Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 6315675, 13 pages
https://doi.org/10.1155/2020/6315675

Research Article

Hybrid Mutation Fruit Fly Optimization Algorithm for Solving the Inverse Kinematics of a Redundant Robot Manipulator

Jianping Shi,[1,2] **Yuting Mao**,[2] **Peishen Li**,[2] **Guoping Liu**,[2] **Peng Liu**,[1,3] **Xianyong Yang**,[2] **and Dahai Wang**[2]

[1]*School of Electronic & Communication Engineering, Guiyang University, Guiyang 550005, China*
[2]*School of Mechanical & Electrical Engineering, Nanchang University, Nanchang 330031, China*
[3]*School of Gems and Materials Technology, Hebei GEO University, Shijiazhuang 050031, China*

Correspondence should be addressed to Peng Liu; llp080@126.com

Received 7 October 2019; Revised 31 January 2020; Accepted 4 April 2020; Published 7 May 2020

Academic Editor: Jixiang Yang

The inverse kinematics of redundant manipulators is one of the most important and complicated problems in robotics. Simultaneously, it is also the basis for motion control, trajectory planning, and dynamics analysis of redundant manipulators. Taking the minimum pose error of the end-effector as the optimization objective, a fitness function was constructed. Thus, the inverse kinematics problem of the redundant manipulator can be transformed into an equivalent optimization problem, and it can be solved using a swarm intelligence optimization algorithm. Therefore, an improved fruit fly optimization algorithm, namely, the hybrid mutation fruit fly optimization algorithm (HMFOA), was presented in this work for solving the inverse kinematics of a redundant robot manipulator. An olfactory search based on multiple mutation strategies and a visual search based on the dynamic real-time updates were adopted in HMFOA. The former has a good balance between exploration and exploitation, which can effectively solve the premature convergence problem of the fruit fly optimization algorithm (FOA). The latter makes full use of the successful search experience of each fruit fly and can improve the convergence speed of the algorithm. The feasibility and effectiveness of HMFOA were verified by using 8 benchmark functions. Finally, the HMFOA was tested on a 7-degree-of-freedom (7-DOF) manipulator. Then the results were compared with other algorithms such as FOA, LGMS-FOA, AE-LGMS-FOA, IFOA, and SFOA. The pose error of end-effector corresponding to the optimal inverse solution of HMFOA is 10^{-14} mm, while the pose errors obtained by FOA, LGMS-FOA, AE-LGMS-FOA, IFOA, and SFOA are 10^{2} mm, 10^{-1} mm, 10^{-2} mm, 10^{2} mm, and 10^{2} mm, respectively. The experimental results show that HMFOA can be used to solve the inverse kinematics problem of redundant manipulators effectively.

1. Introduction

Forward kinematics and inverse kinematics are two basic problems in robot kinematics. The forward kinematics which determines the pose of the end-effector relative to the reference coordinate system according to the joint variables of the robot is relatively easy, and its solution is analytical, deterministic, and unique, while the inverse kinematics, to obtain the joint variables from the pose of the end-effector, is a complex system of nonlinear equations with the strong coupling of variables. As a result, it is a much more difficult problem than the forward kinematics. The inverse kinematics problem plays an important role in robotics, which is the premise and foundation of robot motion control, trajectory planning, and dynamic analysis [1].

Generally, the conventional methods for solving the inverse kinematics problem of robots are closed-form methods and numerical methods [2]. The closed-form methods, which consist of algebraic [3] and geometric [4], have the advantages of fast solution speed, high accuracy, and accessibility in obtaining all possible inverse solutions. However, the closed-form methods highly depend on the configuration of the robot. If a robot does not meet the Pieper criterion [5], the closed-form methods cannot be applied to solve the inverse kinematics problem. Therefore, only a very small class of kinematically simple robots is

suitable for the closed-form methods to obtain complete analytical solutions. The numerical methods are the main methods to resolve the inverse kinematics of complex articulated manipulators. Unfortunately, the numerical methods may have the singularity of Jacobian matrix and may also cause considerable computational load. In addition, the redundant manipulator has an infinite number of inverse kinematics solutions to reach the same end-effector pose, which makes it more difficult to solve inverse kinematics problems by using the conventional methods. Therefore, more and more researchers have focused on solving the inverse kinematics problem using artificial intelligent optimization algorithms.

So far, various optimization algorithms such as genetic algorithm (GA) [6, 7], particle swarm optimization algorithm (PSO) [8–12], differential evolution algorithm (DE) [2, 13, 14], artificial bee colony algorithm (ABC) [15], and biogeography-based optimization [16] have been proposed to calculate the inverse kinematics solutions of robots; thus the shortcomings of the conventional methods are effectively overcome. In this study, a fitness function was constructed to minimize the pose error of end-effector. Then, the inverse kinematics problem was transformed into an optimization problem using the forward kinematics and the fitness function, and a novel hybrid mutation fruit fly optimization algorithm (HMFOA) was developed to solve the problem more effectively.

The rest of this paper is organized as follows. Section 2 presents a brief introduction to the forward and inverse kinematics of a 7-DOF redundant robot manipulator. The HMFOA is described in Section 3. Experimental design and comparisons are illustrated in Section 4. Finally, the results of the study are summarized in Section 5.

2. Kinematic Analysis of a 7-DOF Robot Manipulator

In this study, the 7-DOF YuMi 14000 ABB industrial robot [17] was used as an example to discuss the inverse kinematics problem. The schematic of the 7-DOF robot is shown in Figure 1, and its links and joints are given in Figure 2. The structure diagram of the left arm of the robot is presented in Figure 3. The Denavit–Hartenberg (DH) parameters as well as lower and upper joint limits for the left arm of the robot are listed in Table 1. In the work, the left arm of the robot was only used as the object to be investigated.

In Table 1, the parameters θ_i, α_i, d_i, a_i, l_i, and u_i ($i = 1, 2, 3, \ldots, 7$) represent the joint angle, link twist angle, link offset, link length, lower joint limit, and upper joint limit, respectively. The homogeneous transformation matrix ${}_i^{i-1}\mathbf{T}$ of link $i$is formed by DH parameters in

$$ {}_i^{i-1}\mathbf{T} = \begin{bmatrix} c\theta_i & -c\alpha_i s\theta_i & s\alpha_i s\theta_i & a_i c\theta_i \\ s\theta_i & c\alpha_i c\theta_i & -s\alpha_i c\theta_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1} $$

where $s\theta_i$ and $c\theta_i$ denote $\sin(\theta_i)$ and $\cos(\theta_i)$, respectively, while $s\alpha_i$ and $c\alpha_i$ denote $\sin(\alpha_i)$ and $\cos(\alpha_i)$, respectively.

Thus, the forward kinematic can be calculated as

$$ {}_7^0\mathbf{T} = {}_1^0\mathbf{T} \cdot {}_2^1\mathbf{T} \cdot {}_3^2\mathbf{T} \cdot {}_4^3\mathbf{T} \cdot {}_5^4\mathbf{T} \cdot {}_6^5\mathbf{T} \cdot {}_7^6\mathbf{T} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2} $$

where ${}_7^0\mathbf{T}$ is the pose matrix of the end-effector relative to the base coordinate system. n_x, n_y, n_z, o_x, o_y, o_z, a_x, a_y, and a_z represent the rotational elements of the pose matrix, while p_x, p_y, and p_z are the elements of position vector.

According to equations (1) and (2), the pose equations are as follows.

$$ \begin{aligned} n_x = {} & c\theta_7\left(s\theta_6\left(c\theta_4\left(s\theta_1 s\theta_3 - c\theta_1 c\theta_2 c\theta_3\right) + c\theta_1 s\theta_2 s\theta_4\right) + c\theta_6\left(c\theta_5\left(s\theta_4\left(s\theta_1 s\theta_3\right.\right.\right.\right. \\ & \left.\left.\left.\left. - c\theta_1 c\theta_2 c\theta_3\right) - c\theta_1 c\theta_4 s\theta_2\right) - s\theta_5\left(c\theta_3 s\theta_1 + c\theta_1 c\theta_2 s\theta_3\right)\right)\right) - s\theta_7\left(s\theta_5\left(s\theta_4\left(s\theta_1 s\theta_3\right.\right.\right. \\ & \left.\left.\left. - c\theta_1 c\theta_2 c\theta_3\right) - c\theta_1 c\theta_4 s\theta_2\right) + c\theta_5\left(c\theta_3 s\theta_1 + c\theta_1 c\theta_2 s\theta_3\right)\right), \end{aligned} \tag{3} $$

$$ \begin{aligned} n_y = {} & s\theta_7\left(s\theta_5\left(s\theta_4\left(c\theta_1 s\theta_3 + c\theta_2 c\theta_3 s\theta_1\right) + c\theta_4 s\theta_1 s\theta_2\right) + c\theta_5\left(c\theta_1 c\theta_3 - c\theta_2 s\theta_1 s\theta_3\right)\right) \\ & - c\theta_7\left(s\theta_6\left(c\theta_4\left(c\theta_1 s\theta_3 + c\theta_2 c\theta_3 s\theta_1\right) - s\theta_1 s\theta_2 s\theta_4\right) + c\theta_6\left(c\theta_5\left(s\theta_4\left(c\theta_1 s\theta_3\right.\right.\right.\right. \\ & \left.\left.\left.\left. + c\theta_2 c\theta_3 s\theta_1\right) + c\theta_4 s\theta_1 s\theta_2\right) - s\theta_5\left(c\theta_1 c\theta_3 - c\theta_2 s\theta_1 s\theta_3\right)\right)\right), \end{aligned} \tag{4} $$

