adv. Eng. Softw.*, vol.37, no.7, pp.432-438, DOI: 10.1016/j.advengsoft.2005.09.010, 2006.

[21] C. Lin, H. Wang, M. Fu, J. Yuan and D. Yu, A convolution neural network based obstacle avoiding method for unmanned underwater vehicle, *ICIC Express Letters*, vol.13, no.11, pp.1079-1086, 2019.

[22] S. J. Raudys and A. K. Jain, Small sample size effects in statistical pattern recognition: Recommendations for practitioners, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol.13, no.3, pp.252-264, DOI: 10.1109/34.75512, 1991.

[23] R. L. Figueroa, Q. Zeng-Treitler, S. Kandula and L. H. Ngo, Predicting sample size required for classification performance, *BMC Med. Inform. Decis. Mak.*, vol.12, no.1, DOI: 10.1186/1472-6947-12-8, 2012.

[24] A. Vabalas, E. Gowen, E. Poliakoff and A. J. Casson, Machine learning algorithm validation with a limited sample size, *PLOS ONE*, vol.14, no.11, DOI: 10.1371/journal.pone.0224365, 2019.

[25] M. Dorigo and C. Blum, Ant colony optimization theory: A survey, *Theor. Comput. Sci.*, vol.344, no.2, pp.243-278, DOI: 10.1016/j.tcs.2005.05.020, 2005.

[26] W.-J. Chen, L.-J. Jheng, Y.-T. Chen and D.-F. Chen, Optimization path programming using improved multigroup ant colony algorithms, in *Intelligent Technologies and Engineering Systems. Lecture Notes in Electrical Engineering*, J. Juang and Y.-C. Huang (eds.), New York, NY, Springer, 2013.

[27] J. Kennedy and R. Eberhart, Particle swarm optimization, *Proc. of IEEE International Conference on Neural Networks*, vol.4, pp.1942-1948, DOI: 10.1109/ICNN.1995.488968, 1995.

[28] D. Tang, Y. Cai, J. Zhao and Y. Xue, A quantum-behaved particle swarm optimization with memetic algorithm and memory for continuous non-linear large scale problems, *Inf. Sci.*, vol.289, pp.162-189, DOI: 10.1016/j.ins.2014.08.030, 2014.