Robotica (2024), **42**, pp. 833–845
doi:10.1017/S0263574723001789

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Kinematics inverse solution of assembly robot based on improved particle swarm optimization

Shixiong Zhang, Ang Li, Jianxin Ren and Ruilong Ren

School of Mechanical and Electrical Engineering, Henan University of Technology, Zhengzhou, China
Corresponding author: Shixiong Zhang; Email: cehon@126.com

Received: 10 August 2023; **Revised:** 19 November 2023; **Accepted:** 6 December 2023; **First published online:** 4 January 2024

Keywords: assembly robot; inverse kinematics; particle swarm algorithm; nonlinear weight strategy

Abstract

Inverse kinematics of robot is the basis of robot assembly, which directly determines the pose of robot. Because the traditional inverse solution algorithm is limited by the robot topology structure, singular pose and inverse solution accuracy, it affects the use of robots. In order to solve the above problems, an improved particle swarm optimization (PSO) algorithm is proposed to solve the inverse problem of robot. This algorithm initializes the particle population based on joint angle limitations, accelerating the convergence speed of the algorithm. In order to avoid falling into local optima and premature convergence, we have proposed a nonlinear weight strategy to update the speed and position of particles, enhancing the algorithm's search ability, in addition introducing a penalty function to eliminate particles exceeding joint limits. Finally, the positions of common points and singular points are selected on PUMA 560 robot and redundant robot for inverse kinematics simulation verification. The results show that, compared with other algorithms, the improved PSO algorithm has higher convergence accuracy and better convergence speed in solving the inverse solution, and the algorithm has certain universality, which provides a new solution for the inverse kinematics solution of the assembly robot.

1. Introduction

Since the beginning of the 21st century, robots have played an increasingly important role in the industrial field. At present, robots are widely used in industrial, medical, living and other aspects, of which a considerable number of robots are used in assembly tasks in industry. For the assembly task, Xu et al. [1] used a binocular camera to roughly measure the pose of the target hole and then used a monocular camera to improve the image quality to determine the exact pose of the hole. Song et al. [2] use training learning and Gaussian strategy to implement the assembly process and use the compliant control method to complete the assembly in the case of uncertain parameters. The above methods are not only expensive but also require feedback based on the actual assembly results, which can easily damage the assembly parts. By utilizing robot kinematics, assembly trajectory planning can be accomplished without these drawbacks.

Usually, the assembly operation transforms the position information of the hole into the target position of the robot end-effector. Then, the joint angle reaching the target position is obtained by using the inverse kinematics solution. However, the inverse solution of the robot is a nonlinear problem, with multiple groups of joint angles in the same attitude, from the Cartesian space to the joint space. For serial robots, the methods to obtain the inverse solutions are numerical method, analytical method and intelligent optimization algorithm. Among them, the analytic method requires the robot to meet the Pieper criterion and then select the appropriate inverse solution according to the operation needs; the numerical method has error accumulation, which increases the complexity. Meanwhile, the result is dependent on the selection of the initial value. However, with the development of artificial intelligence, many scholars have transformed nonlinear problems into optimization problems and utilized various

optimization algorithms to solve them. Examples of these algorithms include genetic algorithm [3], particle swarm optimization (PSO) [4] and neural network algorithm [5]. The intelligent algorithm not only compensates for the mentioned limitations but also operates without constraints posed by the robot model. Moreover, it exhibits strong versatility and high solution accuracy.

Based on the PSO, Netjinda et al. [6] introduce a new mechanism to increase the diversity of the algorithm and obtain the optimal solution in the test function experiment. Nagata et al. [7] use neural network to generate temporary sets and use temporary sets and original sets, which can effectively learn the inverse kinematics of the robot and shorten the convergence time. Zhang Libo et al. [8] utilized an improved genetic algorithm for the inverse solution of a mechanical arm that did not meet the proper criterion, reducing pose errors and enabling precise control of position and posture. Shen Xiaolong et al. [9] use the improved differential evolution algorithm to solve the robot inverse kinematics, which improves the convergence speed and solution accuracy of the algorithm. Jing Liang et al. [10] proposed an algorithm based on an improved Kalman PSO. Compared to similar modeling methods, it demonstrates better predictive accuracy and exhibits superior generalization performance. Chen Mei et al. [11] used the chicken swarm algorithm to solve the inverse kinematics and used it for welding operations, which improved the accuracy of the attitude error.

Some scholars use the particle swarm algorithm and the improved algorithm to solve the inverse solution. Liang et al. [12] introduce the covariance guidance strategy to adjust the particle motion through the relationship between the fitness value of the particle and the Euclidean distance, which obviously improves the convergence speed and accuracy of the algorithm. Based on the particle swarm algorithm, Netjinda et al. [6] introduce a new mechanism to increase the diversity of the algorithm and obtain the optimal solution in the test function experiment. Deng et al. [13] used the adaptive weight particle swarm algorithm to solve the inverse solution of PUMA 560 robot and proposed a special boundary processing method, which improves the situation where the algorithm falls into the local optimum. Liu et al. [14] divided the particle swarm algorithm into multiple populations, searched the space simultaneously, learned the worst particles from each other, improved the search speed of the algorithm and verified the inverse solution effect on the UR 5 mechanical arm. Zhao et al. [15] used adaptive weight to update the particle speed and limit the speed, after which the inverse solution was verified in PUMA 560 robot and redundant mechanical arm. Liu et al. [16] divided the population into multiple subgroups, introduced nonlinear dynamic weights to adjust the particle velocity and used the migration operator to exchange the worst individuals in the adjacent group with the optimal individual after each iteration to increase the diversity of the group.

