IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 3 June 2024, accepted 24 June 2024, date of publication 1 July 2024, date of current version 9 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3420902

RESEARCH ARTICLE

β-FGJO: A General Metaheuristic Method for Inverse Kinematics Solution of Multi-DOF Robotic Manipulators

XIAO-YU ZHANG[1,2], ZHONG-QING FANG[3], JIA-PAN LI[4], WEI-BIN KONG[3], YI DU[3], TING-LIN ZHANG[3], AND RU-GANG WANG[3]

[1]School of Mechanical Engineering, Yancheng Institute of Technology, Yancheng 224051, China
[2]UGS College, Yancheng Institute of Technology, Yancheng 224000, China
[3]School of Information Engineering, Yancheng Institute of Technology, Yancheng 224051, China
[4]College of Information and Communication Engineering, Harbin Engineering University, Harbin 150001, China

Corresponding authors: Zhong-Qing Fang (fangzq@ycit.edu.cn) and Wei-Bin Kong (kongweibin@ycit.cn)

This work was supported in part by the Postgraduate Research and Practice Innovation Program of Jiangsu Province under Grant SJCX23-1871, Grant SJCX23-XY069, and Grant SJCX23-XY071; and in part by the College Students Innovation and Entrepreneurship Training Program under Grant 2023591 and Grant 2023576.

ABSTRACT Robotic manipulators play a crucial role in providing support for automation and intelligence in various fields. The inverse kinematics problem becomes a significant challenge for modern robotic manipulator systems. This work proposes an enhanced Golden Jackal Optimization (β-FGJO) to solve the inverse kinematics problem in multi-degree-of-freedom (multi-DOF) robotic manipulators. In β-FGJO, the Fuch chaotic map is utilized to generate an efficient initial population to enhance search efficiency. Individual behavior is regulated by the adaptive β-distribution to improve both global exploration and local exploitation capabilities at different stages. Meanwhile, predators and prey in the population dynamically explore and exploit based on their energy level and hunger rate. Simulation results demonstrate that β-FGJO has shorter computation time, higher numerical precision and greater robustness. Compared to the best-performing method on PUMA560, β-FGJO improved time performance by 24.57%, while maintaining the same level of accuracy.

INDEX TERMS Adaptive β-distribution, enhanced golden jackal optimization, manipulator, metaheuristic, robot kinematics.

I. INTRODUCTION

The application of robotic manipulators spans diverse domains, solidifying them as indispensable instruments within contemporary industrial and scientific research. Robotic manipulators serve as pivotal components within the realm of industrial automation, capable of executing multifarious intricate production tasks [1]. Simultaneously, they assume a paramount role in high-precision applications such as microelectronic manufacturing [2], space science research [3], and surgical procedures [4]. Accurate and effective inverse kinematics solutions are required to achieve precise control of robotic manipulators. Solving the inverse kinematics problem entails the computation of the robot joints' configuration according to the intended position or posture of the end effector. Traditional methods rely on complex algebraic or differential equation systems [5], [6], [7]. The analytical solution can be computed only when satisfying the Pieper criterion [8]. The complexity of analytical methods becomes evident when applied to robotic manipulators with multiple degrees of freedom or configured in parallel arrangements. In certain cases of specific robotic structures and configurations, traditional analytical methods may even lead to numerical instability or singularity problem [9].

In contemporary research, the inverse kinematics problem is commonly approached as a nonlinear single objective optimization problem, where the objective function is

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

conceived as a representation of position and orientation errors of the end effector. The main objective of optimization is to minimize these errors and achieve precise control of the end effector. An effective method is to transform the inverse kinematics problem into a nonlinear single objective optimization problem, which exhibits obvious advantages in adaptability, universality, and numerical stability. Consequently, it is particularly well-suited for modern complex robotic systems. Traditional optimization methods [10], [11], [12] often show poor performance in intricate systems. They are susceptible to converge towards local optima, limiting their effectiveness. With the development of evolutionary computation, the ascendance of meta-heuristic algorithms is notable, primarily due to their robust performance. Meta-heuristic algorithms [13] are capable of searching through large and complex solution spaces to find optimal or near-optimal solutions and do not require smooth and continuous objective functions, making them suitable for problems with non-linearity, multi-modality, and discontinuities. Notable instances of meta-heuristic algorithms encompass Particle Swarm Optimization (PSO) [14], Grey Wolf Optimizer (GWO) [15], Ant Colony Optimization (ACO) [16], Whale Optimization Algorithm (WAO) [17], as well as recently introduced methods such as the Golden Jackal Optimization (GJO), Artificial Gorilla Troops Optimizer (GTO) [18] and African Vulture Optimization Algorithm (AVOA) [19]. Although meta-heuristic algorithms have good performance, there are still challenges in terms of optimization accuracy and solution time, especially in complex and high-dimensional robot systems. Existing methods often struggle with achieving a balance between global exploration and local exploitation, leading to suboptimal solutions or longer convergence times. More efficient and accurate metaheuristic algorithm for solving inverse kinematics problems is required in complex robot systems.

This study proposes the β-FGJO which is based on the framework of Golden Jackal Optimization (GJO). β-FGJO enhances the optimization accuracy and efficiency of solving inverse kinematics problems for robotic manipulators. By introducing the Fuch Chaotic Map and the adaptive β-distribution, β-FGJO improves the distribution of the initial population and regulates the search space dynamically, enhancing both global exploration and local exploitation at different stages of the optimization process. This leads to more reliable and faster convergence to optimal or near-optimal solutions.

The following sections outline the work as follows: Section II reviews the state and the main contributions of meta-heuristic algorithm for inverse kinematics problems. Section III analyzes the kinematics of robotic manipulators and optimization model of inverse Kinematics. Section IV introduces β-FGJO. Section V describes the experimental design, parameter settings and provides results analysis of experiments. In Section VI, β-FGJO is compared to methods that have demonstrated good performance in the literature. Section VII provides the conclusion and prospects for future work.

II. RELATED WORK

Meta-heuristic algorithms have been applied in numerous research endeavors. Particle Swarm Optimization (PSO), a widely acclaimed meta-heuristic algorithm, has been extensively employed to solve the inverse kinematics problem for a long time. The use of PSO in inverse kinematics can be traced back to Rokbani and Alimi's research [20]. Notably, the comparative analysis of PSO variants revealed that PSO-VG exhibits rapid convergence compared to other PSO variants. Reyes and Gardini [21] introduced an improved PSO with inertia weight and acceleration coefficients, which demonstrated superior performance compared to conventional algorithms. Ram et al. [22] employed bidirectional particle swarm optimization for mobile manipulator inverse kinematics, facilitating bidirectional search capabilities. Deng and Xie [23] explored the application of adaptive PSO (APSO) with an inertia weight strategy, which can achieve accurate joint configurations for the desired position and orientation of a 6-DOF robotic manipulator. Dereli and Köker [24] employed two PSO variants for the inverse kinematics of a novel 7-revolute jointed robotic manipulator, highlighting the enhanced effectiveness of PSO variables over standard PSO. The utilization of Quantum Behaved Particle Swarm Optimization (QPSO) [25] on the inverse kinematics of a 7-DOF serial manipulator demonstrated its computational efficiency with shorter computation time, fewer iterations, and fewer particles. Ayyıldız and Cetinkaya [26] compared optimization algorithms on a 4-DOF serial robotic manipulator and QPSO's performance was proved to be superior to GA, GSA and PSO. Meanwhile, a spectrum of other meta-heuristic algorithms has been applied to solve the inverse kinematics of robotic manipulators too. Firefly Algorithm (FA) [27] and Artificial Bee Colony (ABC) [28] also exhibited efficacy for the inverse kinematics of a 7-DOF redundant robotic manipulator. However, the Strengthened PSO, inspired by new techniques, outperforms FA and ABC in the Dereli and Köker's [29] reasearch. Çavdar et al. [30] introduced a modified ABC for solving inverse kinematics problems in robotic manipulators. The modified ABC demonstrated improved performance in both position accuracy and solution time. Sui et al. [31] proposed an inverse kinematics algorithm based on Multiple Population Genetic Algorithm (MPGA), capable of calculating all globally optimal solutions for a general geometric structure with pose errors up to two decimal places. In subsequent research, Köker and Çavdar [32] also presented a hybrid intelligent solution system integrating neural networks, genetic algorithms and simulated annealing for robotic manipulator inverse kinematics, achieving micrometer-level error. Rokbani et al. [33] introduced a novel inverse kinematics solver based on beta distributed Salp Swarm Algorithm

TABLE 1. Well-performing methods proposed in the related work.

Research	DOF	Method	Ave. Error(m)	Ave. Time(s)
S.V.Reyes [21]	5DOF	Improved PSO	2.08E-14	3.7
T.Çavdar [30]	6DOF	Improved ABC	6.31E-13	0.0286
S.V.Reyes [22]	6DOF	Improved PSO	2.17E-16	3.63
S.Dereli [24]	7DOF	IW-PSO	1.00E-16	0.65
S.Dereli [25]	7DOF	QPSO	2.78E-17	0.2319

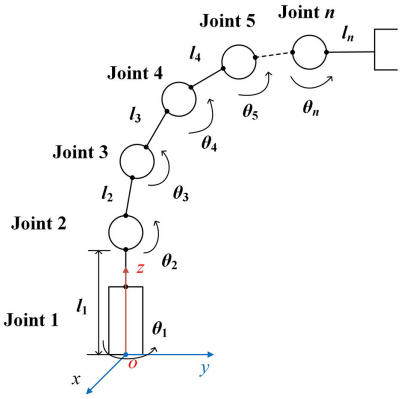

FIGURE 1. Structure of the Multi-DOF Generic Robotic Manipulator (MDOF-GRM).

(β-SSA) which outperformed classical SSA, QPSO, Bi-PSO, K-ABC, and FA on the 8-DOF robotic manipulator and Kr05 industrial robot. Zhao et al. [34] used an improved particle swarm algorithm for robot inverse kinematics on PUMA560, ensuring higher position and orientation accuracy of the robotic arm end effector. In Table 1, well-performing methods in the current related work are listed in with the average error and average time which are given in their paper.

III. MODELING AND KINEMATICS ANALYSIS

Robotic manipulators are widely used in current research as they are popular types in the field of robotics. These manipulators have many advantages, such as ease of avoiding obstacles, flexible movement, and a larger workspace. Despite all these advantages, their structure is very complex. According to research [22], [25], [30], [33], we use the multi-DOF generic robotic manipulator (MDOF-GRM) through simulation and verify on the PUMA560.