$$ \begin{aligned} n_z = {} & s\theta_7\left(s\theta_5\left(c\theta_2 c\theta_4 - c\theta_3 s\theta_2 s\theta_4\right) + c\theta_5 s\theta_2 s\theta_3\right) - c\theta_7\left(c\theta_6\left(c\theta_5\left(c\theta_2 c\theta_4\right.\right.\right. \\ & \left.\left.\left. - c\theta_3 s\theta_2 s\theta_4\right) - s\theta_2 s\theta_3 s\theta_5\right) - s\theta_6\left(c\theta_2 s\theta_4 + c\theta_3 c\theta_4 s\theta_2\right)\right), \end{aligned} \tag{5} $$

$$ \begin{aligned} o_x = {} & -c\theta_7\left(s\theta_5\left(s\theta_4\left(s\theta_1 s\theta_3 - c\theta_1 c\theta_2 c\theta_3\right) - c\theta_1 c\theta_4 s\theta_2\right) + c\theta_5\left(c\theta_3 s\theta_1 + c\theta_1 c\theta_2 s\theta_3\right)\right) \\ & - s\theta_7\left(s\theta_6\left(c\theta_4\left(s\theta_1 s\theta_3 - c\theta_1 c\theta_2 c\theta_3\right) + c\theta_1 s\theta_2 s\theta_4\right) + c\theta_6\left(c\theta_5\left(s\theta_4\left(s\theta_1 s\theta_3\right.\right.\right.\right. \\ & \left.\left.\left.\left. - c\theta_1 c\theta_2 c\theta_3\right) - c\theta_1 c\theta_4 s\theta_2\right) - s\theta_5\left(c\theta_3 s\theta_1 + c\theta_1 c\theta_2 s\theta_3\right)\right)\right), \end{aligned} \tag{6} $$

$$o_y = c\theta_7\left(s\theta_5\left(s\theta_4\left(c\theta_1 s\theta_3 + c\theta_2 c\theta_3 s\theta_1\right) + c\theta_4 s\theta_1 s\theta_2\right) + c\theta_5\left(c\theta_1 c\theta_3 - c\theta_2 s\theta_1 s\theta_3\right)\right) + s\theta_7\left(s\theta_6\left(c\theta_4\left(c\theta_1 s\theta_3 + c\theta_2 c\theta_3 s\theta_1\right) - s\theta_1 s\theta_2 s\theta_4\right) + c\theta_6\left(c\theta_5\left(s\theta_4\left(c\theta_1 s\theta_3 + c\theta_2 c\theta_3 s\theta_1\right) + c\theta_4 s\theta_1 s\theta_2\right) - s\theta_5\left(c\theta_1 c\theta_3 - c\theta_2 s\theta_1 s\theta_3\right)\right)\right), \tag{7}$$

$$o_z = c\theta_6\left(c\theta_4\left(c\theta_1 s\theta_3 + c\theta_2 c\theta_3 s\theta_1\right) - s\theta_1 s\theta_2 s\theta_4\right) - s\theta_6\left(c\theta_5\left(s\theta_4\left(c\theta_1 s\theta_3 + c\theta_2 c\theta_3 s\theta_1\right) + c\theta_4 s\theta_1 s\theta_2\right) - s\theta_5\left(c\theta_1 c\theta_3 - c\theta_2 s\theta_1 s\theta_3\right)\right), \tag{8}$$

$$a_x = s\theta_6\left(c\theta_5\left(s\theta_4\left(s\theta_1 s\theta_3 - c\theta_1 c\theta_2 c\theta_3\right) - c\theta_1 c\theta_4 s\theta_2\right) - s\theta_5\left(c\theta_3 s\theta_1 + c\theta_1 c\theta_2 s\theta_3\right)\right) - c\theta_6\left(c\theta_4\left(s\theta_1 s\theta_3 - c\theta_1 c\theta_2 c\theta_3\right) + c\theta_1 s\theta_2 s\theta_4\right), \tag{9}$$

$$a_y = c\theta_6\left(c\theta_4\left(c\theta_1 s\theta_3 + c\theta_2 c\theta_3 s\theta_1\right) - s\theta_1 s\theta_2 s\theta_4\right) - s\theta_6\left(c\theta_5\left(s\theta_4\left(c\theta_1 s\theta_3 + c\theta_2 c\theta_3 s\theta_1\right) + c\theta_4 s\theta_1 s\theta_2\right) - s\theta_5\left(c\theta_1 c\theta_3 - c\theta_2 s\theta_1 s\theta_3\right)\right), \tag{10}$$

$$a_z = -s\theta_6\left(c\theta_5\left(c\theta_2 c\theta_4 - c\theta_3 s\theta_2 s\theta_4\right) - s\theta_2 s\theta_3 s\theta_5\right) - c\theta_6\left(c\theta_2 s\theta_4 + c\theta_3 c\theta_4 s\theta_2\right), \tag{11}$$

$$p_x = a_1 c\theta_1 - d_7\left(c\theta_6\left(c\theta_4\left(s\theta_1 s\theta_3 - c\theta_1 c\theta_2 c\theta_3\right) + c\theta_1 s\theta_2 s\theta_4\right) - s\theta_6\left(c\theta_5\left(s\theta_4\left(s\theta_1 s\theta_3 - c\theta_1 c\theta_2 c\theta_3\right) - c\theta_1 c\theta_4 s\theta_2\right) - s\theta_5\left(c\theta_3 s\theta_1 + c\theta_1 c\theta_2 s\theta_3\right)\right)\right) - d_5\left(c\theta_4\left(s\theta_1 s\theta_3 - c\theta_1 c\theta_2 c\theta_3\right) + c\theta_1 s\theta_2 s\theta_4\right) - a_4 s\theta_4\left(s\theta_1 s\theta_3 - c\theta_1 c\theta_2 c\theta_3\right) - a_5 s\theta_5\left(c\theta_3 s\theta_1 + c\theta_1 c\theta_2 s\theta_3\right) + a_2 c\theta_1 c\theta_2 + d_3 c\theta_1 s\theta_2 - a_3 s\theta_1 s\theta_3 + a_5 c\theta_5\left(s\theta_4\left(s\theta_1 s\theta_3 - c\theta_1 c\theta_2 c\theta_3\right) - c\theta_1 c\theta_4 s\theta_2\right) + a_6 s\theta_6\left(c\theta_4\left(s\theta_1 s\theta_3 - c\theta_1 c\theta_2 c\theta_3\right) + c\theta_1 s\theta_2 s\theta_4\right) + a_6 c\theta_6\left(c\theta_5\left(s\theta_4\left(s\theta_1 s\theta_3 - c\theta_1 c\theta_2 c\theta_3\right) - c\theta_1 c\theta_4 s\theta_2\right) - s\theta_5\left(c\theta_3 s\theta_1 + c\theta_1 c\theta_2 s\theta_3\right)\right) + a_3 c\theta_1 c\theta_2 c\theta_3 + a_4 c\theta_1 c\theta_4 s\theta_2, \tag{12}$$

$$p_y = d_7\left(c\theta_6\left(c\theta_4\left(c\theta_1 s\theta_3 + c\theta_2 c\theta_3 s\theta_1\right) - s\theta_1 s\theta_2 s\theta_4\right) - s\theta_6\left(c\theta_5\left(s\theta_4\left(c\theta_1 s\theta_3 + c\theta_2 c\theta_3 s\theta_1\right) + c\theta_4 s\theta_1 s\theta_2\right) - s\theta_5\left(c\theta_1 c\theta_3 - c\theta_2 s\theta_1 s\theta_3\right)\right)\right) + a_1 s\theta_1 + d_5\left(c\theta_4\left(c\theta_1 s\theta_3 + c\theta_2 c\theta_3 s\theta_1\right) - s\theta_1 s\theta_2 s\theta_4\right) - a_6 c\theta_6\left(c\theta_5\left(s\theta_4\left(c\theta_1 s\theta_3 + c\theta_2 c\theta_3 s\theta_1\right) + c\theta_4 s\theta_1 s\theta_2\right) - s\theta_5\left(c\theta_1 c\theta_3 - c\theta_2 s\theta_1 s\theta_3\right)\right) + a_4 s\theta_4\left(c\theta_1 s\theta_3 + c\theta_2 c\theta_3 s\theta_1\right) + a_5 s\theta_5\left(c\theta_1 c\theta_3 - c\theta_2 s\theta_1 s\theta_3\right) + a_2 c\theta_2 s\theta_1 + a_3 c\theta_1 s\theta_3 + d_3 s\theta_1 s\theta_2 - a_5 c\theta_5\left(s\theta_4\left(c\theta_1 s\theta_3 + c\theta_2 c\theta_3 s\theta_1\right) + c\theta_4 s\theta_1 s\theta_2\right) - a_6 s\theta_6\left(c\theta_4\left(c\theta_1 s\theta_3 + c\theta_2 c\theta_3 s\theta_1\right) - s\theta_1 s\theta_2 s\theta_4\right) + a_3 c\theta_2 c\theta_3 s\theta_1 + a_4 c\theta_4 s\theta_1 s\theta_2, \tag{13}$$

$$p_z = d_1 - d_5\left(c\theta_2 s\theta_4 + c\theta_3 c\theta_4 s\theta_2\right) - d_7\left(s\theta_6\left(c\theta_5\left(c\theta_2 c\theta_4 - c\theta_3 s\theta_2 s\theta_4\right) - s\theta_2 s\theta_3 s\theta_5\right) + c\theta_6\left(c\theta_2 s\theta_4 + c\theta_3 c\theta_4 s\theta_2\right) + d_3 c\theta_2 - a_2 s\theta_2 - a_5 c\theta_5\left(c\theta_2 c\theta_4 - c\theta_3 s\theta_2 s\theta_4\right) + a_6 s\theta_6\left(c\theta_2 s\theta_4 + c\theta_3 c\theta_4 s\theta_2\right) + a_4 c\theta_2 c\theta_4 - a_3 c\theta_3 s\theta_2 - a_6 c\theta_6\left(c\theta_5\left(c\theta_2 c\theta_4 - c\theta_3 s\theta_2 s\theta_4\right) - s\theta_2 s\theta_3 s\theta_5\right) - a_4 c\theta_3 s\theta_2 s\theta_4 + a_5 s\theta_2 s\theta_3 s\theta_5. \tag{14}$$