However, the improvement theory of the above intelligent optimization algorithm is complex. Based on the advantages of adaptive weights and linear weights, a nonlinear weight particle swarm algorithm is proposed to solve the inverse kinematics of PUMA 560 robots. At the same time, compared with other optimization algorithms, the simulation results show that the improved particle swarm algorithm obviously improves the convergence accuracy. Later, it is used to solve the strange position of the robot, which makes up for the shortcomings of the numerical solution method without increasing the complexity of the algorithm. Finally, a redundant robotic arm is used to verify the generality of the inverse solution of the proposed optimization algorithm.

The rest of this paper is organized as follows. Section 2 introduces the kinematics modeling of the robot and the application analysis of the assembly operation. Section 3 designs the fitness function. Section 4 introduces PSO and its improvement. Section 5 sets up a variety of test schemes, uses a variety of optimization algorithms for comparative experiments and analyzes the effects of the algorithms in convergence, solution accuracy and operation time. Finally, the conclusion of this paper is given.

2. Kinematic modeling and analysis

In this paper, we use the six-degree-of-freedom robot PUMA 560 as the research subject and establish the link coordinate system according to the MDH method. Figure 1 and Table I demonstrate the obtained DH parameters of the robot.

https://doi.org/10.1017/S0263574723001789 Published online by Cambridge University Press

Table I. *Robot DH parameter table.*

Joint^i	θ_i/(rad)	d_i/m	a_i/m	α_i/(rad)	**Joint angle boundary/rad**
1	θ_1	0	0	0	[− 2.9409, 2.9409]
2	θ_2	0.1491	0	$-\pi/2$	[− 2.5045, 0.7592]
3	θ_3	0	0.4318	0	[− 2.1555, 1.3963]
4	θ_4	0.4331	0.0203	$-\pi/2$	[− 5.0615, 5.0615]
5	θ_5	0	0	$\pi/2$	[− 3.9968, 3.9968]
6	θ_6	0	0	$-\pi/2$	[− 2.9409, 2.9409]

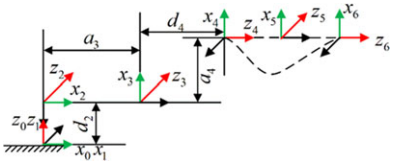

Figure 1. *The linkage coordinate system.*

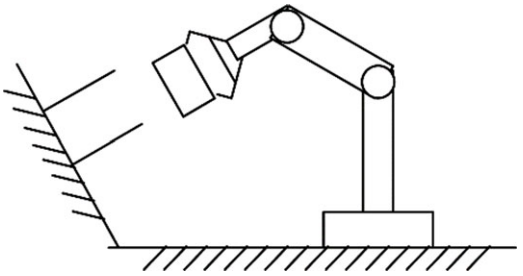

Figure 2. *Assembly operation scenario.*

Then, establish the transformation relationship between axes by rotating the x axis, translating the x axis, and then rotating the z axis and translating the z axis:

$$ {}^{i-1}\boldsymbol{T}_i = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_i \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -d_i\sin\alpha_{i-1} \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & d_i\cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1} $$

Then, the transformation relationship between the robot end and the base is established according to the formula (1):

$$ {}^0\boldsymbol{T}_6 = {}^0\boldsymbol{T}_1\,{}^1\boldsymbol{T}_2\,{}^2\boldsymbol{T}_3\,{}^3\boldsymbol{T}_4\,{}^4\boldsymbol{T}_5\,{}^5\boldsymbol{T}_6 = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{P} \\ 000 & 1 \end{pmatrix} \tag{2} $$

In the equation, $\boldsymbol{R}$ is a 3×3 matrix representing the orientation of the robot, and $\boldsymbol{P}$ is a 3×1 vector representing the position of the robot.

The scenario of robot assembly operation can be simplified as shown in Fig. 2. Assuming that the end of the robot and the installation axis are regarded as one, it is necessary to ensure that the position and direction of the end and the hole overlap highly, so as to avoid collision and affect the quality of the workpiece.

https://doi.org/10.1017/S0263574723001789 Published online by Cambridge University Press

Corresponding the specific information of the hole $(x, y, z, \alpha, \beta, \gamma)$ to the end attitude of the robot, the assembly problem can be transformed into the inverse solution problem of the robot.

$$f^{-1}\begin{pmatrix} \boldsymbol{R} & \boldsymbol{P} \\ 000 & 1 \end{pmatrix} = \boldsymbol{q}\left(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\right) \tag{3}$$

According to literature [17], the inverse solution exists only when the robot satisfies the Piper principle. In order to enable the general machine to find the inverse solution of the robot, the optimization algorithm is used to solve the inverse solution.

3. Fitness function design

The mounting hole's positional data are transformed into the robot's base coordinate system using MATLAB's transl and rpy2tr functions, which are part of Professor Peter Corke's robotic toolbox. By applying these functions to combine the position and rotation angles, we can obtain the desired pose matrix $\boldsymbol{T}_g$. Among them, the transl function transforms the X, Y and Z information of holes in Cartesian space into a homogeneous transformation matrix. The rpy2tr function is usually used to convert Euler angles (pitch angle, roll angle and yaw angle) into elements in the rotation matrix or the direction cosine matrix in the homogeneous transformation matrix. The jtraj function generates a set of joint angles according to the initial joint position and the target joint position to ensure that the trajectory is smooth. Please note that Professor Peter Corke's robotic toolbox is not a standard component of MATLAB and needs to be separately installed, as mentioned earlier.