A. THE MULTI-DOF GENERIC ROBOTIC MANIPULATOR

The multi-DOF generic robotic manipulator (MDOF-GRM) serves as a standardized test platform that offers a consistent and well-defined test environment across various degrees of freedom. In Fig.1, each link of the MDOF-GRM has a uniform length of 1m and the joint angles vary from $-180°$ to $180°$. The DOF of the MDOF-GRM can expand with a range spanning from 2 to n, depending on the specific operational configuration. The spatial coordinates of the end effector can be calculated using Eq.(1), which is derived from geometric relationships.

$$\begin{cases} p_x^n = \sum_{i=2}^{n} \left(l_i \cos\left(\sum_{j=2}^{i} \theta_j \right) \right) \times \cos\theta_1 \\ p_y^n = \sum_{i=2}^{n} \left(l_i \cos\left(\sum_{j=2}^{i} \theta_j \right) \right) \times \sin\theta_1 \\ p_z^n = l_1 + \sum_{i=2}^{n} \left(l_i \sin\left(\sum_{j=2}^{i} \theta_j \right) \right) \end{cases} \tag{1}$$

In Eq.(1), P_x, P_y, P_z are the relative positions of end effector on n DOF, according to the frame axes (o, x, y, z), θ_j is the angle of the joint j.

B. THE PUMA560

The PUMA560 [35] was designed and manufactured by Unimation in 1981 which is a 6-DOF robotic manipulator and has been widely used in the manufacture industry. DH parameters were developed by Denavit and Hartenberg [36]. These parameters widely used in the field of robotics modeling. The DH coordinate system of the PUMA560 is shown in Fig.2 and DH parameters are given in the Table 2, where $a_2 = 0.4318$m, $a_3 = 0.15005$m, $d_2 = 0.15005$m, $d_4 = 0.4318$m, $d_6 = 0.05625$m.

$$\mathbf{T}_1^0 = \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$\mathbf{T}_2^1 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$\mathbf{T}_3^2 = \begin{bmatrix} c_3 & 0 & -s_3 & a_3c_3 \\ s_3 & 0 & c_3 & a_3s_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$\mathbf{T}_4^3 = \begin{bmatrix} c_4 & 0 & s_4 & 0 \\ s_4 & 0 & -c_4 & 0 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$\mathbf{T}_5^4 = \begin{bmatrix} c_5 & 0 & -s_5 & 0 \\ s_5 & 0 & c_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$\mathbf{T}_6^5 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$\mathbf{T}_6^0 = \mathbf{T}_1^0\mathbf{T}_2^1\mathbf{T}_3^2\mathbf{T}_4^3\mathbf{T}_5^4\mathbf{T}_6^5 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

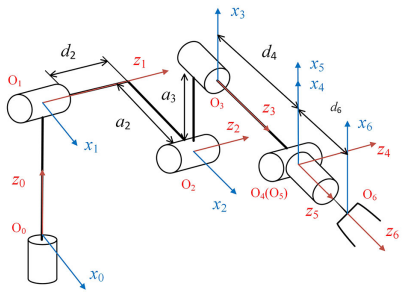

FIGURE 2. The PUMA560 with DH coordinate system.

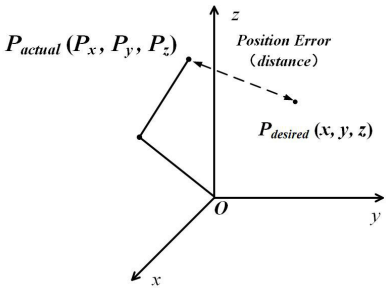

FIGURE 3. Position error.

TABLE 2. DH parameters of the PUMA560.

Joint i	θ_i(∘)	d_i(m)	a_i(m)	α_i(∘)	Range(∘)
1	θ_1	0	0	-90	-160 to 160
2	θ_2	d_2	a_2	0	-225 to 45
3	θ_3	0	a_3	-90	-45 to 225
4	θ_4	d_4	0	90	-110 to 170
5	θ_5	0	0	-90	-100 to 100
6	θ_6	d_6	0	0	-266 to 266

Then the conversion matrix can be obtained by Eqs.(2)-(8), where P_x, P_y, P_z are the relative positions of end effector, according to the frame axes (Oo, x, y, z), is detailed in Eq.(9) and Eq.(10).

$$\begin{cases} p_x = a_2c_1c_2 - d_4c_1s_{23} - d_2s_1 \\ \quad -d_6\left(s_5\left(s_1s_4 + c_4c_1c_{23}\right) + c_5c_1s_{23}\right) + a_3c_1c_{23} \\ p_y = d_6\left(s_5\left(c_1s_4 - c_4s_1c_{23}\right) - c_5s_1s_{23}\right) \\ \quad -d_4s_1s_{23} + d_2c_1 + a_2c_2s_1 + a_3s_1c_{23} \\ p_z = -d_4c_{23} - a_2s_3 - d_6\left(c_5c_{23} - c_4s_5s_{23}\right) - a_3s_{23} \end{cases} \tag{9}$$

$$\begin{cases} s_i = \sin(\theta_i),\ c_i = \cos(\theta_i) \\ s_{ij} = \sin(\theta_i + \theta_j),\ c_{ij} = \cos(\theta_i + \theta_j) \end{cases} \tag{10}$$

C. OPTIMIZATION MODEL FOR SOLVING INVERSE KINEMATICS

The joint angles determine the relative position and orientation of each link, thereby influencing the overall position and orientation of the end effector. These factors ultimately serve as constraints on joint angles in the inverse kinematics problem. However, related works [20], [22], [24], [25], [26], [27], [28], [29], [30], [31], [33] used position errors as the optimization objective function without considering other constraints. The study focus on the performance of relevant metaheuristic algorithms in solving inverse kinematics, particularly in improving optimization accuracy and solution time. Based on this, our research also simplifies the orientation constraint. Additionally, employing the same objective function facilitates consistent comparison of results across different methods or algorithms. This uniformity in evaluation criteria ensures a fair and objective assessment of different approach's performance on solving the inverse kinematics problem.

As is shown in Fig.3, the position error is the distance between P_{actual} and $P_{desired}$. P_{actual} represents the actual position of the end effector and $P_{desired}$ is the desired position that the robotic manipulator aims to reach. The primary goal is to determine the joint angles that minimize the position error between P_{actual} and $P_{desired}$. The position error can be calculated using Eq.(11). Therefore, Eq.(11) serves as the objective function to be optimized.

$$Error = \sqrt{(P_x - x)^2 + \left(P_y - y\right)^2 + (P_z - z)^2} \tag{11}$$

In Eq.(11), (P_x, P_y, P_z) is the position of P_{actual} and (x, y, z) is the position of $P_{desired}$.

IV. THE β-FGJO ALGORITHM

In this section, we first provide motivation and a detailed explanation of β-FGJO, followed by a thorough introduction of the β-FGJO algorithm. In subsection B, we explaine the initialization based on the Fuch chaotic map. Subsequently, in subsection C, we introduce a three-stage strategy to update the population in β-FGJO. Additionally, subsection D covers the Elite Strategy, while subsection E provides a detailed explanation of the principles behind the Adaptive Search Strategy.

A. INSPIRATION

Chopra and Ansari [37] proposed the Golden Jackal Optimization (GJO) by simulating hunting behavior of golden jackals. The nominal GJO typically has few control parameters, a simple structure and efficient convergence, making it easier to implement and tune for specific problems. However, the performance of GJO is suboptimal when dealing with the inverse kinematics problem. The position update mechanism of GJO is too simple and depends on the jackal pair positions which reduces the diversity of the prey population. This constrains the search ability of GJO and make it easy to trap in local optimum. The optimizing effectiveness of GJO for inverse kinematics solving optimization deserves further discussion.

The original GJO algorithm conducts global and local searches in two separate stages and requires fewer parameters, but it has poor accuracy. The β-FGJO extends

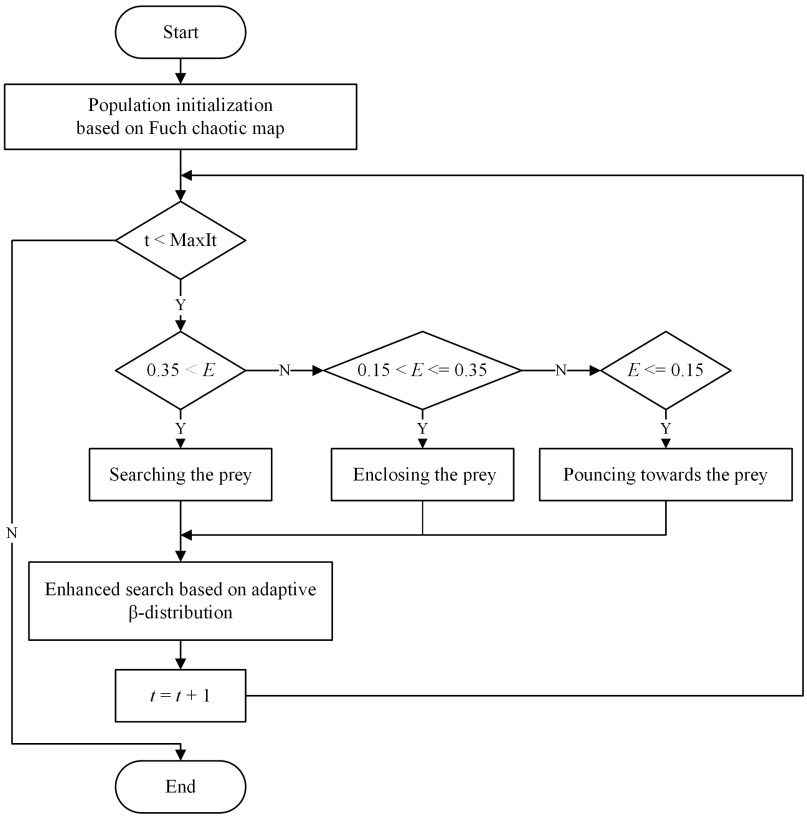

FIGURE 4. The flowchart of β-FGJO.

the original GJO into a three-stage search and redesigns the population update methods for each stage. In these strategies, some effective schemes from other algorithms are incorporated. Additionally, new strategies are introduced, such as the Fuch chaotic map to generate the initial population in the initialization phase and using an adaptive β-distribution to enhance the search ability. Fig.4 is the flowchart of β-FGJO which shows the main process.

B. INITIALIZATION BASED ON FUCH CHAOTIC MAP

1) FUCH CHAOTIC MAP

Chaotic maps generate sequences with high unpredictability and randomness, effectively preventing the initial population from converging prematurely to local optima. By using chaotic maps for initialization, the population diversity is maintained, which enhances the global search capability and reduces the risk of premature convergence. Fu and Ling [38] proposed the Fuch chaotic map in 2013 to improve the optimal results generated from the existing chaotic optimization methods. The Fuch chaotic map is an adaptive iterative chaos optimization technique known for its high search effectiveness, and the results generated by the Fuch chaotic map are independent of the initial values. Fu's research pointed out that The Fuch chaotic map shows better performance compared with the Logistic chaotic map and the Tent chaotic map. Therefore, the Fuch chaotic map is employed for the initializing the prey population. Eq.(12) provides the definition of the Fuch chaotic map.

$$x_{n+1} = \cos\frac{1}{{x_n}^2}, \quad x_n \neq 0 \; and \; n \in Z^+ \tag{12}$$

In Eq.(12), x_n is an array of random numbers generated based on fuch chaotic map, the value of x_{n+1} is calculated from x_n. The initial value x_1 is typically a random number between 0 and 1.