It can be seen that the pose equations of the redundant manipulator are the functions of joint variables. Given the joint variables, the pose of the end-effector relative to the reference coordinate system can be calculated through the pose equations, and the result is unique; that is, the forward kinematics is a one-to-one relationship. However, given the pose of the end-effector, the corresponding joint variables will not be unique; that is, the inverse kinematics is a one-to-many relationship, and there are infinite inverse kinematics solutions for the 7-DOF robot manipulator.

3. Improved FOA

Inspired by the foraging behaviour of fruit flies, Pan [18] proposed a novel meta-heuristic algorithm, namely, the fruit fly optimization algorithm (FOA). Compared with other swarm intelligence-based algorithms, FOA has the merits of simple algorithm framework, few adjustable parameters, easy understanding, and implementation. Therefore, FOA has been used to resolve many science and engineering problems [19–22]. However, FOA also has lots of shortcomings. Particularly, because the smell concentration judgment value cannot be taken a negative value, FOA cannot deal with the optimization problems with negative decision variables. In addition, the optimization performance of FOA is highly dependent on the optimal solution of the current generation. As a result, the ability to maintain population diversity is not strong, making the algorithm easy to fall into local extremum.

To improve the convergence performance of FOA, a novel hybrid mutation mechanism for osphresis searching and a real-time dynamic update mechanism for vision

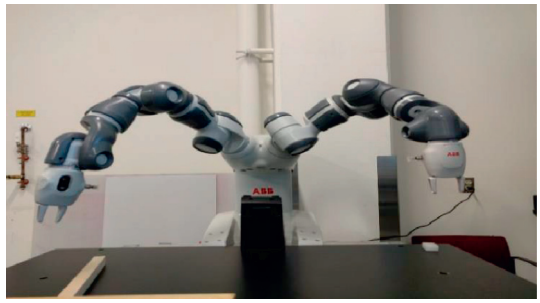

Figure 1: 7-DOF YuMi 14000 ABB industrial robot.

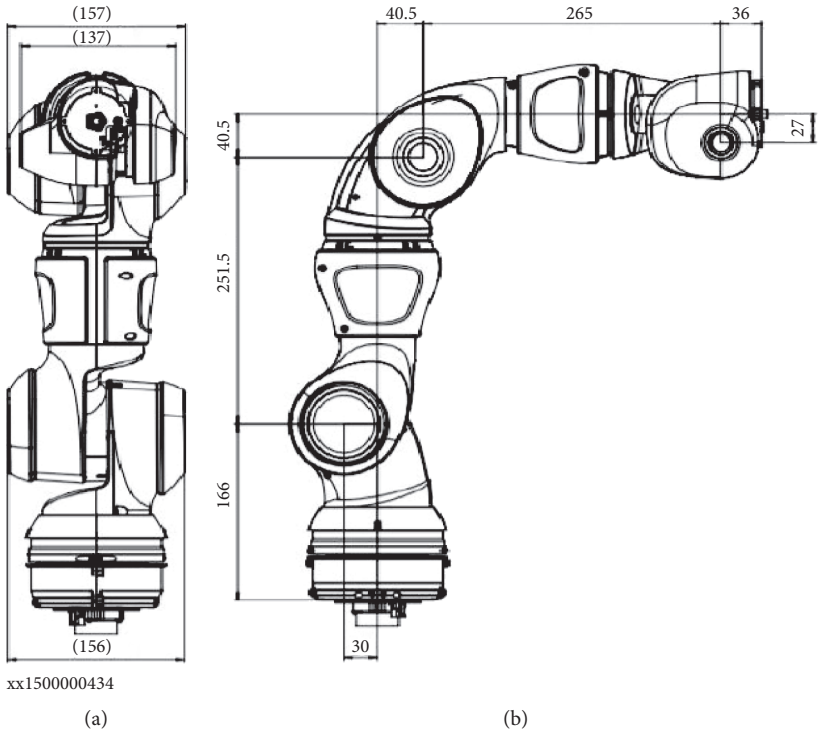

Figure 2: YuMi robot arm links and joints (robotics product specification IRB 14000, 2015).

searching are introduced, and a new improved FOA called HMFOA is proposed in this section.

3.1. A Hybrid Mutation Mechanism for Osphresis Search. In the original FOA, the smell concentration judgment value S_i corresponds to the decision vector of the optimization problem, and its value is the reciprocal of distance. As a result, the smell concentration judgment value cannot be assigned a negative value and cannot be uniformly distributed in the search space. In HMFOA, the position of the ith fly in D-dimensional search space is presented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, and the smell concentration judgment value of the ith fly can be directly obtained according to $S_i = X_i$. In this way, the above shortcomings can be overcome.

Inspired by the mutation idea of the differential evolution algorithm, the osphresis searching of HMFOA is executed according to equations (15)–(18).

$$X_i = \begin{cases} X_i', & \text{rand}_1 < 1 - \omega, \\ X_i'', & \text{rand}_1 \geq 1 - \omega, \end{cases} \tag{15}$$

$$X_i' = \begin{cases} \text{Pb}_g + \omega \times (\text{Pb}_{r1} - \text{Pb}_{r2}), & \text{rand}_2 < 0.5, \\ \text{Pb}_g \times \text{rand}_3 + \omega \times (\text{Pb}_{r1} - \text{Pb}_{r2}), & \text{rand}_2 \geq 0.5, \end{cases} \tag{16}$$

$$X_i'' = \begin{cases} \text{Pb}_i + \omega \times (\text{Pb}_{r1} - \text{Pb}_{r2}), & \text{rand}_4 < 0.5, \\ L + \text{rand}_5 \times (U - L), & \text{rand}_4 \geq 0.5, \end{cases} \tag{17}$$

$$\omega = \frac{1 + \cos(\pi \times (t-1)/t_{\max})}{2}, \tag{18}$$

where Pb_g denotes the best previous position of all fruit flies in the swarm, Pb_{r1} denotes the best previous position of the $r1$ th fruit fly, Pb_{r2} denotes the best previous position of the $r2$ th fruit fly, $r1$ and $r2$ are random and mutually different

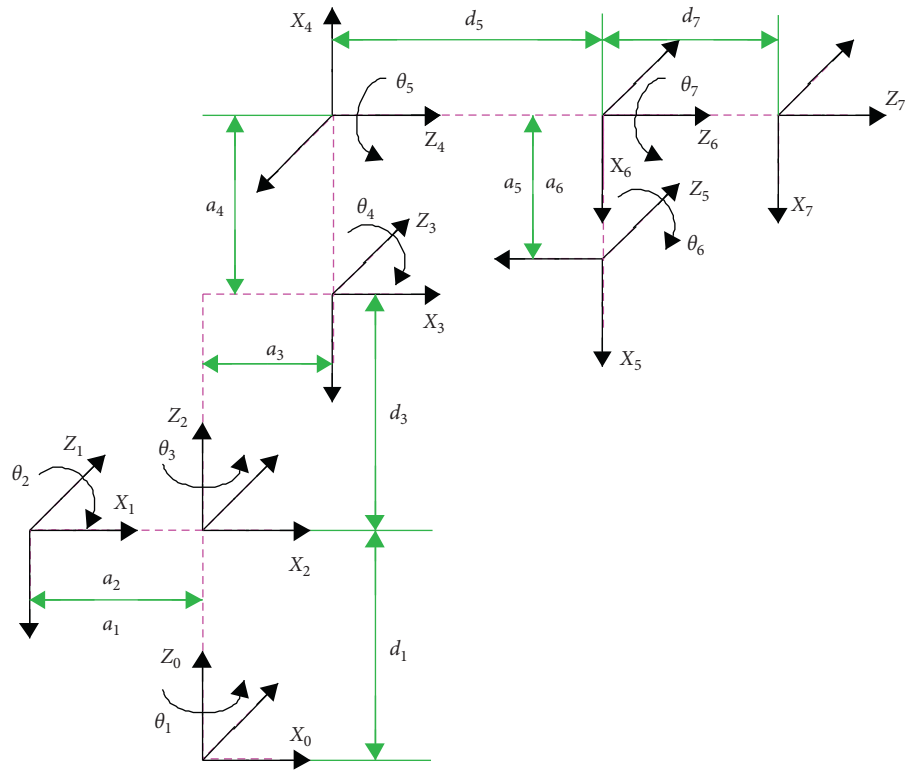

Figure 3: The structure diagram of the left arm of the YuMi robot.