In order to realize the shaft hole assembly, the current pose of the robot $\boldsymbol{T}_p$ is required not to exceed the limit of the joint Angle, which is simplified to a mathematical model:

$$f_1 = \begin{cases} \boldsymbol{q}_{\min} \le \boldsymbol{q} \le \boldsymbol{q}_{\max} \\ \left\| \boldsymbol{T}_g - \boldsymbol{T}_p \right\| = 0 \end{cases} \tag{4}$$

When the optimization algorithm is used to solve the inverse, the formula (4) can be regarded as the objective function and the minimum value. Since the last row of the target matrix $\boldsymbol{T}_g$ remains constant, the objective function can be organized as:

$$f_1 = \begin{cases} \min\left(\left\| \boldsymbol{R}_g - \boldsymbol{R}_p \right\| + \left\| \boldsymbol{P}_g - \boldsymbol{P}_p \right\|\right) \\ \boldsymbol{q}_{\min} \le \boldsymbol{q} \le \boldsymbol{q}_{\max} \end{cases} \tag{5}$$

At the same time, in the assembly operation, it is necessary to ensure that the robot joint Angle change is gentle, and there is no joint direction change phenomenon, which affects the quality of the workpiece. Therefore, it is also necessary to introduce the joint angle variation as a fitness function:

$$f_2 = \sum_{i=1}^{6} \left(\left\| \theta_i\left(t\right) - \theta_i\left(t-1\right) \right\|\right) \tag{6}$$

Among them, f_2 represents the sum of joint angle changes.

In conclusion, we design the fitness function as follows when utilizing the intelligent optimization algorithm to solve the kinematic inverse solution of the assembly robot:

$$f = f_1 + f_2 \tag{7}$$

4. PSO algorithm and its improvement

PSO is a stochastic optimization algorithm based on population. Since its proposal, it has undergone numerous formal changes and has been applied in various target optimization scenarios [18]. The principle behind PSO lies in the interaction of information among particles within the population, enhancing

https://doi.org/10.1017/S0263574723001789 Published online by Cambridge University Press

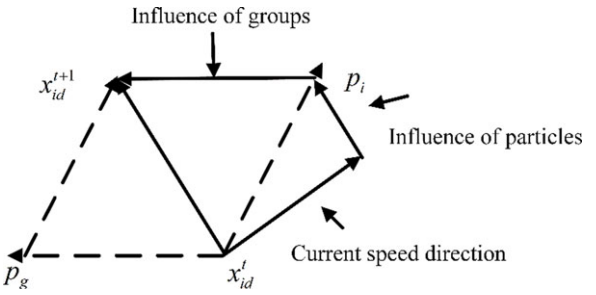

Figure 3. *The principle of PSO.*

their ability to search for optimal solutions. As depicted in Fig. 3, each particle possesses its own position and speed, influenced by both the group and individual interactions, which collectively guide their search toward the optimal location.

You can organize this into a mathematical model:

$$\begin{cases} \boldsymbol{v}_{iD}^{t+1} = \omega^* \boldsymbol{v}_{iD}^{t} + c_1 r_1 \left(\boldsymbol{p}_{iD}^{t} - \boldsymbol{x}_{iD}^{t}\right) + c_2 r_2 \left(\boldsymbol{p}_{gD}^{t} - \boldsymbol{x}_{iD}^{t}\right) \\ \boldsymbol{x}_{iD}^{t+1} = \boldsymbol{x}_{iD}^{t} + \boldsymbol{v}_{iD}^{t+1} \end{cases} \tag{8}$$

Among them, ω is the inertia weight, c_1 and c_2 are the learning factors that determine the influence of the current velocity, the particle-to-particle interaction, and the swarm-to-swarm interaction, respectively, and r_1 and r_2 are random numbers between 0 and 1.

The larger the inertial weight value, the stronger the global search ability; otherwise, the search ability is weakened and easy to converge near the optimal solution. However, fixed inertial weights can result in poor search performance, leading to local optima and premature convergence phenomena. Therefore, several variants of the PSO algorithm are proposed, such as the adaptive PSO (APSO) and the linear weight particle swarm optimization (LWPSO).

Based on the basis of the weight strategy, the nonlinear weight PSO (NWPSO) is proposed, the model is (9):

$$\omega^* = \begin{cases} \omega_z - \dfrac{\omega_z - \omega_{\min}}{t_z}^* t & t < t_z \\ \omega_{\min} + \dfrac{\omega_{\max} - \omega_{\min}}{t_{\max} - t_z}^* (t - t_z) & t \geq t_z \end{cases} \tag{9}$$

Among them, ω_z is the intermediate value of the inertia weight, which is set based on empirical experience, and t_z is the intermediate value of the iteration number. By introducing two parameters, the advantage of linear weight change can be maintained before and after the weight change. At the beginning of iteration, the weight quickly becomes smaller, the particle speed update is accelerated, and the convergence speed is improved. When the optimum is about to be reached, change the weight change mode to improve the convergence accuracy. The flow of the PSO algorithm for inverse kinematics is shown in Fig. 4.

However, when the particle is updated, the particle will exceed the joint angle limit of the robot, so the penalty function is introduced to eliminate the unqualified particles.

$$f = \text{fitness} + 1000 \tag{10}$$

Among them, fitness is the fitness value of the particle.