2) SEARCH SPACE FORMULATION

The initial solution of β-FGJO is uniformly distributed over the search space with Eq.(13).

$$\mathbf{Prey}(0) = (\mathbf{lb} - \mathbf{ub}) \times \mathbf{X} + \mathbf{lb} \tag{13}$$

In Eq.(13), $\mathbf{Prey}(0)$ represents the initial matrix of prey population. $\mathbf{ub}$ and $\mathbf{lb}$ denote the upper and lower bounds for variables, and $\mathbf{X}$ is the vector of random numbers generated using the Fuchs chaotic map. The fittest individual and second fittest individual in **Prey** are deigned as jackal pair. Especially, The fittest individual is Male Jackal and the second fittest individual is Female Jackal.

C. STAGES OF GOLDEN JACKAL PAIR HUNTING

The main stages of golden jackal pair hunting are as follows:

- Searching the prey.
- Enclosing the prey.
- Pouncing towards the prey.

The stage is determined by the prey's energy levels E which is assessed using the following formula:

$$E = 1.5 \times (2 \times rand - 1) \times \left(1 - \frac{t}{MaxIt}\right) \tag{14}$$

In Eq.(14), *rand* represents a random value between 0 and 1, following a uniform distribution. t represents the current iteration, and *MaxIt* is the maximum number of iterations. The stages are controlled by p and q, which are preset to 0.15 and 0.35.

1) SEARCHING PREY

When $|E| > q$, the prey has a high energy level which can not be easily caught. Jackals perceive and follow potential targets. There are two ways to search *prey*, which are determined by the random variable *rand*. *rand* is a random value between 0 and 1, following the uniform distribution.

If $rand > 0.8$, jackals randomly search preys. Eq.(15) simulates this process:

$$\begin{cases} \mathbf{MP}(t) = (rand - E) \times \mathbf{Prey}_r(t) \\ \mathbf{FP}(t) = (rand - E) \times \mathbf{Prey}_r(t) \end{cases} \tag{15}$$

In Eq.(15), t is the current iteration. $\mathbf{MP(t)}$ and $\mathbf{FP}(t)$ are updated positions of male and female jackals. *rand* is a random value in the range 0 to 1. E is the energy calculated with Eq.(14). $\mathbf{Prey_r}(t)$ corresponds to the rth prey in the *Prey*, where r is a random number ranging from 0 to the number of prey.

If $rand \leq 0.8$, Jackals are led by the Male Jackal and Female Jackal. Eq.(16) simulates this process:

$$\begin{cases} \mathbf{MP}(t) = \mathbf{MJP} - E \times (\mathbf{MJP} - \mathbf{Prey}_r(t)) \\ \mathbf{FP}(t) = \mathbf{FJP} - E \times (\mathbf{FJP} - \mathbf{Prey}_r(t)) \end{cases} \tag{16}$$

In Eq.(16), t is the current iteration. $\mathbf{MP}(t)$ and $\mathbf{FP}(t)$ are updated positions of male and female jackals corresponding to the prey. **MJP** and **FJP** are the position of Male Jackal and Female Jackal, respectively. $\mathbf{Prey_r}(t)$ corresponds to the rth prey in the **Prey**, where r is a random number ranging from 0 to the number of prey.

2) ENCLOSING THE PREY

When $p < |E|$ and $|E| \leq q$, jackals are led by Male Jackal and Female Jackal to enclose the target prey. There are two ways to enclose *prey*, which are determined by *rand* too.

If $rand < 0.3$, the energy decreases when the prey is chased by jackals. Meanwhile, jackals enclose the prey and become hungry. In the AVOA, F is used to evaluate the hunger rate of vulture. The hunger rate of jackals here is calculated using the improved computation of F which is defined as follows:

$$F = z \times h \times \left(\sin^{\omega} a + \cos a - 1\right) + z \times (2 \times rand + 1) \times \left(1 - \frac{t}{MaxIt}\right) \tag{17}$$

In Eq.(17), F is the hunger rate, h is a random number between -2 and 2, a is calculated using Eq.(18). z is a random value between -1 and 1, ω is set to 2.5.

$$a = \frac{\pi}{2} \times \frac{t}{MaxIt} \tag{18}$$

Eq.(19) simulate the first way that jackals enclose the prey:

$$\begin{cases} \mathbf{MP}(t) = \mathbf{MJP} - \dfrac{\mathbf{MJP} \times \mathbf{Prey}(t)}{\mathbf{MJP} - \mathbf{Prey}(t)^2} \times F \\ \mathbf{FP}(t) = \mathbf{FJP} - \dfrac{\mathbf{FJP} \times \mathbf{Prey}(t)}{\mathbf{FJP} - \mathbf{Prey}(t)^2} \times F \end{cases} \tag{19}$$

In Eq.(19), t is the current iteration. $\mathbf{MP}(t)$ and $\mathbf{FP}(t)$ are updated positions of male and female jackals corresponding to the prey. $\mathbf{Prey}(t)$ is the matrix of prey population. F is the hunger rate calculated with Eq.(17).

If $rand \geq 0.3$, jackals follow the male and female jackal and randomly select prey with low energy levels. The Lévy flight strategy simulates the random chasing behavior of jackals in nature. Mantegna [39] proposed an accurate and fast algorithm for generating a Lévy stable distribution for arbitrary values. This study uses an improved version based on it, which is defined as follows:

$$LF = \frac{u \times \sigma}{|v|^{1/\beta_L}} \tag{20}$$

In Eq.(20), u and v are random values that follow a normal distribution within the range (0,1). The constant β_L is set to a default value of 1.5. σ is calculated as follows:

$$\sigma = \left(\frac{\Gamma(1+\beta_L) \times sin(\frac{\pi\beta_L}{2})}{\Gamma\left(\frac{1+\beta_L}{2}\right) \times \beta_L \times 2^{\frac{\beta_L - 1}{2}}}\right)^{\frac{1}{\beta_L}} \tag{21}$$

In Eq.(21), Γ is the standard Gamma function and β_L is a default constant set to 1.5.

Eq.(22) models this process:

$$\begin{cases} \mathbf{MP}(t) = \mathbf{MJP} - E \times \mathbf{Levy} \times (\mathbf{MJP} - \mathbf{Prey}(t)) \\ \mathbf{FP}(t) = \mathbf{FJP} - E \times \mathbf{Levy} \times (\mathbf{FJP} - \mathbf{Prey}(t)) \end{cases} \tag{22}$$

In Eq.(22), t is the current iteration. $\mathbf{MP}(t)$ and $\mathbf{FP(t)}$ are updated positions of male and female jackal corresponding to the prey. $\mathbf{Prey}(t)$ is the matrix of prey population. E is the energy level calculated with Eq.(14). **Levy** is a vector of random values obeying the Levy distribution which is define in Eq.(20).

3) POUNCING TOWARDS THE PREY

When $|E| \leq p$, the jackal pair captures the prey. After enclosing the prey, they pounce on prey and devour it. Eq.(23) simulates this behavior.

$$\begin{cases} \mathbf{MP}(t) = \mathbf{MJP} - E \times (2 \times rand \times \mathbf{MJP} - \mathbf{Prey}(t)) \\ \mathbf{FP}(t) = \mathbf{MJP} - E \times (2 \times rand \times \mathbf{FJP} - \mathbf{Prey}(t)) \end{cases} \tag{23}$$

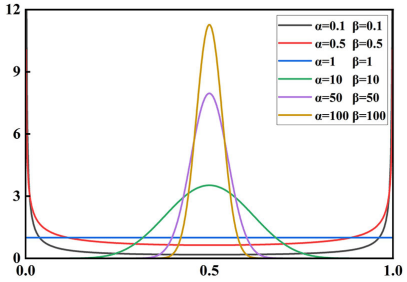

FIGURE 5. β density function with different α and β.

In Eq.(23), t is the current iteration. $\mathbf{MP(t)}$ and $FP(t)$ are updated positions of male and female jackals corresponding to the prey. $Prey\,(t)$ is the matrix of prey population. E is the energy level calculated with Eq.(14). The variable *rand* is a random value between 0 and 1, following the uniform distribution.

Finally, the prey positions are updated using Eq.(24).

$$\mathbf{Pos}(t+1) = \frac{\mathbf{MP}(t) + \mathbf{FP}\,(t)}{2} \tag{24}$$

D. ELITE STRATEGY

The elite strategy helps the algorithm focus on exploiting the best solutions discovered so far, preventing premature convergence to suboptimal solutions. By preserving the elite individuals, the algorithm ensures that highly fit individual is not lost, providing a mechanism to guide the search toward better solutions.

At the end of hunting, the fitness of all solutions in $Pos(t+1)$ is calculated and the prey positions are updated based on the elite strategy. If $\mathbf{Pos}(t+1)$ has a fitness value less than that of $\mathbf{Prey}(t)$, then $\mathbf{Pos}(t+1)$ is used as the updated solution for $\mathbf{Prey}(t+1)$. Otherwise, $\mathbf{Prey}(t+1)$ retains its original solution in $\mathbf{Prey}(t)$. Following this update, a new Male Jackal and Female Jackal are selected from the $\mathbf{Prey}(t+1)$ population.

E. ADAPTIVE SEARCH STRATEGY

1) β-DISTRIBUTION

The β-distribution [40] is versatile and finds applications in various fields, including finance, biology, engineering, and machine learning. It is a continuous probability distribution defined on the interval [0, 1] and characterized by two shape parameters commonly denoted as α and β. The probability density function of the β-distribution is given by:

$$Beta(\alpha,\beta) : prob(x|\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt} \tag{25}$$

The β-distribution is commonly used to model the distribution of random variables representing proportions or probabilities. It has a flexible shape that can be skewed left, right, or be symmetric, depending on the values of α and β. Fig.5 shows the β density function when α and β are set to different values.