Table 1: DH parameters for the left arm (i is the axis).

i	d_i (mm)	θ_i (degree)	a_i (mm)	α_i (degree)	l_i (degree)	u_i (degree)
1	166	0	−30	−90	−168.5	168.5
2	0	0	30	90	−143.5	43.5
3	251.5	0	40.5	−90	−123.5	80
4	0	−90	40.5	−90	−290	290
5	265	180	27	−90	−88	138
6	0	0	−27	90	−229	229
7	36	0	0	0	−168.5	168.5

integers generated in the range $[1, m]$ (m presents the population size), and they also are different from the fly's index i and g; that is, $r1 \neq r2 \neq i \neq g$. U and L are the upper and lower boundary vectors of the search space, respectively. rand_1, rand_2, rand_3, rand_4, and rand_6 are random variables uniformly distributed on the interval [0, 1]. ω is a perturbation scaling factor that controls the search radius. t is the current iteration of the algorithm and $t_{\max}$ is the maximum iteration number of the algorithm.

According to equation (15), equation (17) is selected as the searching operator with a larger probability in the early stage of algorithm evolution, which ensures that the algorithm has the strong global search ability. As the number of iterations of the algorithm increases gradually, the probability that equation (16) is selected as the searching operator also increases gradually. Therefore, in the later stage of algorithm evolution, the local perturbation search near the global optimal position is enhanced. Thus, the convergence accuracy of the algorithm is steadily improved.

According to equations (16) and (17), it can be seen that the essence of olfactory searching is to randomly select one of the four mutation strategies according to probability as the current olfactory searching operator of the ith fly. Thus, a hybrid coevolution mechanism can be formed by randomly alternating different mutation strategies, and the diversity of the swarm is well maintained. The difference vector $\omega \times (\text{Pb}_{r1} - \text{Pb}_{r2})$ is used to randomly disturb the reference vector, and it is also helpful to improve the diversity of algorithm search. According to equation (18), a nonlinear decreasing perturbation scaling factor ω is introduced to balance the global exploration and local exploitation of the HMFOA. In the early stage of iteration, the value of factor ω is relatively large, which ensures that the algorithm can carry out optimal search within a larger search radius and thus has a strong global search capability. As the value of factor ω decreases, the search radius of fruit fly individual also decreases. Hence, the local search capability of the algorithm is enhanced in the later stage of algorithm evolution, and the candidate solution with better quality can be found in local neighbourhood.

3.2. A Real-Time Dynamic Update Mechanism for Vision Search. In the original FOA, when all fruit flies have completed their olfactory search, they fly toward the current optimal location using their visual senses. This search method, which only learns from the best individual, has the following disadvantages. Firstly, if the position corresponding to the best fruit fly individual is a local extreme value, it will easily lead to premature convergence of the algorithm. Secondly,

because the contribution of nonoptimal fruit flies to population search is ignored, this search method places too much emphasis on competition among fruit flies and weakens cooperation among individuals.

In order to accelerate the convergence speed of the algorithm, a dynamic real-time update mechanism is employed in the visual search. In HMFOA, the vision searching refers to the dynamic real-time update of the search center position vector in equations (16) and (17). That is, when the position of the fly is updated, the new position of the fly is evaluated immediately, and then the disturbance center position vectors Pb_g, $\mathrm{Pb}_g \times \mathrm{rand}_3$, and Pb_i are further updated in real-time using equations (19) and (20) (in this paper, the optimization problem refers to the minimum optimization problem).

$$\mathrm{Pb}_g = \begin{cases} X_i, & \mathrm{fitness}(X_i) \le \mathrm{fitness}(\mathrm{Pb}_g), \\ \mathrm{Pb}_g, & \mathrm{fitness}(X_i) > \mathrm{fitness}(\mathrm{Pb}_g), \end{cases} \tag{19}$$

$$\mathrm{Pb}_i = \begin{cases} X_i, & \mathrm{fitness}(X_i) \le \mathrm{fitness}(\mathrm{Pb}_i), \\ \mathrm{Pb}_i, & \mathrm{fitness}(X_i) > \mathrm{fitness}(\mathrm{Pb}_i), \end{cases} \tag{20}$$

where $\mathrm{fitness}(\cdot)$ is the fitness function.

In short, HMFOA can make full use of the successful search experience of each fly individual by using a real-time updated vision searching mechanism, which can effectively improve the search efficiency of the swarm.

3.3. Cross-Border Processing. During the evolution of HMFOA, it may occur that the position of the fly is beyond the search space. In order to enhance the diversity of the swarm, the cross-border position component is processed as follows.

$$x_{ij} = \begin{cases} u_j - \min\left(x_i^j - u_j, u_j - l_j\right) * \mathrm{rand}_6, & x_{ij} > u_j, \\ l_j + \min\left(l_j - x_i^j, u_j - l_j\right) * \mathrm{rand}_6, & x_{ij} < l_j, \end{cases} \tag{21}$$

where rand_6 is a random variable uniformly distributed on the interval [0, 1] and l_j and u_j are the lower and upper bounds for x_{ij}, respectively.

3.4. Implementation of HMFOA. The implementation process of HMFOA is shown in Figure 4. The detailed steps of implementing the HMFOA are described as follows:

Step 1: initialize relate parameters, including population size and the maximum number of iterations. Then, randomly initialize the locations of all fruit flies in the search space. The best position of the fly individual is the current position. The global optimal position is selected as the current optimal position of the swarm. Assign $i = 1$, $t = 1$.

Step 2: the olfactory search of the ith fly is performed according to equations (15)–(18). The cross-border position components are processed according to equation (21) and calculate the fitness of the ith fly.

Step 3: according to equation (19), the best previous position of the swarm is updated in real time, and its objective fitness is recorded.

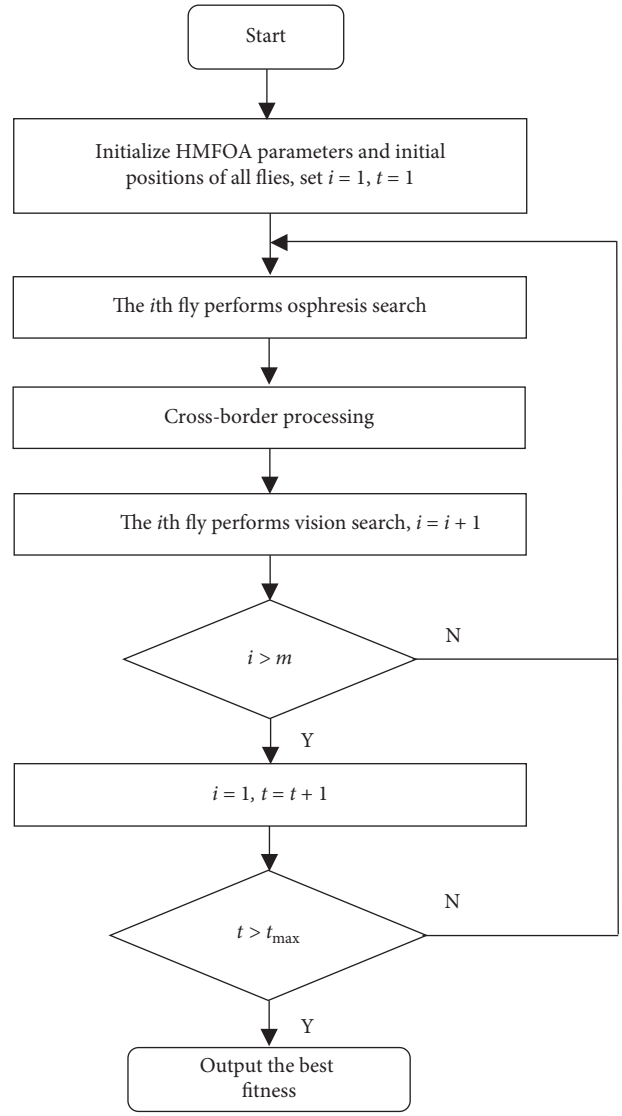

Figure 4: The process of HMFOA.

Step 4: according to equation (20), the best previous position of the ith fly is updated in real time, and its objective fitness is recorded.

Step 5: if all flies of the current generation complete the evolution operation, that is, $i = m$, the next step is executed. Otherwise, set $i = i + 1$ and return to step 2.

Step 6: if the iterative number reaches the maximum, that is, $t = t_{\max}$, the algorithm ends and the optimization results are output. Otherwise, set $t = t + 1$, $i = 1$ and return to step 2.

4. Simulation Results and Discussion

In this section, 8 typical benchmark functions are used to verify the performance of the HMFOA, and then the algorithm is used to solve the inverse kinematics problem of the 7-DOF YuMi 14000 ABB industrial robot. The optimization results of HMFOA are compared with those of FOA, IFOA [23], LGMS-FOA [24], AE-LGMS-FOA [25], and SFOA [26].