5. Simulation validation

In this paper, the improved PSO algorithm is applied to solve the inverse kinematics of the PUMA 560 robot, and the optimization results are compared with several other algorithms. The parameter settings

https://doi.org/10.1017/S0263574723001789 Published online by Cambridge University Press

Table II. *Algorithm parameters.*

Algorithm	ω	ω_{max}	ω_{min}	c_1	c_2	ω_z	t_z
PSO	0.9	–	–	1.5	1.5	–	–
APSO	–	0.9	0.4	1.5	1.5	–	–
Starling-PSO	0.9	0.9	0.4	1.5	1.5	–	–
LWPSO	–	0.9	0.4	1.5	1.5	–	–
NWPSO	–	0.9	0.4	1.5	1.5	0.6	t_z
MPPSO	ω_1	–	–	1.5	1.5	–	–

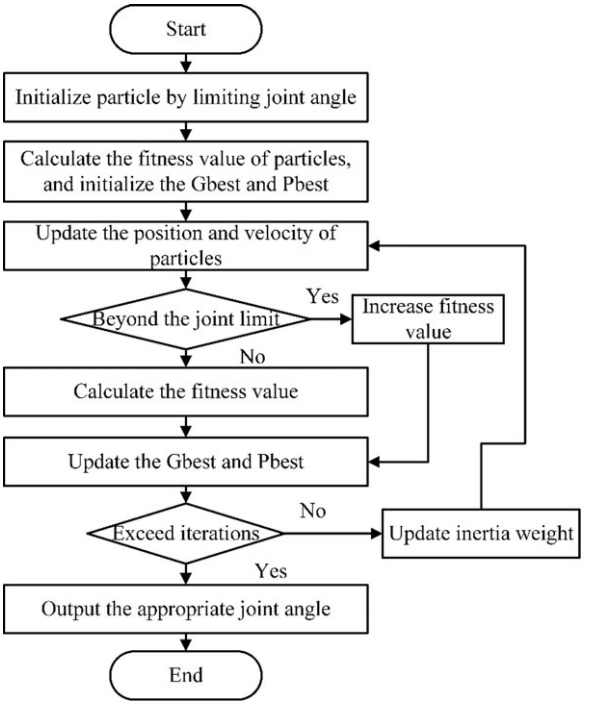

Figure 4. *Flow chart of the PSO.*

for the algorithm are shown in Table II. PSO refers to the standard PSO algorithm, APSO refers to the adaptive PSO algorithm, Starling-PSO refers to the algorithm used in reference [12], LWPSO refers to the linear weight PSO algorithm, NWPSO refers to the nonlinear weight PSO algorithm, MPPSO refers to the multi-population PSO algorithm, and GA refers to the genetic algorithm. The population size and iteration number for the genetic algorithm are the same as those for the PSO algorithm, with a crossover rate of 0.8 and a mutation rate of 0.05.

In Table II, the value of ω_1 ranges from 0.2 to 1.0 in increments of 0.2. t_z is a parameter introduced to improve the PSO, which modifies the way the weight changes near the iteration where the optimal solution is reached to avoid the algorithm being trapped in a local optimum state.

$$t_z = 10 + \text{round}\,(10^{*}\text{rand}\,(1, 1)) \tag{11}$$

Among them, round is a circular function and rand is a random function.

Then, MATLAB R2021b is used to model the robot and multiple optimization algorithms on a Dell desktop processor configured as i5-10500, 3.10 GHZ and 16G memory. To verify the performance of the improved PSO, multiple sets of experiments are set in this paper.

https://doi.org/10.1017/S0263574723001789 Published online by Cambridge University Press

Table III. *The fitness function values corresponding to the algorithm.*

Algorithm	Best	Worst	Mean	Var
PSO	1.1456×10^{-1}	3.7025×10^{-1}	2.0601×10^{-1}	1.5488×10^{-2}
APSO	5.2237×10^{-8}	5.7418×10^{-2}	1.2666×10^{-2}	5.8227×10^{-4}
Starling-PSO	3.4289×10^{-3}	3.8675×10^{-1}	8.9700×10^{-2}	2.3483×10^{-2}
NWPSO	2.1368×10^{-5}	3.9435×10^{-3}	5.4327×10^{-4}	1.5920×10^{-6}
MPPSO	8.0046×10^{-4}	9.8221×10^{-2}	5.2433×10^{-2}	3.5690×10^{-3}
LWPSO	6.4494×10^{-4}	6.8697×10^{-2}	2.5488×10^{-2}	1.3522×10^{-3}
GA	5.3015×10^{-2}	3.6282×10^{-1}	2.2112×10^{-1}	3.8891×10^{-2}

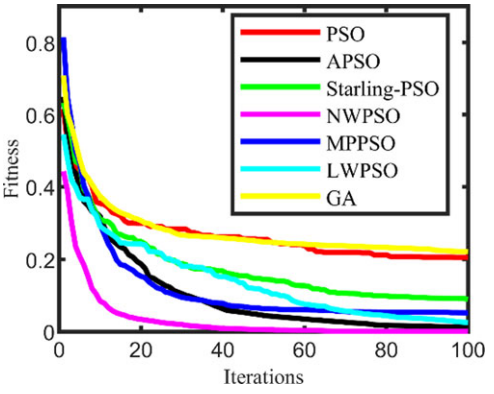

Figure 5. *Average fitness function curve.*

5.1. Analysis of the algorithm performance at the random points

In the experiment, the position information of the installation hole is set as $[0.5, 0.2, 0.3, 0, \pi/2, \pi/3]$, which is transformed into a homogeneous transformation matrix, that is, the target matrix $\boldsymbol{T}_g$:

$$\boldsymbol{T}_g = \begin{pmatrix} 0.9995 & -0.0137 & 0.0274 & 0.5000 \\ 0.0137 & 0.9999 & 0.0004 & 0.2000 \\ -0.0274 & 0 & 0.9996 & 0.3000 \\ 0 & 0 & 0 & 1.0000 \end{pmatrix} \tag{12}$$