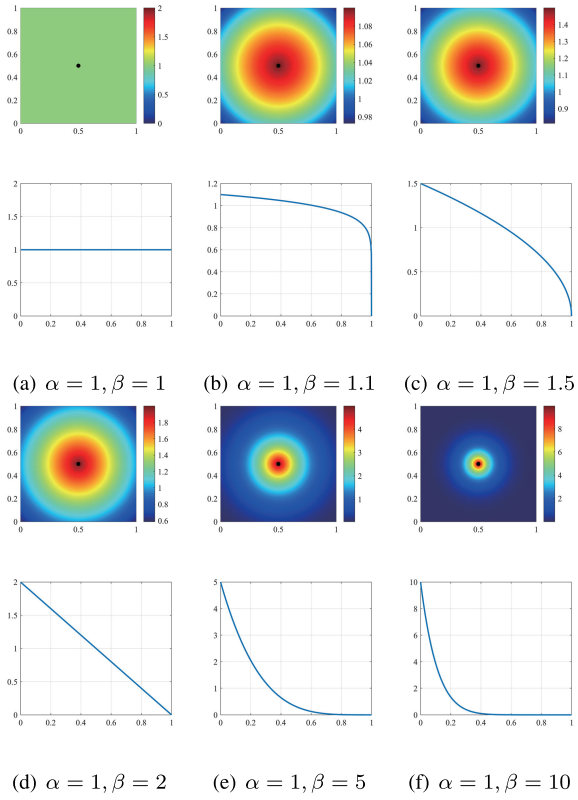

FIGURE 6. Search space and probability density distribution based on adaptive β-distribution.

2) ENHANCED SEARCH BASED ON ADAPTIVE β-DISTRIBUTION

Maintaining prey diversity during exploration and accelerating convergence during exploitation are necessary to enhance search effectiveness. This dynamic process can be effectively modeled using β-distribution. β-distribution can help individuals to a better control of their behaviors. Rokbani employs β-distributions with different shapes for exploration in the beta salp swarm algorithm for inverse kinematics and optimization, demonstrating robust search capabilities. Fig.6 shows the search space with different probability densities around the search point. It is notable that the probability density of the β-distribution varies as the parameters β change. By introducing an adaptive factor, dynamic variation in the β1-distribution can be obtained. In the early stages, it controls a broad search range, while in the later stages, the search range becomes narrower, thereby achieving a balance between global and local search. The probability density function of the adaptive β-distribution is defined as follows:

$$betarand = Beta\left(1, 1 + \frac{t}{200}\right) \tag{26}$$

In Eq.(26), t is the current iteration and $1 + \frac{t}{200}$ is the adaptive factor, α is set to 1. *betarand* is a random value generated using Eq.(25).

Therefore, the dynamic process is modeled as follow:

$$\mathbf{Pos}(t+1) = \begin{cases} (\mathbf{ub}-\mathbf{lb}) \times \mathbf{rand}+\mathbf{lb}, & rand < 0.03 \\ \mathbf{MJP} \times (F+1)+(\mathbf{Prey}(t) - MJP) \times \mathbf{BR}, & rand \geq 0.03 \end{cases} \tag{27}$$

In Eq.(27), t is the current iteration. $\mathbf{Pos}(t+1)$ is updated positions of prey and $\mathbf{Prey}(t)$ is current prey positions. **ub** and **lb** are the upper and lower bounds for variables. **BR** is the vector of random value obeying adaptive β-distribution, Each element in the **BR** is generated by computing random numbers betarand with Eq.(26). **MJP** is the position of Male Jackal and F is the hunger rate calculated using Eq.(17).

The prey position still use the elite strategy to update according to $\mathbf{Pos}(t+1)$. The pseudocode of the β-FGJO algorithm is as follow:

Algorithm 1 β-FGJO

```
Input: Population size: N, maximum number of iterations: MaxIt
Output: MJP, the fitness of MJP
Initialization the prey position Prey(0) using Eq.(13)
while t ≤ MaxIt do
    for i = 1 to N do
        Calculate E using Eq.(14), F with Eq.(17)
        if |E| > q then
            if rand > 0.8 then
                Update MP and FP with Eq.(15)
            else
                Update MP and FP with Eq.(16)
            end if
        end if
        if p < |E| and |E| ≤ q then
            if rand < 0.3 then
                Update MP and FP with Eq.(19)
            else
                Update MP and FP with Eq.(22)
            end if
        end if
        if |E| ≤ p then
            Update MP and FP with Eq.(23)
        end if
        Update Pos with Eq.(24)
    end for
    Update Prey with elite strategy, update MJP.
    for i = 1 to N do
        Update Pos with Eq.(27)
    end for
    Update Prey with elite strategy, update MJP
    t = t + 1
end while
Return MJP
```

TABLE 3. Algorithm parameters setting.

Algorithm	Parameter
β-FGJO	$p = 0.15$, $q = 0.35$, $\omega = 0.25$
GTO	$p = 0.03$, $\beta = 3$, $w = 0.8$
AVOA	$L_1 = 0.8$, $L_2 = 0.2$, $w = 2.5$, $p_1 = 0.6$, $p_2 = 0.4$, $p_3 = 0.6$
GJO	$C = 1.5$, $\beta = 1.5$
GWO	Convergence constant $a = [2, 0]$
PSO	$V_{max} = 6$, Inertia factor $w = 0.3$, $C_1 = C_2 = 2$
DE	CR = 0.9, F = 0.5
IMODE	$NP_{min} = 4$, Rate $A = 2.6$

V. SIMULATION EXPERIMENTS AND ANALYSIS

In this section, we investigated the performance of β-FGJO in addressing the inverse kinematics problem for a general multi-degree-of-freedom robotic manipulator model and the industrial manipulator PUMA560. Two experiments were designed to assess the performance and stability of single-point and multi-point solutions generated by the algorithm. The next section will provide detailed explanations of the experimental setup, parameter settings, and analysis of the comparison results.

A. EXPERIMENT AND PARAMETER SETTING

Two experiments were designed to evaluate the performance of the algorithm. Experiment 1 aims to compare the solution accuracy of all algorithms at a single point with the population size set to 50. The maximum number of iterations is set to 1000. The Wilcoxon analysis is conducted based on 20 repeated experiments, with the target point randomly selected within the workspace of the robotic arm. Experiment 2 involves tracking a set of 50 randomly generated end effector target positions. The performance and stability of each algorithm are then assessed based on the average accuracy and average time to solution, with the population size set to 50 and the maximum number of iterations set to 1000.

The algorithm is tested on both the Multi-DOF Generic Robotic Manipulator and PUMA560, compared with Artificial Gorilla Troops Optimizer (GTO), African Vulture Optimization Algorithm (AVOA), Golden Jackal Optimization (GJO), Grey Wolf Optimizer (GWO), Particle Swarm Optimization (PSO), Differential Evolution (DE) [41] and Improved Multi-operator Differential Evolution Algorithm (IMODE) [42]. Table 3 gives the parameter settings of test algorithms. The degrees of freedom of MDOF-GRM covers a range from 2 to 10. Each experiment is implemented in Matlab and executed on the computer equipped with an Intel Core i7-12700H processor and 32GB of RAM.

B. RESULT OBTAINED FOR MULTI-DOF GENERIC ROBOTIC MANIPULATOR

1) RESULT ANALYSIS OF EXPERIMENT 1

Table 5 provides the comparative results from Experiment 1, including the best, worst, average, and standard deviation

TABLE 4. Comparative results of experiment 1 for MDOF-GRM.