4.1. Parameter Setting. For all the algorithms, the population size is 60, and the maximum number of iterations is 1000. In LGMS-FOA, we set $\omega_0 = 1$, $\alpha = 0.95$, $n = 0.005$. The parameters of AE-LGMS-FOA are $p = 0.005$, $\omega_0 = 1$, $n = 10$, and 80% of the best population is used to generate X_{Av}. For FOA, the random initialization fruit fly swarm location zone is [0, 10], and the random direction and distance of iterative fruit fly food searching are [−1, 1]. In IFOA, 50% of the individuals in a swarm fly toward the local optimal solution, and the others fly randomly; the perturbation amplification factor ω is 0.3.

The DH parameters as well as lower and upper joint limits for the left arm of the robot are listed in Table 1, as mentioned above.

All algorithms are coded in MATLAB R2013a. The computation is conducted on a personal computer (PC) with Intel (R) Core (TM) i7-7700, 3.6 GHz CPU, 16 GB RAM, and Windows 10 Operational System.

4.2. Function Simulation. Eight benchmark functions in Table 2 are used to test the performance of HMFOA. Among them, f_1, f_2, f_3, and f_4 are unimodal functions, which are employed to test the convergence speed and accuracy of the algorithm. The latter four functions are multimodal functions, which are made use of testing the global searching ability of the algorithm.

To evaluate the reliability of the results, each function was tested for 50 runs; the fitness results of the best value (Best), the mean value (Mean), the worst value (Worst), and the standard deviation (Std) are reported in Table 3 for all functions with dimensions equal to 30. The average convergence curve of each algorithm is shown in Figure 5.

As can be seen from Table 3, for all benchmark functions, the HMFOA is far superior to other algorithms involved in comparison in terms of the best value, mean value, worst value, and standard deviation. Among the 8 benchmark functions, HMFOA can converge to the theoretical optimal values of 7 functions (except f_7). For function f_7, HMFOA also achieves satisfactory convergence quality. It can be seen from Figure 5, compared with the other five algorithms, the evolution curve of HMFOA drops faster and reaches a lower level (to facilitate evaluation and observation, the fitness of the objective function in the graph is a logarithm with the base of 10), indicating that HMFOA has the advantages of fast convergence speed and high convergence accuracy. The results in Figure 5 also show that FOA, IFOA, LGMS-FOA, AE-LGMS-FOA, and SFOA are easy to fall into local optimum.

4.3. Inverse Kinematics Solution of Redundant Manipulator. In the inverse kinematics calculation of the manipulator, the position vector of the ith fly contains the set of the joint variables of the 7-DOF redundant manipulator, that is, $X_i = [x_{i1}, x_{i2}, ..., x_{i7}] = [\theta_{i1}, \theta_{i2}, ..., \theta_{i7}]$, where $\theta_{i1}, \theta_{i2}, ..., \theta_{i7}$ denote 7 joint angles, respectively. For a desired pose matrix ${}_7^0\mathbf{T}^* = \begin{bmatrix} n_x^* & o_x^* & a_x^* & p_x^* \\ n_y^* & o_y^* & a_y^* & p_y^* \\ n_z^* & o_z^* & a_z^* & p_z^* \\ 0 & 0 & 0 & 1 \end{bmatrix}$ of the end-effector, the position vector X_i of the ith fly is substituted into the forward kinematics equations (3)–(14) to obtain the actual pose matrix ${}_7^0\mathbf{T} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$. If ${}_7^0\mathbf{T}^* = {}_7^0\mathbf{T}$ holds, then X_i is the solution of the corresponding inverse kinematics. Therefore, the optimization objective of the algorithm should be to make the matrix ${}_7^0\mathbf{T}$ infinitely close to the matrix ${}_7^0\mathbf{T}^*$ by changing the position of the fruit fly.

The orientation error and position error of the end-effector are calculated by equations (22) and (23), respectively.

$$f' = \left|n_x^* - n_x\right| + \left|n_y^* - n_y\right| + \left|n_z^* - n_z\right| + \left|o_x^* - o_x\right| + \left|o_y^* - o_y\right| + \left|o_z^* - o_z\right| + \left|a_x^* - a_x\right| + \left|a_y^* - a_y\right| + \left|a_z^* - a_z\right|, \tag{22}$$

$$f'' = \left|p_x^* - p_x\right| + \left|p_y^* - p_y\right| + \left|p_z^* - p_z\right|. \tag{23}$$

Thus, the fitness function of the inverse kinematics problem can be defined as follows.

$$f = \min\left(f' + \lambda f''\right), \tag{24}$$

where $\min(\cdot)$is the minimum function and λ is the weight factor.

In order to solve the inverse kinematics more effectively, we expand the population size of all the algorithms and set it to 100, and the rest of the parameters are the same as mentioned above. The weight factor λ is set as 10^{-2}. In this research, the desired pose matrix of the end-effector is set as follows.

$${}_7^0\mathbf{T}^* = \begin{bmatrix} -0.4294219697 & -0.3634256892 & 0.8267518009 & 16.0540518588 \\ 0.8450970831 & 0.1610933636 & 0.5097645028 & 66.8595013541 \\ -0.3184457443 & 0.9175896123 & 0.2379529604 & 243.1491200419 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{25}$$

In order to investigate the reliability of the results, each algorithm runs independently and continuously for 50 times, respectively. The fitness results of the Best, the Worst, the Mean, the Std, and the successful ratio (SR) are reported in Table 4. The average convergence curve of each algorithm is shown in Figure 6. One of the best inverse kinematics

Table 2: Benchmark functions.

Test function	Search space	Optimal value
$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	0
$f_2(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	$[-30, 30]^D$	0
$f_3(x) = \sum_{i=1}^{D} (\sum_{k=1}^{i} x_k)^2$	$[-100, 100]^D$	0
$f_4(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$	0
$f_5(x) = \sum_{i=1}^{D} x_i^2/4000 - \prod_{i=1}^{D} \cos(x_i/\sqrt{i}) + 1$	$[-600, 600]^D$	0
$f_6(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^D$	0
$f_7(x) = -20\exp(-0.2\sqrt{(1/D)\sum_{i=1}^{D} x_i^2}) - \exp((1/D)\sum_{i=1}^{D} \cos(2\pi x_i)) + 20 + e$	$[-100, 100]^D$	0
$f_8(x) = f(x_1, x_2) + f(x_2, x_3) + \cdots + f(x_D, x_1), f(x, y) = 0.5 + ((\sin^2(\sqrt{x^2 + y^2}) - 0.5)/(1 + 0.001(x^2 + y^2)))^2$	$[-100, 100]^D$	0

Table 3: Comparison results of benchmark functions.

		FOA	LGMS-FOA	AELGMS-FOA	IFOA	SFOA	HMFOA
f_1	Best	1.411770$e-4$	3.466251$e-40$	1.911653$e-19$	1.356467$e-4$	3.625604$e-5$	0
	Mean	4.623815$e-4$	5.019091$e-40$	3.351138$e-18$	1.387048$e-4$	3.721975$e-5$	0
	Worst	4.794164$e-4$	7.370712$e-40$	1.821983$e-17$	1.411770$e-4$	3.830758$e-5$	0
	Std.	1.432076$e-6$	9.099148$e-41$	3.523805$e-18$	1.432076$e-6$	4.633699$e-7$	0
f_2	Best	27.244893	26.430489	26.945519	27.752978	28.688393	0
	Mean	28.693051	28.056813	27.956949	28.228172	28.722672	0
	Worst	32.020138	29.377247	28.994375	28.596902	28.774159	0
	Std.	0.892672	0.640186	0.497804	0.232890	0.024310	0
f_3	Best	0.133342	0.029958	3.335927$e-4$	0.039419	1.503231$e-5$	0
	Mean	0.138847	0.923459	0.006701	0.040155	2.124765$e-5$	0
	Worst	0.143332	4.750825	0.036739	0.040996	3.583190$e-5$	0
	Std.	0.001782	1.030883	0.007965	3.230054$e-4$	4.625523$e-6$	0
f_4	Best	0.118299	2.814927$e-13$	7.590946$e-9$	0.063400	0.035800	0
	Mean	0.119498	0.010992	4.956341$e-6$	0.064237	0.035856	0
	Worst	0.121374	0.103356	2.242247$e-4$	0.064880	0.035957	0
	Std.	6.145446$e-4$	0.021724	3.133953$e-5$	2.749579$e-4$	3.254229$e-5$	0
f_5	Best	2.409987$e-5$	0	1.920686$e-14$	6.970412$e-6$	1.028335$e-6$	0
	Mean	2.481638$e-5$	0.005076	0.003841	7.121020$e-6$	1.226157$e-6$	0
	Worst	2.562478$e-5$	0.019697	0.024573	7.284919$e-6$	1.517875$e-6$	0
	Std.	3.167733$e-7$	0.005384	0.006266	7.113268$e-8$	9.531260$e-8$	0
f_6	Best	0.092326	0	0	0.026792	0.008531	0
	Mean	11.785692	5.968559$e-15$	6.430412$e-15$	0.027441	0.008581	0
	Worst	79.015636	1.598721$e-14$	1.598721$e-14$	0.028106	0.008613	0
	Std.	16.989014	3.440084$e-15$	4.603335$e-15$	2.965811$e-4$	1.659589$e-5$	0
f_7	Best	0.016704	1.509903$e-14$	1.443619$e-10$	0.008752	0.004740	8.881784$e-16$
	Mean	0.016867	2.376765$e-14$	4.627204$e-10$	0.008835	0.004774	8.881784$e-16$
	Worst	0.017080	3.996803$e-14$	1.311684$e-09$	0.008934	0.004807	8.881784$e-16$
	Std.	9.602465$e-5$	6.518431$e-15$	2.675685$e-10$	4.148766$e-5$	1.263296$e-5$	0
f_8	Best	9.296021$e-4$	0	0	2.719695$e-4$	7.178698$e-5$	0
	Mean	0.067527	1.554312$e-17$	1.831868$e-16$	2.772358$e-4$	7.458798$e-5$	0
	Worst	1.109406	2.220446$e-16$	8.881784$e-16$	2.829657$e-4$	7.696815$e-5$	0
	Std.	0.163524	5.329071$e-17$	2.332129$e-16$	2.697164$e-6$	9.418096$e-7$	0

solutions obtained by each algorithm is presented in Table 5. According to the inverse kinematics solutions in Table 5, the poses of the end-effector are obtained as shown in Figure 7. The pose errors of different algorithms are listed in Table 6.