Afterward, the number of particles was set to 150, with 100 iterations, and the remaining parameters are detailed in Table III. Utilizing f_1 as the fitness function, each algorithm was executed 10 times in MATLAB, capturing both the optimal and mean values of the results. As shown in Table IV, the NWPSO algorithm proposed in this paper proves to be feasible for solving the inverse robot solution. While its optimal optimization effect may not be as impressive as the APSO algorithm, in comparison to other algorithms, it significantly reduces the robot's attitude error. Furthermore, the algorithm exhibits the smallest average and variance values, increasing by two orders of magnitude. This demonstrates that the NWPSO algorithm is well-suited for solving inverse problems, providing the best optimal stability while maintaining solution accuracy.

Combined with Figs. 5 and 6, the NWPSO algorithm obviously accelerates the convergence rate of the algorithm, and the average fitness function of the algorithm is the smallest, that is, the error value of the robot from the target attitude.

https://doi.org/10.1017/S0263574723001789 Published online by Cambridge University Press

Table IV. *Optimization effects of the algorithm at the singularity.*

Algorithm	Best	Worst	Mean	Var
PSO	3.7545×10^{-2}	2.4053×10^{-1}	1.6473×10^{-1}	1.9822×10^{-2}
APSO	6.0325×10^{-5}	7.7603×10^{-2}	2.7101×10^{-2}	1.6249×10^{-3}
Starling-PSO	1.1496×10^{-3}	4.3817×10^{-2}	1.7008×10^{-2}	4.0423×10^{-4}
NWPSO	4.5763×10^{-3}	2.6756×10^{-2}	1.7067×10^{-2}	2.0057×10^{-4}
MPPSO	7.6440×10^{-3}	2.5440×10^{-1}	6.4979×10^{-2}	8.3288×10^{-3}
LWPSO	4.7738×10^{-4}	1.3389×10^{-1}	3.0069×10^{-2}	3.1069×10^{-3}
GA	8.8452×10^{-3}	1.1879×10^{-1}	4.8055×10^{-2}	2.5254×10^{-3}

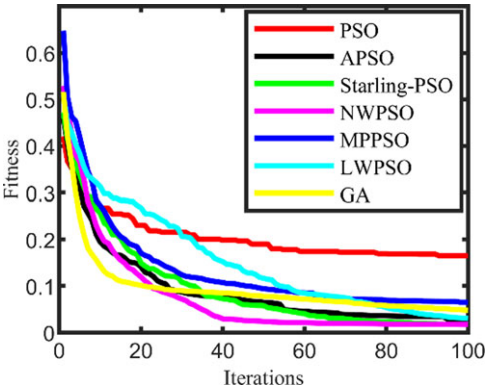

Figure 6. *The mean fitness function curves at the singular points.*

5.2. Algorithm analysis at the singularity points

To further validate the optimization effectiveness of the algorithm, a specific target point in an unconventional position is selected. The algorithm parameters and fitness function remain unchanged. Initially, all the joint points of the robot are obtained through traversal, and then, the corresponding Jacobian matrix is derived. Subsequently, the Jacobian matrix's full-rank condition is checked to identify singular position points. Using the target pose matrix with positive kinematics: $[\pi/4, \pi/6, \pi/3, \pi/5, 0, \pi/2]$:

$$\boldsymbol{T}_g = \begin{pmatrix} 0.5721 & -0.4156 & -0.7071 & -0.1473 \\ -0.5721 & 0.4156 & 0.7071 & 0.0636 \\ 0.5878 & 0.8090 & 0 & -0.2362 \\ 0 & 0 & 0 & 1.0000 \end{pmatrix} \tag{13}$$

Then, the algorithm parameters and other information are set to be consistent with 5.1, and the optimal value and average value of the running results are also taken.

The NWPSO algorithm reduces the error at the singularity to 0.0046 m, which is less effective than the APSO and LWPSO algorithms. However, the overall effect of NWPSO algorithm is the same as that of the general point. The algorithm converges in approximately 45 generations, making it the fastest. Meanwhile, the optimization result of the algorithm has the least volatility, and the average result can constrain the attitude error to 0.0171 m.

5.3. Analysis of the algorithm performance in the assembly process

In the assembly process, not only the end attitude of the robot is required to coincide with the installation hole but also the joint Angle is constrained from large fluctuations. Therefore, the above universal point

https://doi.org/10.1017/S0263574723001789 Published online by Cambridge University Press