DOF	Indices	β-FGJO	GTO	AVOA	GJO	GWO	PSO	DE	IMODE
2	Best	1.3878E-17	1.3878E-17	8.3267E-17	3.6417E-05	1.6490E-05	1.3878E-17	1.2413E-16	6.9389E-18
	Worst	4.2367E-16	2.9675E-14	8.4751E-14	1.0216E-03	1.0118E-01	1.0118E-01	3.3171E-02	1.2738E-16
	Ave	9.2477E-17	1.6827E-15	7.0456E-15	1.4779E-04	1.5422E-02	1.5176E-02	1.6585E-03	3.4943E-17
	Std	1.0511E-16	6.5952E-15	1.9046E-14	2.3016E-04	3.6776E-02	3.7065E-02	7.4172E-03	4.1169E-17
	ρ	NaN	1.9158E-03	2.9713E-08	8.0065E-09	8.0065E-09	5.7444E-05	6.5226E-08	0.2149
	Ave. Time(s)	0.0538	0.0740	0.1233	0.2622	0.0510	0.0440	2.1449	0.9816
3	Best	5.5511E-17	5.5511E-17	7.5627E-11	3.2634E-04	1.2393E-04	3.2902E-15	7.7340E-13	1.1102E-16
	Worst	4.4409E-16	1.0069E-15	1.2790E-07	1.4802E-01	2.4754E-01	4.8667E-01	7.1363E-10	3.8459E-16
	Ave	2.0004E-16	3.7008E-16	8.3700E-09	1.1997E-02	1.2873E-02	4.9087E-02	1.2476E-10	1.7533E-16
	Std	9.9686E-17	3.0112E-16	2.8392E-08	3.4138E-02	5.5235E-02	1.2798E-01	1.7907E-10	9.9037E-17
	ρ	NaN	2.0562E-03	8.0065E-09	8.0065E-09	8.0065E-09	7.9919E-09	6.5226E-08	2.5025E-07
	Ave. Time(s)	0.0611	0.0745	0.1342	0.2698	0.0615	0.0559	2.0127	1.7938
4	Best	2.2204E-16	2.2204E-16	2.1887E-08	4.5909E-04	3.2111E-04	7.8914E-10	2.1807E-06	2.2204E-16
	Worst	5.4390E-16	6.6613E-16	1.6679E-05	5.1069E-02	6.3190E-02	3.4306E-05	2.2935E-03	6.6613E-16
	Ave	3.3883E-16	3.6754E-16	2.4963E-06	7.1667E-03	4.1219E-03	3.1310E-06	3.1521E-04	3.0119E-16
	Std	1.0414E-16	1.5243E-16	4.5859E-06	1.4268E-02	1.3922E-02	8.5181E-06	5.0405E-04	1.4588E-16
	ρ	NaN	4.2943E-03	8.0065E-09	8.0065E-09	8.0065E-09	8.0065E-09	6.5226E-08	3.3159E-08
	Ave. Time(s)	0.0619	0.0922	0.1328	0.2810	0.0681	0.0578	1.8881	3.2467
5	Best	2.2204E-16	2.2204E-16	1.5286E-12	1.0505E-03	3.2394E-04	1.6297E-09	4.8482E-04	2.2204E-16
	Worst	7.6919E-16	1.0175E-15	7.1701E-07	1.5201E-01	3.3197E-03	5.8916E-05	6.5420E-03	6.6613E-16
	Ave	3.4156E-16	3.5064E-16	8.6480E-08	1.0058E-02	1.0839E-03	5.6089E-06	2.5575E-03	3.0119E-16
	Std	1.6645E-16	2.2854E-16	2.1739E-07	3.3472E-02	8.2165E-04	1.4360E-05	1.5311E-03	1.4588E-16
	ρ	NaN	8.0359E-02	8.0065E-09	8.0065E-09	8.0065E-09	8.0065E-09	6.5226E-08	3.2247E-08
	Ave. Time(s)	0.0697	0.1068	0.1402	0.2785	0.0730	0.0631	1.7953	3.3467
6	Best	4.4409E-16	4.4409E-16	1.1259E-11	1.0886E-03	5.0437E-04	1.0458E-07	1.3627E-03	2.2204E-16
	Worst	4.4409E-16	9.9301E-16	4.7952E-05	2.2109E-01	6.0820E-03	1.5255E+00	1.3627E-03	2.2204E-16
	Ave	4.4409E-16	5.5417E-16	2.7091E-06	2.4652E-02	1.7239E-03	7.6483E-02	3.6490E-02	8.8818E-16
	Std	0.0000E+00	1.7219E-16	1.0670E-05	5.9606E-02	1.3728E-03	3.4107E-01	1.5897E-02	4.1870E-16
	ρ	NaN	8.0359E-02	8.0065E-09	8.0065E-09	8.0065E-09	8.0065E-09	6.5226E-08	5.6775E-08
	Ave. Time(s)	0.0680	0.0983	0.1358	0.2881	0.0777	0.0711	1.7625	4.5515
7	Best	1.1102E-16	1.1102E-16	6.0454E-12	1.3055E-03	6.0980E-04	5.7619E-07	1.6595E-03	4.4409E-16
	Worst	6.6613E-16	9.9920E-16	2.4690E-06	1.0662E-02	9.4331E-01	1.0424E+00	1.3843E-02	9.9301E-16
	Ave	2.4234E-16	2.6976E-16	1.3250E-07	4.2913E-03	4.9579E-02	1.5441E-01	6.2690E-03	6.5441E-16
	Std	1.9845E-16	2.3472E-16	5.5047E-07	3.1982E-03	2.1037E-01	3.7425E-01	2.8382E-03	2.4044E-16
	ρ	NaN	NaN	8.0065E-09	8.0065E-09	8.0065E-09	8.0065E-09	6.5226E-08	4.3589E-08
	Ave. Time(s)	0.0719	0.1004	0.1366	0.3166	0.0832	0.0715	1.7673	4.3446
8	Best	2.2204E-16	2.2204E-16	3.0305E-13	1.3204E-03	5.1573E-04	4.9858E-06	7.8034E-03	2.2204E-16
	Worst	8.8818E-16	1.4218E-15	3.3196E-07	1.0490E-01	1.4564E+00	1.4564E+00	4.6039E-02	1.2561E-15
	Ave	2.8675E-16	4.1788E-16	3.4030E-08	1.0964E-02	1.1668E-01	2.3849E-01	2.2320E-02	5.4526E-16
	Std	1.7383E-16	2.6057E-16	9.8408E-08	2.4037E-02	3.6519E-01	4.4042E-01	1.1456E-02	4.4232E-16
	ρ	NaN	3.4211E-01	8.0065E-09	8.0065E-09	8.0065E-09	8.0065E-09	6.5226E-08	4.5988E-08
	Ave. Time(s)	0.0750	0.1081	0.1451	0.3021	0.0901	0.0760	1.7671	4.8760
9	Best	4.4409E-16	4.4409E-16	3.7042E-11	2.3005E-03	8.4248E-04	3.1251E-04	5.8096E-03	2.2204E-16
	Worst	1.9860E-15	1.2561E-15	5.4248E-01	2.0246E-02	2.7321E+00	2.6503E+00	4.4639E-02	1.3506E-15
	Ave	8.0462E-16	6.8596E-16	2.7124E-02	5.7375E-03	1.5560E-01	4.1244E-01	2.5869E-02	4.3078E-16
	Std	4.4810E-16	2.9319E-16	1.2130E-01	4.4011E-03	6.1089E-01	6.2519E-01	1.0008E-02	3.0869E-16
	ρ	NaN	NaN	8.0065E-09	8.0065E-09	8.0065E-09	8.0065E-09	6.5226E-08	4.7161E-08
	Ave. Time(s)	0.0715	0.1049	0.1510	0.3578	0.0882	0.0789	1.6530	4.8709
10	Best	4.4409E-16	4.4409E-16	2.0395E-11	1.9917E-03	1.0955E-03	3.3992E-05	1.3654E-02	4.4409E-16
	Worst	2.9790E-15	1.3323E-15	2.9031E-07	2.6630E-01	5.3046E-03	1.2768E+00	6.8567E-02	1.2561E-15
	Std	6.9119E-16	2.6893E-16	7.2327E-08	5.8163E-02	1.3787E-03	2.8461E-01	3.2736E-02	7.4876E-16
	Ave	9.8790E-16	8.0079E-16	2.6330E-08	1.9489E-02	3.0289E-03	6.7860E-02	1.4713E-02	3.0807E-16
	ρ	NaN	NaN	8.0065E-09	8.0065E-09	8.0065E-09	8.0065E-09	6.5226E-08	5.0317E-08
	Ave. Time(s)	0.0852	0.1095	0.1550	0.3674	0.0989	0.0828	1.7760	4.8627

values computed over 20 times. Additionally, the table includes the ρ value. It is evident that β-FGJO achieves a level of accuracy similar to that of GTO and IMODE, with error reaching as low as 10^{-16}. This is corroborated by the ρ value, which confirms the similarity between β-FGJO and GTO. In contrast, AVOA, GJO, GWO, PSO and DE exhibit poor performance on the MDOF-GRM, characterized by lower accuracy and unstable results.

Fig.7 displays the average convergence curve for MDOF-GRM (2-10 DOF). When compared to the other 7 algorithms, β-FGJO demonstrates competitiveness in terms of convergence, achieving convergence within just 200 iterations.

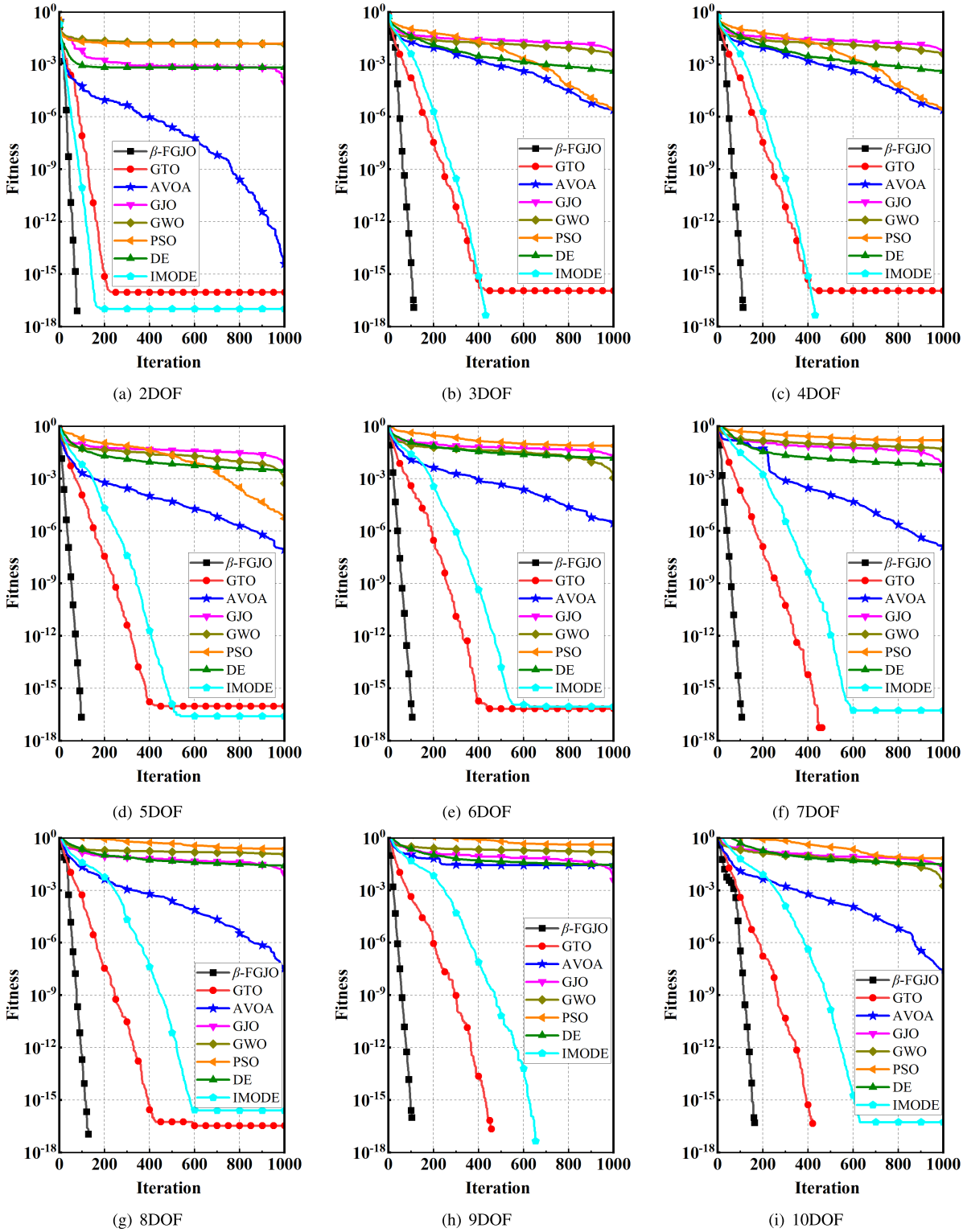

FIGURE 7. Average convergence curves of experiment 1 for MDOF-GRM.

Consequently, β-FGJO requires less time to reach the optimal value.

Fig.8 illustrates the trend of computational time. As anticipated, the average computational time required for convergence increases as the degrees of freedom range from 2 to 10. β-FGJO demonstrates an average computational time ranging from 0.05s to 0.09s within the 2 to 10 DOF range. In comparison to GTO, AVOA, GJO, DE and IMODE, β-FGJO exhibits a clear advantage in terms of computational efficiency.

TABLE 5. Comparative results of experiment 2 for MDOF-GRM.