The SR, i.e., the percentage of trials where algorithms converge with a specified accuracy (in this study it is assigned a value of 10^{-9}), is defined by

$$\mathrm{SR} = \frac{n'}{n} \times 100\%, \tag{26}$$

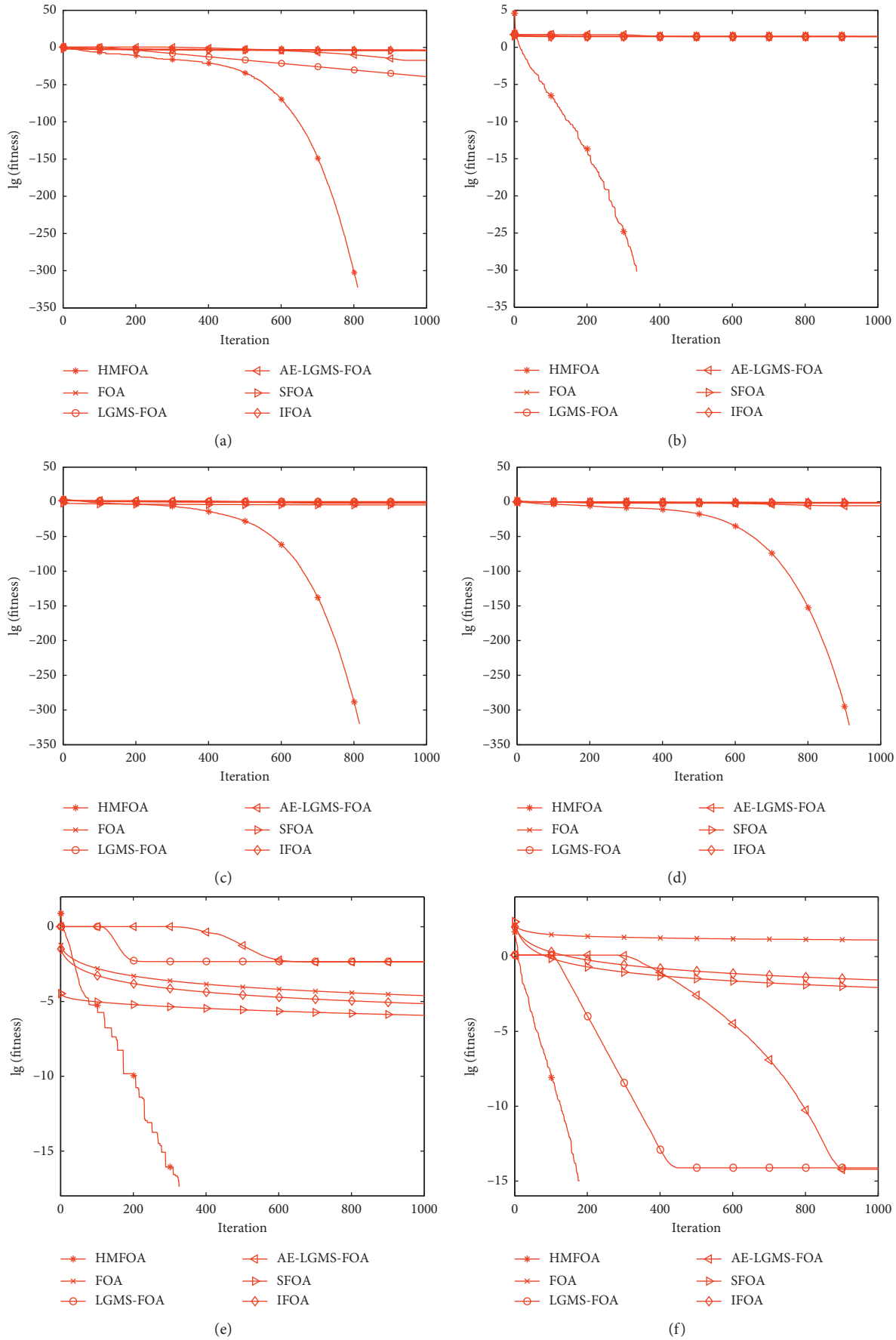

Figure 5: Continued.

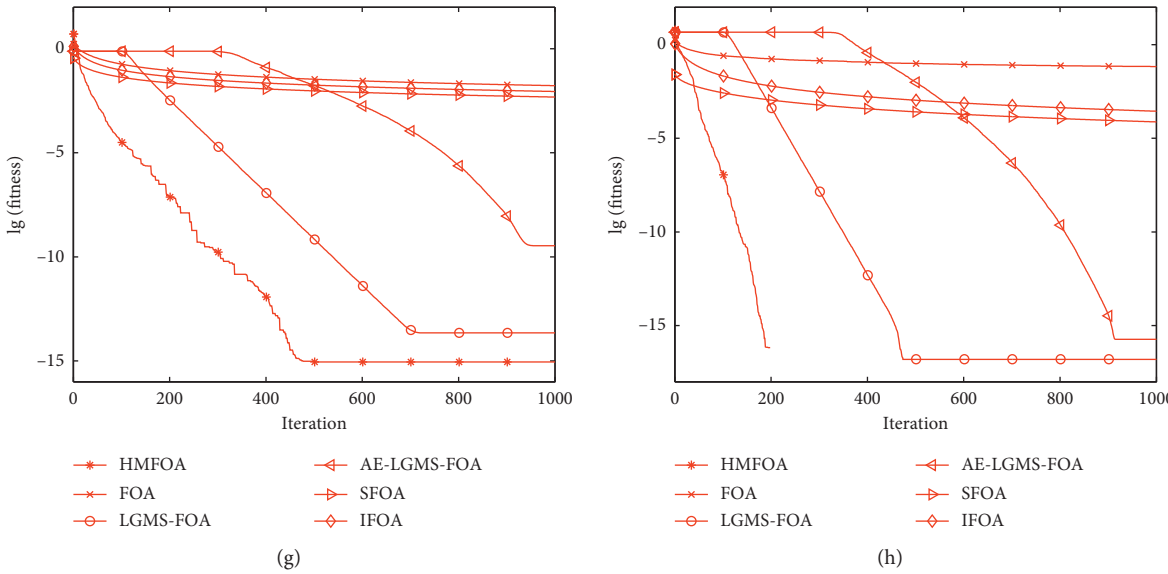

Figure 5: Average convergence curves for the selected functions. (a) f_1. (b) f_2. (c) f_3. (d) f_4. (e) f_5. (f) f_6. (g) f_7. (h) f_8.

Table 4: Performance statistics of different algorithms.

	FOA	LGMS-FOA	AE-LGMS-FOA	IFOA	SFOA	HMFOA
Best	2.3664845407	$8.6689214717e-3$	$8.0997417917e-4$	5.8509445796	5.1327857288	$4.8294701571e-16$
Worst	5.8858025146	1.5095793139	1.3589677774	6.1056362998	8.3300837795	$9.1473267081e-2$
Mean	5.7220157213	$4.5801009616e-1$	$3.1026807967e-1$	5.9634463791	7.0334301557	$2.1510704486e-3$
Std	$4.8093695407e-1$	$4.2380970796e-1$	$3.0905484214e-1$	$5.3295364512e-2$	$6.6812284973e-1$	$1.2857130856e-2$
SR (%)	0	0	0	0	0	94

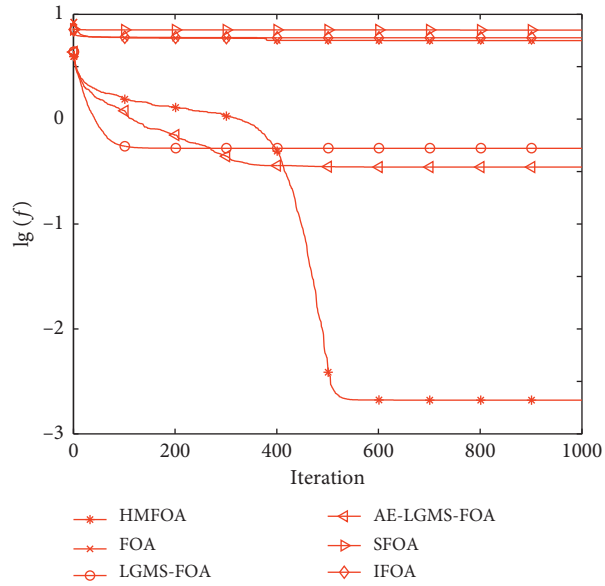

Figure 6: Average convergence curves for a redundant robot manipulator.

where n is the total number of trials and $n\prime$ is the number of successful trials.