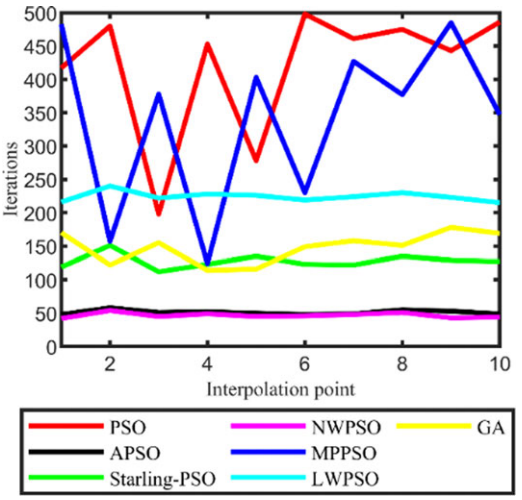

Figure 7. *Comparison of iteration numbers graph.*

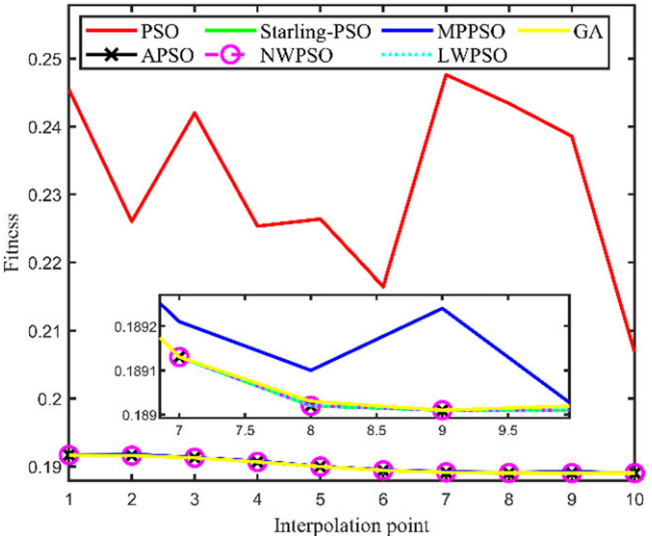

Figure 8. *Fitness function.*

as the termination point introduces the amount of position change to simulate the actual position of the hole and set the starting point of the hole.

$$\begin{cases} x_{\text{start}} = x_{\text{end}} + \Delta x \\ y_{\text{start}} = y_{\text{end}} + \Delta y \\ z_{\text{start}} = z_{\text{end}} + \Delta z \end{cases} \tag{14}$$

In the equation: Δx: 0.1 m; Δy: 0.05 m; Δz: 0.1 m.

Selected f as the objective function, the number of iterations was set to 500 and the population individuals to 150, and 10 intermediate points were set using the jtraj function of MATLAB to minimize the joint angles between continuities.

From Fig. 7, it can be seen that the NWPSO algorithm proposed in the paper has the fastest convergence speed, all converging around 50 iterations. Compared with other algorithms, it significantly improves the convergence speed of the algorithm. Combined with Fig. 8, it can be concluded that the

https://doi.org/10.1017/S0263574723001789 Published online by Cambridge University Press

Table V. *Algorithm optimization effects of redundant robotic arms.*

Algorithm	Best	Worst	Mean	Var
PSO	7.4827×10^{-2}	3.1866×10^{-1}	1.8434×10^{-1}	2.0492×10^{-2}
APSO	6.6064×10^{-3}	1.8112×10^{-1}	5.2667×10^{-2}	4.3274×10^{-3}
Starling-PSO	3.5243×10^{-4}	1.7216×10^{-1}	5.2891×10^{-2}	5.3369×10^{-3}
NWPSO	1.8362×10^{-10}	7.7236×10^{-2}	2.6473×10^{-2}	1.3659×10^{-3}
MPPSO	2.8045×10^{-2}	1.2822×10^{-1}	6.4437×10^{-2}	2.4437×10^{-3}
LWPSO	1.9465×10^{-9}	3.9852×10^{-2}	2.1080×10^{-2}	6.2121×10^{-4}
GA	2.2610×10^{-2}	2.3702×10^{-1}	1.1207×10^{-1}	1.4279×10^{-2}

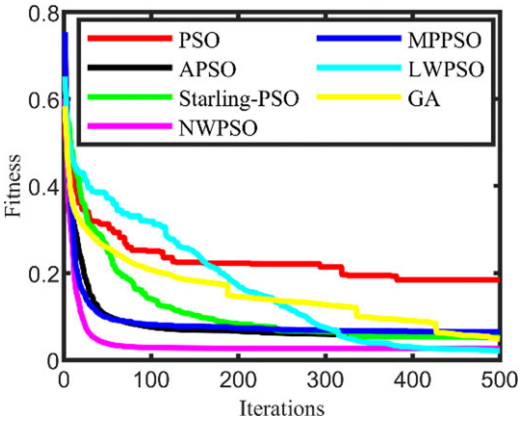

Figure 9. *Average fitness function curves of the algorithm on the redundant robotic arms.*

convergence accuracy of the NWPSO algorithm has not decreased, and its fitness function value is close to that of the APSO algorithm and the LWPSO algorithm, remaining around 0.19. At this point, the fitness function value includes attitude error and joint angle change, and the smaller the value, the better the assembly accuracy of the robot.

5.4. Algorithm performance analysis of redundant robotic arm

To further verify the generality of the proposed algorithm, we used a seven-degrees-of-freedom redundant robotic arm for model validation [19]. The algorithm parameters remained consistent with experiment 5.1, changing the number of iterations to 500. We used f_1 as a function of fitness, and the experimental results are shown in Table V and Fig. 9.

From Table V, it can be seen that the NWPSO algorithm has the smallest fitness value and achieves a convergence accuracy of m. At this point, the posture of the robot is closer to the target posture, and the error is minimized. This indicates that the algorithm has a good inverse kinematics optimization effect on redundant manipulators and verifies the universality of the proposed algorithm. However, the average and variance values of the NWPSO algorithm are only slightly larger than those of the LWPSO algorithm, with a difference of about 0.004. This indicates that the optimization effect and data fluctuation of the algorithm are still relatively small. Combined with Fig. 9, it can be seen that the NWPSO algorithm converges around 50 iterations, while the LWPSO algorithm converges around 400 iterations, indicating a slower convergence speed. The other algorithms do not perform as well as the NWPSO algorithm in terms of convergence speed and accuracy.