DOF	Indices	β-FGJO	GTO	AVOA	GJO	GWO	PSO	DE	IMODE
2	Ave. Error (m)	1.4600E-16	2.6600E-16	1.0300E-14	9.0400E-04	5.9600E-03	2.1900E-03	6.6342E-04	3.0452E-17
	Std	1.6373E-16	2.3566E-16	1.7090E-14	7.8985E-04	1.3331E-02	5.3764E-03	4.6910E-03	3.6944E-17
	Ave. Time (s)	0.0595	0.0743	0.1077	0.2087	0.0420	0.0414	2.1233	0.9336
3	Ave. Error (m)	1.6700E-16	2.1500E-16	2.0100E-04	2.9474E-02	3.2936E-02	1.8600E-07	1.9128E-10	1.7528E-16
	Std	1.6756E-16	1.1722E-16	5.6143E-04	4.1987E-02	7.4557E-02	5.4063E-07	3.3526E-10	9.2981E-17
	Ave. Time (s)	0.0693	0.0759	0.1302	0.2109	0.0616	0.0549	2.0704	1.7052
4	Ave. Error (m)	1.8400E-16	4.0300E-16	1.8800E-04	1.2573E-02	1.2681E-03	2.5200E-02	3.2046E-10	3.3436E-16
	Std	1.4873E-16	7.4191E-16	8.3685E-04	3.1541E-02	1.6584E-03	7.2617E-02	6.7645E-10	1.7266E-16
	Ave. Time (s)	0.0675	0.0869	0.1305	0.2755	0.0718	0.0579	2.1736	3.1432
5	Ave. Error (m)	1.8900E-16	3.5600E-16	3.0400E-05	4.0757E-03	2.0494E-03	1.5200E-02	2.9915E-03	3.3130E-16
	Std	1.6353E-16	3.1715E-16	1.1881E-04	7.1229E-03	3.8020E-03	5.1723E-02	1.6744E-03	1.3715E-16
	Ave. Time (s)	0.0702	0.0964	0.1325	0.3172	0.0800	0.0662	2.1971	3.7254
6	Ave. Error (m)	3.1000E-16	4.2200E-16	9.3600E-03	3.4278E-03	1.6889E-03	5.2800E-02	1.4820E-02	4.4313E-16
	Std	2.1948E-16	2.7355E-16	4.1870E-02	2.2794E-03	1.0644E-03	2.3557E-01	7.3757E-03	1.4489E-16
	Ave. Time (s)	0.0666	0.1004	0.1452	0.3366	0.0769	0.0681	2.3110	4.3394
7	Ave. Error (m)	2.6400E-16	3.4100E-16	1.4200E-02	1.5841E-02	3.6519E-03	8.5200E-02	6.6228E-03	7.8536E-16
	Std	2.3833E-16	3.5296E-16	6.6615E-02	3.7027E-02	1.0402E-02	2.3481E-01	3.0571E-03	5.1588E-16
	Ave. Time (s)	0.0768	0.1120	0.1397	0.3171	0.0828	0.0746	2.2599	4.3048
8	Ave. Error (m)	2.4700E-16	3.3800E-16	1.0900E-05	1.2096E-02	1.2857E-02	1.2430E-01	2.5598E-02	4.8167E-16
	Std	1.7642E-16	2.7288E-16	7.7966E-05	7.0010E-02	7.0010E-02	2.5017E-01	1.0042E-02	4.1081E-16
	Ave. Time (s)	0.0756	0.1163	0.1578	0.3318	0.0924	0.0782	2.2482	4.6544
9	Ave. Error (m)	2.3600E-16	3.3000E-16	2.4800E-05	4.7792E-02	6.5717E-03	1.3165E-01	2.7561E-02	3.9303E-16
	Std	2.3143E-16	3.0605E-16	1.0529E-04	9.6829E-02	1.5066E-02	2.1145E-01	1.0779E-02	3.1457E-16
	Ave. Time (s)	0.0850	0.1168	0.1506	0.3604	0.0979	0.0802	2.3879	4.7940
10	Ave. Error (m)	3.2100E-16	7.0263E-16	4.4398E-05	5.4199E-02	2.1741E-03	1.3956E-01	7.3886E+02	7.8088E-16
	Std	2.3723E-16	7.8394E-16	1.7781E-04	1.6270E-01	1.2000E-03	2.5837E-01	1.3162E-02	5.9986E-16
	Ave. Time (s)	0.0903	0.1271	0.1604	0.3509	0.1025	0.0907	2.3433	4.7934

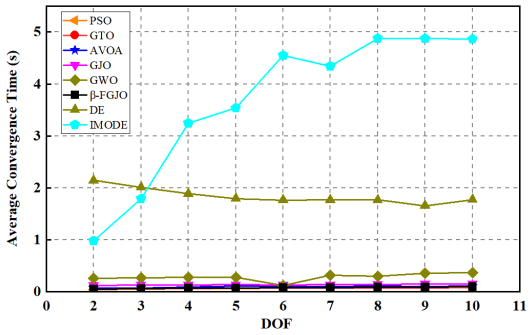

FIGURE 8. Average solution time of experiment 1 for MDOF-GRM.

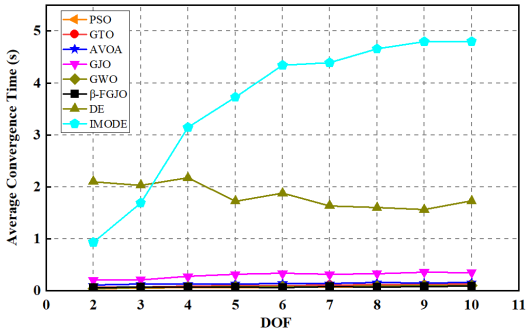

FIGURE 9. Average solution time of experiment 2 for MDOF-GRM.

2) RESULTS ANALYSIS OF EXPERIMENT 2

Table 5 presents the comparative results of Experiment 2, including the average error, standard deviation of error, average time, and iterations required for convergence. The error levels observed in Experiment 2 align with those from Experiment 1. It is apparent that β-FGJO attains an exceptionally low error level of 10^{-16}, comparable to that of GTO and IMODE, yet with superior stability. In contrast, AVOA, GJO, GWO, PSO and DE continue to exhibit poor performance with error levels around 10^{-3}.

Similarly, the average computational time required for convergence of each algorithm increases as the degrees of freedom range from 2 to 10 as is shown in Fig.9. In the case of β-FGJO, the average computational time ranges from 0.04s to 0.08s within the 2 to 10 DOF range, which is in close proximity to PSO, GWO and DE. In comparison to GTO, AVOA, GJO and IMODE, β-FGJO maintains a clear advantage in terms of computational efficiency too.

Overall, β-FGJO does well in solving the inverse kinematic problem of MDOF-GRM, which has competitive performance compared with other 7 algorithms. Compared to the best-performing GTO, β-FGJO improved time performance by 19.91%, 8.69%, 22.32%, 27.17%, 33.66%, 31.42%, 34.99%, 27.22%, and 28.95% for each degree of freedom, respectively, while maintaining the same level of accuracy.

TABLE 6. Comparative results of experiment 1 for PUMA560.

Indices	β-FGJO	GTO	AVOA	GJO	GWO	PSO	DE	IMODE
Best	1.3878E-17	1.3878E-17	1.2286E-13	1.4635E-04	6.5119E-05	2.0817E-16	3.6529E-05	1.3878E-17
Worst	7.8505E-17	6.2063E-17	3.9878E-09	4.3136E-03	3.9773E-03	5.0807E-02	7.6699E-04	2.0063E-16
Ave	2.9446E-17	2.1800E-17	3.0593E-10	1.5906E-03	6.0312E-04	5.0379E-03	2.8191E-04	3.0154E-17
Std	2.3816E-17	1.7626E-17	9.1748E-10	1.1961E-03	9.6812E-04	1.5507E-02	1.7747E-04	4.1946E-17
ρ	NaN	NaN	8.0065E-09	8.0065E-09	8.0065E-09	8.0065E-09	6.5226E-08	6.5226E-08
Ave. time (s)	0.0812	0.0938	0.1507	0.3118	0.0827	0.0694	4.1940	0.9100

TABLE 7. Best obtained solution configuration.

Method	$\theta_1(rad)$	$\theta_2(rad)$	$\theta_3(rad)$	$\theta_4(rad)$	$\theta_5(rad)$	$\theta_6(rad)$	Error(m)	Time(s)
β-FGJO	-0.1184	-1.7756	0.0104	0.4526	-0.2298	-2.5720	1.3878E-17	0.0766
GTO	0.0048	-0.2429	3.4092	-1.7863	1.3663	-2.6988	1.2490E-16	0.0916
AVOA	-0.1623	-0.1884	3.1843	0.2646	0.8539	0.2705	7.1911E-12	0.1380
GJO	-0.1330	-0.1422	3.2356	0.0066	-0.0486	-0.1287	4.3650E-04	0.3138
GWO	-2.5664	-2.9842	0.0196	0.5615	-0.5647	0.8763	5.3399E-04	0.0762
PSO	-0.1445	-0.1531	3.2899	2.9671	0.4650	4.6426	7.9928E-13	0.0644
DE	-2.0983	-2.9547	-0.3243	0.0246	1.4735	-2.9211	1.3627E-03	1.2026
IMODE	1.0435	-0.0195	-0.0545	0.2343	-1.5293	2.9384	4.4409E-16	4.4729

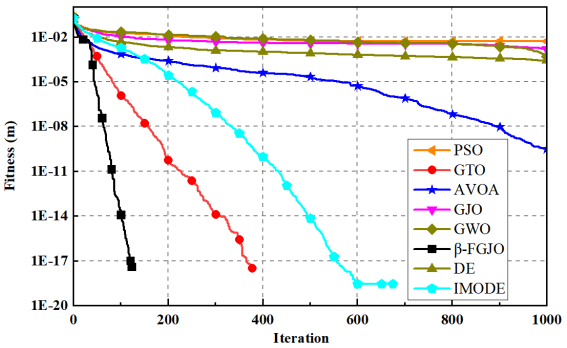

FIGURE 10. Average convergence curve of experiment 1 for PUMA560.

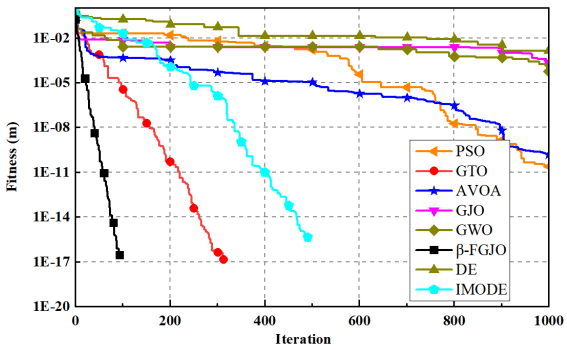

FIGURE 11. Convergence curve of best obtained solution from experiment 1 for PUMA560.

C. RESULT OBTAINED FOR PUMA560

This section conducted two experiments on PUMA560 to validate the algorithm's performance and the effectiveness of MDOF-GRM for test.

1) RESULTS ANALYSIS OF EXPERIMENT 1

The desired position is set to $P_{desired}(0.3978, 0.0981, 0.5477)$ which is randomly selected in the workspace. Comparative results of Experiment 1 are given in Table 6 which shows the best, worst, average, standard deviation value, the average time and average iteration computed over 20 times when converge.

On PUMA560, it is evident that β-FGJO has the same level of accuracy as GTO, which can reach the level of 10^{-17}. The value of ρ prove this too. AVOA reach a better level of accuracy 10^{-10}. GJO, GWO, PSO and DE still have poor performance whose level of error is around 10^{-3} too. Meanwhile, The average computational time of β-FGJO is 0.08s which is comparable to PSO and GWO. In comparisons with GTO, AVOA, GJO, DE and IMODE, β-FGJO exhibits a huge advantage on time consumption.