According to Table 4, the best value, worst value, mean value, standard deviation, and successful ratio of HMFOA are superior to those of other algorithms involved in comparison. The successful ratio of HMFOA is up to 94%, while the successful ratios of FOA, LGMS-FOA, AE-LGMS-FOA, IFOA, and SFOA are all zero. The best values of LGMS-

Table 5: An inverse kinematics solution corresponding to the best fitness of each algorithm.

	FOA	LGMS-FOA	AE-LGMS-FOA	IFOA	SFOA	HMFOA
θ_1/rad	0.0976126056	−1.0839133510	−2.6595857928	0.0879773602	0.2928215864	−2.7714570693
θ_2/rad	−1.0722179601	−2.4772682366	−1.3854268665	0.1076406335	−0.3252784200	−1.2864166263
θ_3/rad	0.1066073486	−0.4238387176	−0.8803102621	0.0778478430	0.2793738833	−1.1659304990
θ_4/rad	5.0187740095	−4.5222500892	−4.8530768097	0.0983433196	0.5282601969	1.4498210529
θ_5/rad	0.0666429414	1.3790609675	0.0041955026	0.3085488511	0.3596046188	0.3811435232
θ_6/rad	0.0906599000	1.3828761741	3.3703822119	0.0943219901	−0.2018160421	3.4299412880
θ_7/rad	0.9180238304	0.3766713486	−1.0054925913	0.0726914072	0.5789474603	−0.3168192999

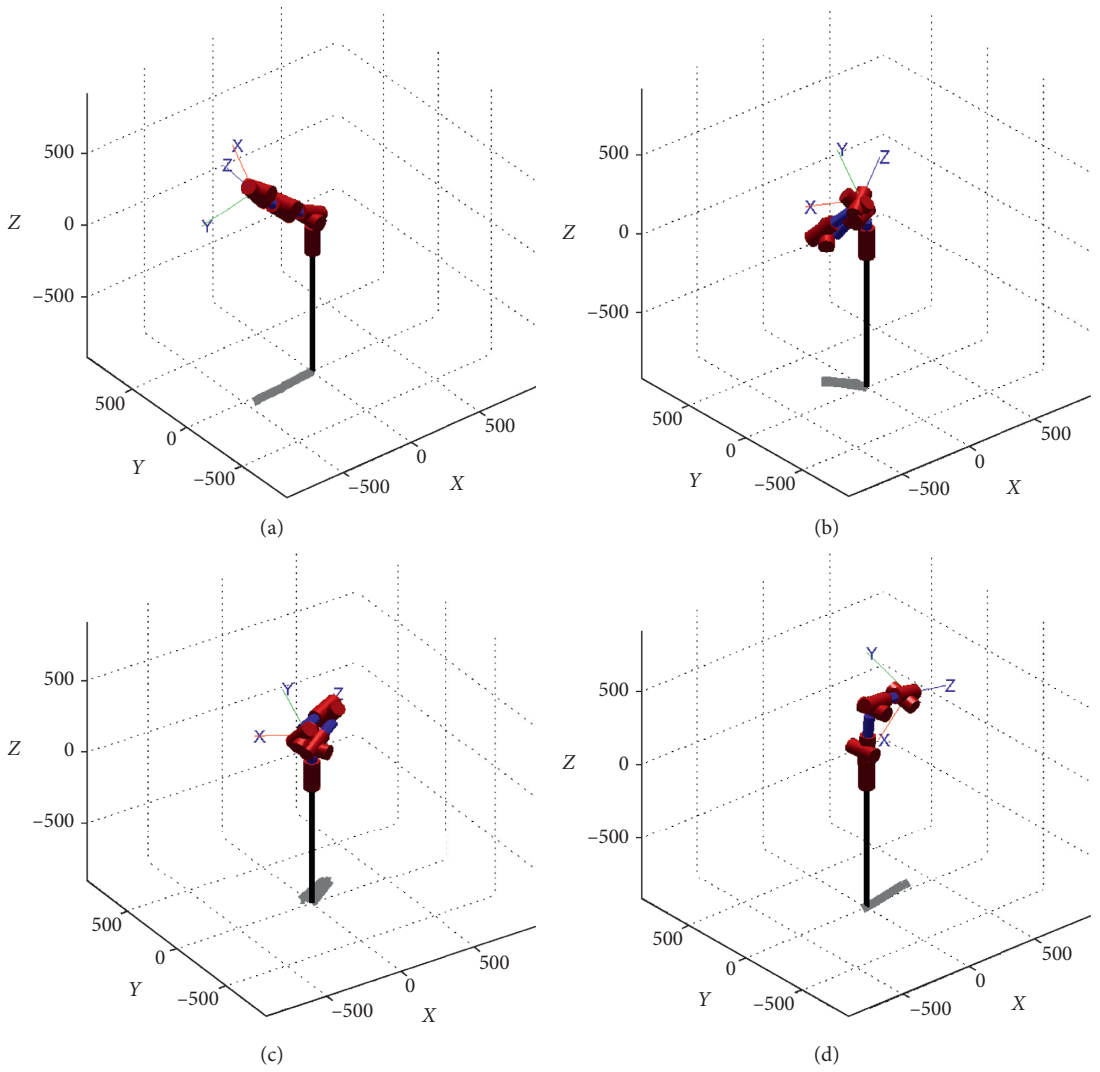

Figure 7: Continued.

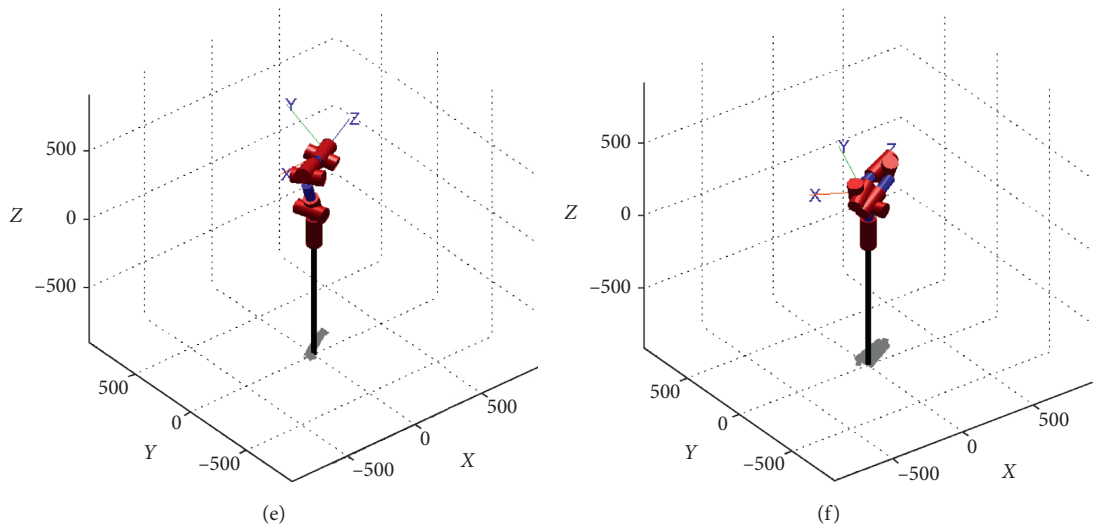

Figure 7: Poses of the end-effector corresponding to different algorithms. (a) FOA. (b) LGMS-FOA. (c) AE-LGMS-FOA. (d) IFOA. (e) SFOA. (f) HMFOA.

Table 6: The pose errors of different algorithms.

	FOA	LGMS-FOA	AE-LGMS-FOA	IFOA	SFOA	HMFOA
$n_x^* - n_x$	$-7.3284224305e-1$	$5.5511151231e-17$	$-5.5511151231e-17$	$-9.0361801676e-2$	$-6.9918463297e-2$	0
$n_y^* - n_y$	$-7.1127054555e-2$	$-1.1102230246e-16$	0	$5.1834675479e-1$	$1.9132713058e-1$	0
$n_z^* - n_z$	$-5.8011245465e-1$	$-5.5511151231e-17$	$-1.1102230246e-16$	$5.6375175368e-1$	$3.4739363429e-1$	0
$o_x^* - o_x$	$3.5307444229e-1$	$-5.5511151231e-17$	$-5.5511151231e-17$	$-2.9259749224e-1$	$-6.5206087533e-2$	$-5.5511151231e-17$
$o_y^* - o_y$	$-0.2393223149e-1$	$-2.7755575616e-17$	$-8.3266726847e-17$	$-7.6512409616e-1$	$-4.3453605161e-1$	$-1.1102230246e-16$
$o_z^* - o_z$	1.4888118509	0	$1.1102230246e-16$	$5.4731325106e-1$	$1.7174065382e-1$	0
$a_x^* - a_x$	1.4548948573	$2.2204460493e-16$	$1.1102230246e-16$	$-1.1134284726e-1$	$-5.7455356949e-2$	$1.1102230246e-16$
$a_y^* - a_y$	$5.2392832745e-1$	$-1.1102230246e-16$	$-2.2204460493e-16$	$3.2173407939e-1$	$4.3062832710e-2$	0
$a_z^* - a_z$	$-5.4001598940e-1$	0	$-2.7755575616e-17$	$5.2885327132e-1$	$2.5711680884e-1$	$2.7755575616e-17$
$p_x^* - p_x$	$4.6935429781e+2$	$5.3290705182e-14$	$-8.0995335681e-2$	$-3.5061043328e+2$	$-1.9242055348e+2$	$-3.5527136788e-15$
$p_y^* - p_y$	$1.0069936321e+2$	$-8.6290277692e-1$	$-2.0822353690e-6$	6.4331576044	$-8.5433896652e+1$	$-1.4210854715e-14$
$p_z^* - p_z$	$-2.9740412354e+2$	$3.9893702441e-3$	0	$-1.3976150052e+2$	$-1.6629175959e+2$	0