In order to verify the inverse solution performance of redundant manipulator during assembly, the algorithm parameters are consistent with experiment 5.3, the number of iterations is set to 500, the population is set to 150, and the f function is used as the objective function. The experimental results

https://doi.org/10.1017/S0263574723001789 Published online by Cambridge University Press

Table VI. *Optimization effect of arithmetic of F function for redundant manipulators.*

Algorithm	Best	Worst	Mean	Var
PSO	7.6820×10^{-2}	4.5224×10^{-2}	3.0021×10^{-1}	6.0923×10^{-2}
APSO	4.5411×10^{-2}	9.2269×10^{-2}	6.5785×10^{-2}	7.7280×10^{-4}
Starling-PSO	4.5411×10^{-2}	9.2269×10^{-2}	6.5785×10^{-2}	7.7280×10^{-4}
NWPSO	4.5411×10^{-2}	9.2269×10^{-2}	6.5785×10^{-2}	7.7280×10^{-4}
MPPSO	4.5542×10^{-2}	9.3613×10^{-2}	6.6164×10^{-2}	7.8765×10^{-4}
LWPSO	4.5411×10^{-2}	9.2269×10^{-2}	6.5785×10^{-2}	7.7280×10^{-4}
GA	4.5416×10^{-2}	9.2273×10^{-2}	6.5790×10^{-2}	7.7280×10^{-4}

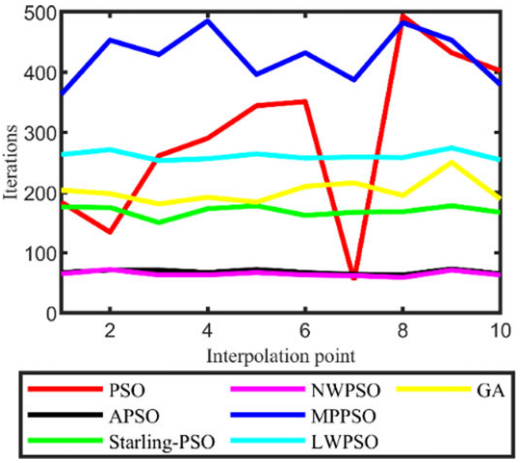

Figure 10. *Comparison of iteration numbers graph.*

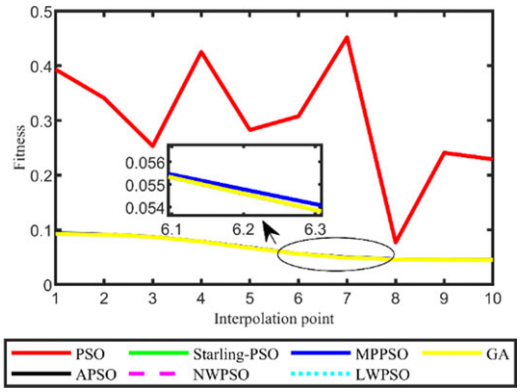

Figure 11. *Fitness function.*

are shown in Table VI and Figs. 10 and 11. Compared with other algorithms, NWPSO algorithm has better convergence speed and accuracy.

Finally, to further compare the performance of the algorithm, the optimization time of each algorithm is compared.

In Fig. 12, it is assumed that the running time of the PSO algorithm in the first experiment is denoted as T. In the first, second and fourth experiments, the running time of the APSO algorithm

https://doi.org/10.1017/S0263574723001789 Published online by Cambridge University Press

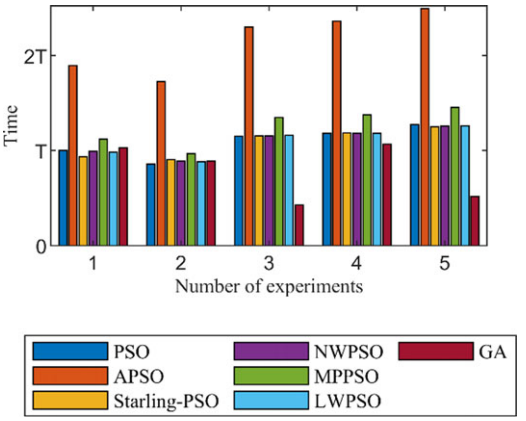

Figure 12. *Runtime comparison of the algorithm.*

is approximately 2T, while other algorithms have a running time of about T. In the third and fifth experiments, the total time is divided by the number of interpolation points, which is regarded as the optimization time of a point. The running time of the APSO algorithm exceeds that of the PSO algorithm, whereas the GA algorithm's running time is shorter than that of the PSO algorithm. The running time of other algorithms is comparable to that of the PSO algorithm. However, the accuracy of the inverse solution obtained by the GA algorithm is inferior to that of the NWPSO algorithm. Considering the optimization performance of the aforementioned algorithms, the NWPSO algorithm proposed in this paper outperforms other algorithms and is suitable for solving robot inverse problems. Importantly, it maintains convergence accuracy without increasing optimization time. Additionally, the algorithm is applicable for the inverse solution optimization of redundant manipulators.

6. Conclusion

In this paper, we transform the complex problem of robot shaft hole assembly into an inverse problem and use the optimization algorithm to find the optimal solution. At the same time, the variation of joint Angle is introduced as the objective function, which avoids the large fluctuation of joint angle during assembly and affects the assembly accuracy. The main work of this paper is as follows:

1. The PSO with nonlinear weight is proposed. The weight-update method of the particle swarm algorithm is improved and applied to many situations for analysis. The simulation results show that compared with other algorithms, NWPSO algorithm has higher convergence accuracy, more stable optimization effect and shorter optimization time.
2. The NWPSO algorithm has been successfully applied to solve the inverse kinematics problem for redundant robotic arms, demonstrating consistent and stable optimization results. The consistency between different robot systems once again proves the universality and robustness of the algorithm, which provides valuable insights for robot assembly process.