Fig.10 displays the average convergence curve of Experiment 1 on PUMA560. β-FGJO demonstrates high competitiveness in terms of convergence, typically converging within around 100 iterations, while GTO converges to the optimal value within 400 iterations and IMODE within 500 iterations. AVOA, GJO, GWO, PSO, and DE, on the other hand, converge at 1000 iterations.

To validate the algorithm's performance on the actual model, an optimization result is randomly selected from Experiment 1. The convergence curve is shown in Fig.11 and best obtained solution configurations are list in the Table 7. Then, different optimal configurations obtained from test algorithms are input into the objective function to measure the error. Fig.12 shows the 3D representations of selected solutions from 6 test algorithms. The actual position of end effector and actual error between the actual position and desired position are calculated and listed on it.

TABLE 8. Comparative results of experiment 2 for PUMA560.

Indices	β-FGJO	GTO	AVOA	GJO	GWO	PSO	DE	IMODE
Ave. Error(m)	2.8700E-17	3.1900E-17	4.7500E-04	1.3609E-03	2.4490E-04	1.1300E-02	2.5819E-04	2.7249E-17
Std	3.0101E-17	2.7196E-17	3.0022E-03	3.2551E-03	2.2430E-04	5.4140E-02	1.9109E-04	3.1797E-17
Ave. time(s)	0.0746	0.0989	0.1441	0.3047	0.0812	0.0671	3.886	4.3472

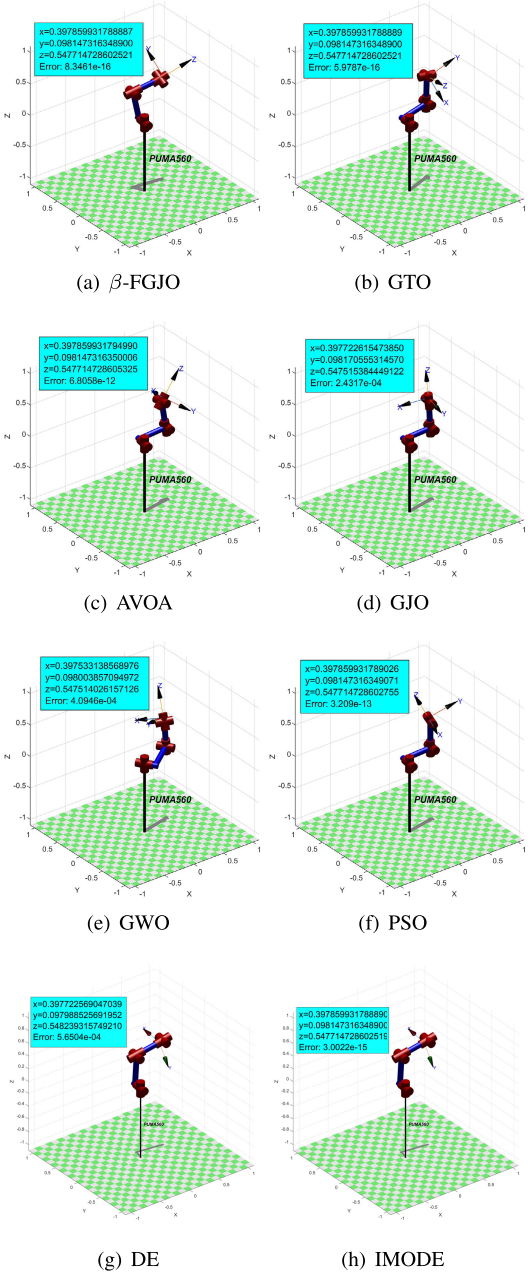

FIGURE 12. 3D representation of solution configuration.

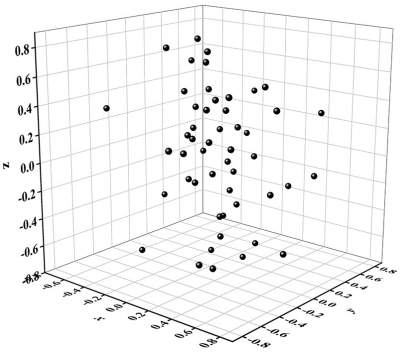

FIGURE 13. 50 points selected from workspace.

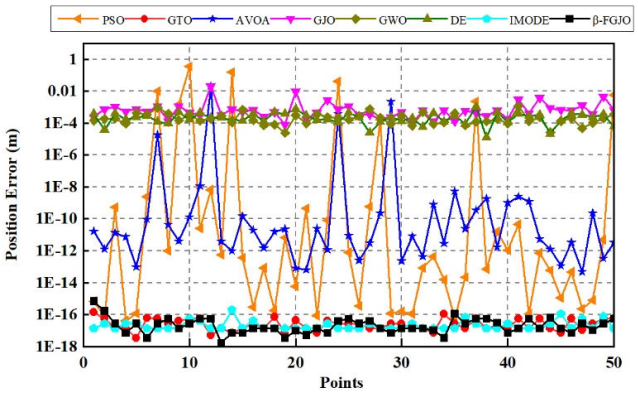

FIGURE 14. Position error obtained for randomly selected points.

The actual error for β-FGJO reaches the level of 10^{-16}, while GTO also achieves 10^{-16}. AVOA is at 10^{-12}, GJO at 10^{-4}, GWO at 10^{-4}, PSO at 10^{-13}, DE at 10^{-4}, and IMODE at 10^{-15}. Considering both the precision of variables, as determined by the significant digits and potential computational errors inherent in the calculation process, the actual error are considered consistent with the error listed in Table 7 and Table 8. The results obtained for each algorithm in the experiment are demonstrated to be accurate and effective.

2) RESULTS ANALYSIS OF EXPERIMENT 2

Fig.13 shows fifty test points randomly selected from the workspace. In Fig.14, the most stable results are β-FGJO, GTO, and IMODE. The level of error is around 10^{-17}, which is even better than the result of MDOF-GRM on 6DOF. PSO and AVOA are unstable, while GJO, GWO, and DE exhibit poor accuracy levels.

The detailed comparison results of the test methods are given in Table 8. The error of β-FGJO is 2.8800×10^{-17}. GTO has similar accuracy, with an error of 3.1900×10^{-17}, but with longer computation time. β-FGJO is more stable.

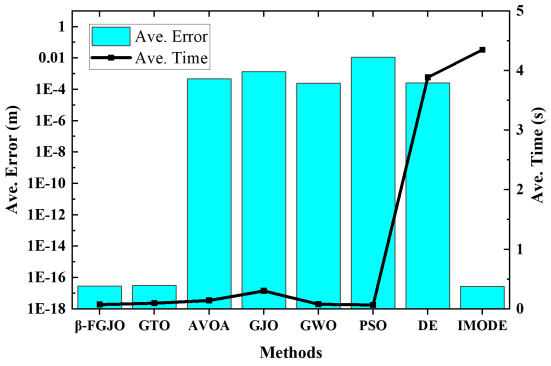

FIGURE 15. Comparison of algorithms for PUMA560.

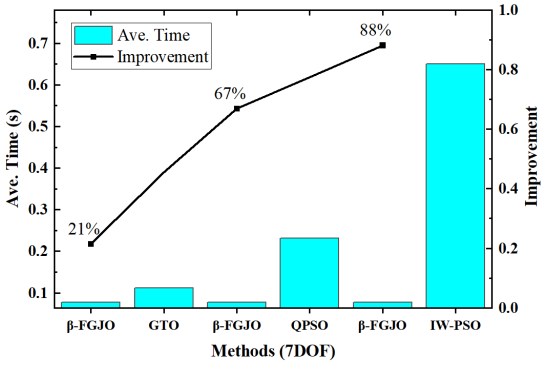

FIGURE 16. Comparison of methods proposed in the literature.

A more detailed comparison is provided in Fig.15. It is evident that the best results are obtained with β-FGJO both in terms of error and computation time. Overall, the test results on the PUMA560 are consistent with those on the MDOF-GRM with 6DOF, demonstrating the effectiveness of MDOF-GRM for evaluating optimization methods. Compared to the best-performing GTO, β-FGJO improved time performance by 24.57%.

D. ACCURACY ANALYSIS

In MATLAB, the precision of numerical calculations is typically constrained by double precision floating-point numbers which is used in this study. Double precision floating-point numbers use a 64-bit representation, with 52 bits dedicated to the mantissa, 11 bits for the exponent, and 1 bit for the sign. It means that double precision floating-point numbers can represent precision in the range of approximately 15 to 17 significant digits, which is consistent with results in Experiment 1 and 2.

VI. COMPARISON WITH PROPOSED METHODS

In this section, a comparison is made between β-FGJO and the well-performing methods proposed in the cited references. Detailed information about these selected methods are shown in Table 1 with method, average position error and average solution time. Compared to the best-performing IW-PSO and QPSO, β-FGJO improved time performance by 88.18% and 66.88%, respectively in Fig.16, while maintaining the same level of accuracy on 7DOF.

VII. CONCLUSION AND PROSPECT

Solving inverse kinematics problem is significant to the control of robotic manipulator. In this study, inverse kinematics problem is transformed into a nonlinear single objective optimization problem. Meanwhile, this paper proposes the β-FGJO algorithm to solve the problem. In β-FGJO, the Fuch chaotic map is used to initialize the prey population which can ensure uniform distribution of the population and increase randomness of the population. The enhanced search strategy based on adaptive β-distribution helps keep diversity of the population when exploration and accelerate convergence speed when exploitation. Adaptive factors in β-distribution helps control the search range. In the early stages, β-FGJO searches in a broad search range, while in the later stages, the search range becomes narrower, thereby achieving a balance between global and local search. Meanwhile, β-FGJO use new position update mechanisms based on energy and hungry level to enhance the search capability. Then β-FGJO is tested on the multi-DOF robotic manipulator with 2-10 DOF and PUMA560 for two experiments. The result verifies the excellent solution accuracy, convergence, and stability of β-FGJO.

The results of experiments show that β-FGJO can effectively solve the inverse kinematics problem in the form of optimization problem. Compared with GTO, AVOA, GJO, GWO and PSO, β-FGJO has higher coverage. the precision level of error obtained by β-FGJO in experiments is 10^{-16} and the solution time ranges from 0.05s to 0.08s. On PUMA560, the level of error obtained by β-FGJO in experiments is 10^{-17} and the solution time is around 0.07s. The β-FGJO algorithm is far superior in terms of iteration speed and convergence accuracy in dealing with inverse kinematics problem. In addition, the Wilcoxon signed-rank test of β-FGJO with the other five algorithms for has shown that the results of β-FGJO are significantly different from the other algorithms. Overall, the β-FGJO is much more efficient and robust in terms of accuracy and solution time.