FOA and AE-LGMS-FOA are 13 and 12 orders of magnitude worse than HMFOA, respectively, while the optimization results of FOA, IFOA, and SFOA are even worse. It can be seen from Figure 6 that the HMFOA has faster convergence speed and higher convergence accuracy than the other five algorithms (to facilitate evaluation and observation, the fitness of the objective function in the graph is a logarithm with the base of 10). From Figure 7, we can directly observe the actual pose of each inverse kinematics solution.

As can be seen from Table 6, the position errors corresponding to the optimal inverse solutions of FOA, IFOA, and SFOA are 10^2 mm, 10^2 mm, and 10^2 mm, respectively. Due to the relatively large position errors, the inverse kinematic solutions obtained by FOA, IFOA, and SFOA are unacceptable. The inverse kinematics solutions corresponding to LGMS-FOA and AE-LGMS-FOA have achieved very high orientation accuracy, and the corresponding maximum errors of the position components are orders of magnitude 10^{-1} mm and 10^{-2} mm, respectively. The orientation and position accuracy corresponding to the optimal inverse kinematics solution of HMFOA are rather higher than other methods mentioned above, which indicates that the HMFOA can be used to effectively solve the inverse kinematics problem of redundant manipulator.

In a word, the calculation results of HMFOA are better than those of FOA, LGMS-FOA, AE-LGMS-FOA, IFOA, and SFOA in the inverse kinematics solution of the 7-DOF manipulator.

5. Conclusions

In this study, the inverse kinematics problem of the manipulator is transformed into a minimum optimization problem, and an improved fruit fly optimization algorithm, namely, the hybrid mutation fruit fly optimization algorithm, is proposed for resolving this problem. The olfactory search mechanism based on hybrid coevolution of multiple mutation strategies is employed in the algorithm, which can effectively balance the global and local search of the algorithm. Simultaneously, the premature convergence problem of the algorithm can also be effectively solved. Through the dynamic real-time update of visual search, the successful search experience of each fruit fly individual is fully utilized. Thus, the search efficiency of the swarm can be effectively improved. The simulation results of 8 typical benchmark functions indicate the feasibility and effectiveness of the proposed algorithm. The inverse kinematics of a 7-DOF redundant manipulator is taken as an example for

experimental simulation. The comparisons of HMFOA with FOA and other variants of FOA illustrate that HMFOA is extremely suitable for solving kinematic problems of redundant manipulator.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (51566012), the Joint Foundation of Guizhou Province (LH word [2015]7302), and the Special Funding of Guiyang Science and Technology Bureau and Guiyang University (GYU-KYZ (2019~2020) DT-13).

References

[1] S. Kucuk and Z. Bingul, "Inverse kinematics solutions for industrial robot manipulators with offset wrists," *Applied Mathematical Modelling*, vol. 38, no. 7-8, pp. 1983–1999, 2014.

[2] X. H. Xie, S. M. Fan, X. Y. Zhou, and Z. Y. Li, "Inverse kinematics of manipulator based on the improved differential evolution algorithm," *Robot*, vol. 41, no. 1, pp. 50–57, 2019, in Chinese.

[3] D. Manocha and J. F. Canny, "Efficient inverse kinematics for general 6R manipulators," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 648–657, 1994.

[4] R. Featherstone, "Position and velocity transformations between robot end-effector coordinates and joint angles," *The International Journal of Robotics Research*, vol. 2, no. 2, pp. 35–45, 1983.

[5] J. Craig, *Introduction to Robotics*, China Machine Press, Beijing, China, 2018, in Chinese, 4th edition.

[6] S. Momani, Z. S. Abo-hammour, and O. M. Alsmadi, "Solution of inverse kinematics problem using genetic algorithms," *Applied Mathematics & Information Sciences*, vol. 10, no. 1, pp. 1–9, 2016.

[7] Y. Lin, H. Zhao, and H. Ding, "Solution of inverse kinematics for general robot manipulators based on multiple population genetic algorithm," *Journal of Mechanical Engineering*, vol. 53, no. 3, pp. 1–8, 2017, in Chinese.

[8] H. C. Huang, C. P. Chen, and P. R. Wang, "Particle swarm optimization for solving the inverse kinematics of 7-DOF robotic manipulators," in *Proceedings of 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 3105–3110, Seoul, Korea, October 2012.

[9] N. Rokbani and A. M. Alimi, "Inverse kinematics using particle swarm optimization, a statistical analysis," *Procedia Engineering*, vol. 64, pp. 1602–1611, 2013.

[10] I. Sancaktar, B. Tuna, and M. Ulutas, "Inverse kinematics application on medical robot using adapted PSO method," *Engineering Science and Technology, an International Journal*, vol. 21, no. 5, pp. 1006–1010, 2018.

[11] S. Dereli and R. Köker, "IW-PSO approach to the inverse kinematics problem solution of a 7-DOF serial robot manipulator," *Sigma Journal of Engineering and Natural Sciences*, vol. 36, pp. 77–85, 2018.

[12] S. Dereli and R. Köker, "A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: quantum behaved particle swarm algorithm," *Artificial Intelligence Review*, vol. 53, pp. 949–964, 2020.

[13] C. G. Uzcátegui and D. B. Rojas, "A memetic differential evolution algorithm for the inverse kinematics problem of robot manipulators," *International Journal of Mechatronics and Automation*, vol. 3, no. 2, pp. 118–131, 2013.

[14] C. López-Franco, J. Hernández-Barragán, A. Y. Alanis, N. Arana-Daniel, and M. López-Franco, "Inverse kinematics of mobile manipulators based on differential evolution," *International Journal of Advanced Robotic Systems*, vol. 15, no. 1, 2018.

[15] A. El-sherbiny, M. A. Elhosseini, and A. Y. Haikal, "A new ABC variant for solving inverse kinematics problem in 5 DOF robot arm," *Applied Soft Computing*, vol. 73, pp. 24–38, 2018.

[16] Z.-w. Ren, Z.-h. Wang, and L.-n. Sun, "A hybrid biogeography-based optimization method for the inverse kinematics problem of an 8-DOF redundant humanoid manipulator," *Frontiers of Information Technology & Electronic Engineering*, vol. 16, no. 7, pp. 607–616, 2015.

[17] 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*, vol. 52, no. 10, pp. 176–181, 2019.

[18] W.-T. Pan, "A new fruit fly optimization algorithm: taking the financial distress model as an example," *Knowledge-Based Systems*, vol. 26, pp. 69–74, 2012.

[19] X. Yuan, X. Dai, J. Zhao, and Q. He, "On a novel multi-swarm fruit fly optimization algorithm and its application," *Applied Mathematics and Computation*, vol. 233, pp. 260–271, 2014.

[20] J. Niu, W. Zhong, Y. Liang, N. Luo, and F. Qian, "Fruit fly optimization algorithm based on differential evolution and its application on gasification process operation optimization," *Knowledge-Based Systems*, vol. 88, pp. 253–263, 2015.

[21] R. Hu, S. Wen, Z. Zeng, and T. Huang, "A short-term power load forecasting model based on the generalized regression neural network with decreasing step fruit fly optimization algorithm," *Neurocomputing*, vol. 221, pp. 24–31, 2017.

[22] X.-l. Zheng, L. Wang, and S.-y. Wang, "A novel fruit fly optimization algorithm for the semiconductor final testing scheduling problem," *Knowledge-Based Systems*, vol. 57, pp. 95–103, 2014.

[23] L. Wang, Y. Shi, and S. Liu, "An improved fruit fly optimization algorithm and its application to joint replenishment problems," *Expert Systems with Applications*, vol. 42, no. 9, pp. 4310–4323, 2015.

[24] D. Shan, G. Cao, and H. Dong, "LGMS-FOA: an improved fruit fly optimization algorithm for solving optimization problems," *Mathematical Problems in Engineering*, vol. 2013, Article ID 108768, 9 pages, 2013.

[25] A. Darvish and A. Ebrahimzadeh, "Improved fruit-fly optimization algorithm and its applications in antenna arrays synthesis," *IEEE Transactions on Antennas and Propagation*, vol. 66, no. 4, pp. 1756–1766, 2018.

[26] A. Babalik, H. İşcan, İ. Babaoğlu, and M. Gündüz, "An improvement in fruit fly optimization algorithm by using sign parameters," *Soft Computing*, vol. 22, no. 22, pp. 7587–7603, 2018.