Author contributions. Shixiong Zhang proposed the methodology and gave experimental guidance in this work. Ang Li completed the experiment and the draft of this paper. Ruilong Ren assists with supplementary experiments and paper revisions. Jianxin Ren conducted experimental processing on the data set.

Financial support. This research received no specific grant from any funding agency, commercial or not-for-profit sectors.

Competing interests. The authors declare no competing interests exist.

Ethical approval. Not applicable.

https://doi.org/10.1017/S0263574723001789 Published online by Cambridge University Press

References

[1] J. Xu, K. Liu, Y. Pei, C. Yang, Y. Cheng and Z. Liu, "A noncontact control strategy for circular peg-in-hole assembly guided by the 6-DOF robot based on hybrid vision," *IEEE Trans. Instrum. Meas.* **71**, 1–15 (2022).

[2] J. Song, Q. Chen and Z. Li, "A peg-in-hole robot assembly system based on Gauss mixture model," *Robot. Comput. Integr. Manuf.* **67**, 101996 (2021).

[3] S. Starke, N. Hendrich, S. Magg and J. Zhang, "An Efficient Hybridization of Genetic Algorithms and Particle Swarm Optimization for Inverse Kinematics," **In:** *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (IEEE, 2016) pp. 1782–1789.

[4] N. Rokbani and A. M. Alimi, "Inverse kinematics using particle swarm optimization, a statistical analysis," *Proc. Eng.* **64**, 1602–1611 (2013).

[5] R. Gao, "Inverse kinematics solution of Robotics based on neural network algorithms," *J. Ambient Intell. Humaniz. Comput.* **11**(12), 6199–6209 (2020).

[6] N. Netjinda, T. Achalakul and B. Sirinaovakul, "Particle Swarm Optimization inspired by starling flock behavior," *Appl. Soft Comput.* **35**, 411–422 (2015).

[7] F. Nagata, S. Kishimoto, S. Kurita, A. Otsuka and K. Watanabe, "Neural Network-Based Inverse Kinematics for an Industrial Robot and Its Learning Method," **In:** *The Proceedings of the 4th International Conference on Industrial Application Engineering* (2016).

[8] L. B. Zhang, Y. P. Li, D. M. Zhu and Y. L. Fu, "Inverse kinematic solution of nursing robot based on genetic algorithm," *J. Beijing Univ. Aeronaut. Astronaut.* **48**(10), 1925–1932 (2022) (in Chinese).

[9] S. Xiao-long, W. Ji-fang, G. Zi-sheng and M. Fei, "Inverse kinematics solution of manipulator based on improved differential evolution algorithm," *Modular Mach. Tool Autom. Manuf. Tech.* **2022**(4), 1–6 (2022).

[10] J. Liang, H. Guo, K. Chen, K. Yu, C. Yue and X. Li, "An improved Kalman particle swarm optimization for modeling and optimizing of boiler combustion characteristics," *Robotica* **41**(4), 1087–1097 (2023). doi: 10.1017/S026357472200145X.

[11] C. Mei, H. Songyuan and H. Huimin, "Welding robot inverse kinematics solution based on improved chicken swarm algorithm," *Mach. Des. Res.* **38**(02), 88–92+104 (2022).

[12] P. Liang, W. Li and Y. Huang, "Multi-population Cooperative Particle Swarm Optimization with Covariance Guidance," **In:** *2022 4th International Conference on Data-driven Optimization of Complex Systems (DOCS)* (IEEE, 2022) pp. 1–6.

[13] H. Deng and C. Xie, "An improved particle swarm optimization algorithm for inverse kinematics solution of multi-DOF serial robotic manipulators," *Soft Comput.* **25**(21), 13695–13708 (2021).

[14] F. Liu, H. Huang, B. Li and F. Xi, "A parallel learning particle swarm optimizer for inverse kinematics of robotic manipulator," *Int. J. Intell. Syst.* **36**(10), 6101–6132 (2021).

[15] G. Zhao, D. Jiang, X. Liu, X. Tong, Y. Sun, B. Tao, J. Kong, J. Yun, Y. Liu and Z. Fang, "A tandem robotic arm inverse kinematic solution based on an improved particle swarm algorithm," *Front. Bioeng. Biotechnol.* **10**, 832829 (2022).

[16] L. Yiyang, J. Xi, B. Hongfei, W. Zhining and S. Liangliang, "A general robot inverse kinematics solution method based on improved PSO algorithm," *IEEE Access* **9**, 32341–32350 (2021).

[17] Y. Liu, F. Xiao, X. Tong, B. Tao, M. Xu, G. Jiang, B. Chen, Y. Cao and N. Sun, "Manipulator trajectory planning based on work subspace division," *Concurr. Comput. Pract. Exp.* **34**(5), e6710 (2022).

[18] D. Wang, D. Tan and L. Liu, "Particle swarm optimization algorithm: An overview," *Soft Comput.* **22**(2), 387–408 (2018).

[19] M. Alebooyeh and R. J. Urbanic, "Neural network model for identifying workspace, forward and inverse kinematics of the 7-DOF YuMi 14000 ABB collaborative robot," *IFAC-PapersOnLine* **52**(10), 176–181 (2019).

Cite this article: S. Zhang, A. Li, J. Ren and R. Ren (2024). "Kinematics inverse solution of assembly robot based on improved particle swarm optimization", Robotica **42**, 833–845. https://doi.org/10.1017/S0263574723001789