The algorithm proposed in this study achieved a testing accuracy in the range of approximately 15 to 17 significant digits. Such high precision is not necessary for practical applications, and reducing the required precision can further decrease the algorithm's solution time. In this study, the testing was conducted under ideal conditions without considering various constraints present in real-world scenarios, leading to some discrepancies in practical applications. In real-world scenarios, multiple constraints need to be considered, and multi-objective solutions may also be required. This will be a focus of future research.

ACKNOWLEDGMENT

The authors declare no competing interests that are directly or indirectly related to the work submitted for publication.

REFERENCES

[1] Z. Li, S. Li, and X. Luo, "An overview of calibration technology of industrial robots," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 1, pp. 23–36, Jan. 2021.

[2] N. A. Meisel, N. Watson, S. G. Bilén, J. P. Duarte, and S. Nazarian, "Design and system considerations for construction-scale concrete additive manufacturing in remote environments via robotic arm deposition," *3D Printing Additive Manuf.*, vol. 9, no. 1, pp. 35–45, Feb. 2022.

[3] S. Yang, W. Zhang, Y. Zhang, H. Wen, and D. Jin, "Development and evaluation of a space robot prototype equipped with a cable-driven manipulator," *Acta Astronautica*, vol. 208, pp. 142–154, Jul. 2023.

[4] K. H. Sheetz, J. Claflin, and J. B. Dimick, "Trends in the adoption of robotic surgery for common surgical procedures," *JAMA Netw. Open*, vol. 3, no. 1, Jan. 2020, Art. no. e1918911.

[5] S. Akram and Q. U. Ann, "Newton Raphson method," *Int. J. Sci. Eng. Res.*, vol. 6, no. 7, pp. 1748–1752, 2015.

[6] C. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, no. 1, pp. 93–101, Jan. 1986.

[7] S. R. Buss, "Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods," *IEEE J. Robot. Autom.*, vol. 17, no. 16, pp. 1–19, Apr. 2004.

[8] J. Li, H. Yu, N. Shen, Z. Zhong, Y. Lu, and J. Fan, "A novel inverse kinematics method for 6-DOF robots with non-spherical wrist," *Mechanism Mach. Theory*, vol. 157, Mar. 2021, Art. no. 104180.

[9] I. Dulęba and M. Opałka, "A comparison of Jacobian-based methods of inverse kinematics for serial robot manipulators," *Int. J. Appl. Math. Comput. Sci.*, vol. 23, no. 2, pp. 373–382, Jun. 2013.

[10] E. Yazan and M. F. Talu, "Comparison of the stochastic gradient descent based optimization techniques," in *Proc. Int. Artif. Intell. Data Process. Symp. (IDAP)*, Sep. 2017, pp. 1–5.

[11] J. Iannacci, G. Tagliapietra, and A. Bucciarelli, "Exploitation of response surface method for the optimization of RF-MEMS reconfigurable devices in view of future beyond-5G, 6G and super-IoT applications," *Sci. Rep.*, vol. 12, no. 1, p. 3543, Mar. 2022.

[12] G. Nannicini, "Fast quantum subroutines for the simplex method," *Oper. Res.*, vol. 72, no. 2, pp. 763–780, Mar. 2024.

[13] V. Lai, Y. F. Huang, C. H. Koo, A. N. Ahmed, and A. El-Shafie, "A review of reservoir operation optimisations: From traditional models to metaheuristic algorithms," *Arch. Comput. Methods Eng.*, vol. 29, no. 5, pp. 3435–3457, Aug. 2022.

[14] R. E. J. Kennedy, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw.*, Oct. 1995, pp. 1942–1948.

[15] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.

[16] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[17] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016.

[18] B. Abdollahzadeh, F. S. Gharehchopogh, and S. Mirjalili, "Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems," *Int. J. Intell. Syst.*, vol. 36, no. 10, pp. 5887–5958, Oct. 2021.

[19] B. Abdollahzadeh, F. S. Gharehchopogh, and S. Mirjalili, "African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems," *Comput. Ind. Eng.*, vol. 158, Aug. 2021, Art. no. 107408.

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

[21] S. V. Reyes and S. P. Gardini, "Inverse kinematics of manipulator robot using a PSO metaheuristic with adaptively exploration," in *Proc. IEEE 26th Int. Conf. Electron., Electr. Eng. Comput. (INTERCON)*, Aug. 2019, pp. 1–4.

[22] R. V. Ram, P. M. Pathak, and S. J. Junco, "Inverse kinematics of mobile manipulator using bidirectional particle swarm optimization by manipulator decoupling," *Mechanism Mach. Theory*, vol. 131, pp. 385–405, Jan. 2019.

[23] H. Deng and C. Xie, "An improved particle swarm optimization algorithm for inverse kinematics solution of multi-DOF serial robotic manipulators," *Soft Comput.*, vol. 25, no. 21, pp. 13695–13708, Nov. 2021.

[24] S. Dereli and R. Köker, "IW-PSO approach to the inverse kinematics problem solution of a 7-DOF serial robot manipulator," *Sigma J. Eng. Natural Sci.*, vol. 36, no. 1, pp. 77–85, 2018.

[25] S. Dereli and R. Köker, "A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: Quantum behaved particle swarm algorithm," *Artif. Intell. Rev.*, vol. 53, no. 2, pp. 949–964, Feb. 2020.

[26] M. Ayyıldız and K. Çetinkaya, "Comparison of four different heuristic optimization algorithms for the inverse kinematics solution of a real 4-DOF serial robot manipulator," *Neural Comput. Appl.*, vol. 27, no. 4, pp. 825–836, May 2016.

[27] S. Dereli and R. Köker, "Calculation of the inverse kinematics solution of the 7-DOF redundant robot manipulator by the firefly algorithm and statistical analysis of the results in terms of speed and accuracy," *Inverse Problems Sci. Eng.*, vol. 28, no. 5, pp. 601–613, May 2020.

[28] S. Dereli and R. Köker, "Simulation based calculation of the inverse kinematics solution of 7-DOF robot manipulator using artificial bee colony algorithm," *Social Netw. Appl. Sci.*, vol. 2, no. 1, pp. 1–11, Jan. 2020.

[29] S. Dereli and R. Köker, "Strengthening the PSO algorithm with a new technique inspired by the golf game and solving the complex engineering problem," *Complex Intell. Syst.*, vol. 7, no. 3, pp. 1515–1526, Jun. 2021.

[30] T. Çavdar, M. Mohammad, and R. A. Milani, "A new heuristic approach for inverse kinematics of robot arms," *Adv. Sci. Lett.*, vol. 19, no. 1, pp. 329–333, Jan. 2013.

[31] Z. Sui, L. Jiang, Y.-T. Tian, and W. Jiang, "Genetic algorithm for solving the inverse kinematics problem for general 6R robots," in *Proc. Chin. Intell. Autom. Conf., Intell. Technol. Syst.*, Z. Deng and H. Li, Eds., Berlin, Germany: Springer, Mar. 2015, pp. 151–161.

[32] R. Köker and T. Çakar, "A neuro-genetic-simulated annealing approach to the inverse kinematics solution of robots: A simulation based study," *Eng. Comput.*, vol. 32, no. 4, pp. 553–565, Oct. 2016.

[33] N. Rokbani, S. Mirjalili, M. Slim, and A. M. Alimi, "A beta salp swarm algorithm meta-heuristic for inverse kinematics and optimization," *Int. J. Speech Technol.*, vol. 52, no. 9, pp. 10493–10518, Jul. 2022.

[34] 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," *Frontiers Bioengineering Biotechnol.*, vol. 10, May 2022, Art. no. 832829.

[35] B. Armstrong, O. Khatib, and J. Burdick, "The explicit dynamic model and inertial parameters of the Puma 560 arm," in *Proc. IEEE Int. Conf. Robot. Autom.*, San Francisco, CA, USA, Apr. 1986, pp. 510–518.

[36] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *J. Appl. Mech.*, vol. 22, no. 2, pp. 215–221, Jun. 1955.

[37] N. Chopra and M. M. Ansari, "Golden jackal optimization: A novel nature-inspired optimizer for engineering applications," *Expert Syst. Appl.*, vol. 198, Jul. 2022, Art. no. 116924.

[38] W. Fu and C. Ling, "An adaptive iterative chaos optimization method," *J. Xi'an Jiaotong Univ.*, vol. 47, no. 2, pp. 33–38, 2013.

[39] R. N. Mantegna, "Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 49, no. 5, p. 4677, 1994.

[40] N. L. Johnson, S. Kotz, and N. Balakrishnan, "Beta distributions," in *Continuous Univariant Distributions*, vol. 2. New York, NY, USA: Wiley, 1994, pp. 210–275.

[41] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, vol. 11, pp. 341–359, Dec. 1997.

[42] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Improved multi-operator differential evolution algorithm for solving unconstrained problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2020, pp. 1–8.

XIAO-YU ZHANG received the B.S. degree in Internet of Things engineering from Nantong University, Nantong, China, in 2019. He is currently pursuing the master's degree in mechanical engineering with Yancheng Institute of Technology. His current research interests include evolutionary computation, robotic manipulators, artificial intelligence, and fault diagnosis.

ZHONG-QING FANG received the B.S. degree in optical information science and technology from Anhui University, in 2014, and the Ph.D. degree in optics from the University of Science and Technology of China (USTC), in 2019. His main research interest includes artificial intelligence algorithms.

JIA-PAN LI received the degree from the College of Information and Communication Engineering, Harbin Engineering University. His current research interests include signal processing and optimum algorithms.

WEI-BIN KONG received the B.S. degree in mathematics from Qufu Normal University, China, in 2007, and the M.S. degree in mathematics and the Ph.D. degree in radio engineering from Southeast University, Nanjing, China, in 2010 and 2015, respectively. Since 2016, he has been an Associate Professor with the College of Information Engineering, Yancheng Institute of Technology, Yancheng. His current research interests include computational electromagnetics, artificial intelligence, and wireless communication.

YI DU received the B.S. degree in electrical information engineering from Suqian University, Suqian, China, in 2022. He is currently pursuing the master's degree in mechanical engineering with Yancheng Institute of Technology. His research interests include fault diagnosis and signal processing.

TING-LIN ZHANG received the B.Sc. degree in automation from Shandong University, in 2009, and the Ph.D. degree in control science and engineering from Zhejiang University, in 2017. She is currently a Lecturer with Yancheng Institute of Technology. Her research interests include artificial intelligence and biomedical signal processing.

RU-GANG WANG received the B.S. degree from Wuhan University of Technology, Wuhan, China, in 1999, the M.S. degree from Jinan University, Guangzhou, China, in 2007, and the Ph.D. degree from Nanjing University, Nanjing, China, in 2012. He is currently a Professor with the College of Information Engineering, Yancheng Institute of Technology, Yancheng, China. His research interests include image processing technology and optical communication networks.