Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 20(2): 2776–2792.
DOI: 10.3934/mbe.2023130
Received: 21 September 2022
Revised: 22 November 2022
Accepted: 23 November 2022
Published: 30 November 2022

Research article

Robotic arm trajectory optimization based on multiverse algorithm

Junjie Liu[1,*], Hui Wang[1], Xue Li[2], Kai Chen[1] and Chaoyu Li[1]

[1] School of Mechanical and Electronic Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

[2] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

* **Correspondence:** Email: liujunjie@nuaa.edu.cn; Tel: +8617860730750.

Abstract: For inefficient trajectory planning of six-degree-of-freedom industrial manipulators, a trajectory planning algorithm based on an improved multiverse algorithm (IMVO) for time, energy, and impact optimization are proposed. The multi-universe algorithm has better robustness and convergence accuracy in solving single-objective constrained optimization problems than other algorithms. In contrast, it has the disadvantage of slow convergence and quickly falls into local optimum. This paper proposes a method to improve the wormhole probability curve, adaptive parameter adjustment, and population mutation fusion to improve the convergence speed and global search capability. In this paper, we modify MVO for multi-objective optimization to derive the Pareto solution set. We then construct the objective function by a weighted approach and optimize it using IMVO. The results show that the algorithm improves the timeliness of the six-degree-of-freedom manipulator trajectory operation within a specific constraint and improves the optimal time, energy consumption, and impact problems in the manipulator trajectory planning.

Keywords: trajectory planning; optimization algorithm; optimal time; optimal energy consumption; optimal impact

1. Introduction

Recently, with the continuous development of robotic arm technology, the requirements for its working environment are becoming increasingly stringent. Therefore, the performance requirements for robotic arms are also getting higher and higher. Under different working conditions, the operating time, speed, acceleration, jerk, and other performance requirements of the manipulator are different. Optimizing the trajectory of the robotic arm has always been the goal of designers.

Trajectory optimization can be generally classified into single-objective optimization, and multi-

objective optimization [1] according to the optimization objective. Single-objective optimization usually optimizes only one objective without considering other performance metrics. For example, in a trajectory optimization problem concerning time, increasing the speed to reduce the running time also increases the impact on the robot arm. Multi-objective optimization needs to consider two or more performance optimalities simultaneously, obtain the corresponding Pareto solution set, and then select the optimal solution based on the working conditions.

Rabab Benotsmane et al. [2] elaborated a new "whip-lashing" method that aims to realize an optimized trajectory for the five-degree-of-freedom RV-2AJ robot arm, and the method lowered energy consumption and increased productivity. Han et al. [3] proposed a particle swarm optimization (PSO) algorithm that can dynamically adjust the learning factor. This method effectively combines the piecewise polynomial interpolation function with PSO, avoids the complex process of particle swarm algorithm to construct the adaptation function, and improves the problem that the traditional PSO is more probably to fall into local extreme value in the early stage and convergence speed is slow in the later stage. Zhao et al. presented an efficient numerical method for trajectory optimization for motion planning involving complicated robot dynamics. Peng et al. [4] employed the non-dominated neighborhood immune genetic algorithm, obtaining each joint's optimal position, velocity, acceleration, and jerk planning curves. Kei Ota et al. [5] proposed a reinforcement learning-based algorithm for trajectory optimization for constrained dynamical systems. Shi et al. [6] proposed a multi-objective genetic algorithm (NSGA-II) and multi-objective particle swarm optimization algorithm (MOPSO) in order to settle trajectory planning with kinematic and dynamic constraints.

In this study, a 3-5-3 spline polynomial and multiverse optimization algorithm are used to plan the trajectory of the manipulator in the joint space. Nevertheless, the multiverse optimization algorithm performs poorly in solving large-scale optimization problems and lacks jumping out of local poles [7]. The ability of value makes it impossible to find the optimal global solution. This paper proposes a multi-objective optimization based on MVO and a method to improve the wormhole probability curve, aiming to improve the convergence speed and enhance the global search ability through population mutation. The final result confirms the method's feasibility, improves the manipulator's efficiency and reduces energy consumption.

2. Trajectory planning

2.1. Construction of joint trajectory based on 3-5-3 spline polynomial

The parameters, including speed, acceleration, and jerk of the manipulator, have better real-time performance in joint space, do not need to consider particular problems, and are more convenient to control and describe. As a result, trajectory planning is generally carried out in joint space. In order to solve the relationship between each joint variable and time, it is necessary to discretize the task trajectory performed by the manipulator first and then perform inverse kinematics analysis. In the trajectory planning of the manipulator, the 3rd-degree polynomial interpolation can only ensure that the velocity and displacement are continuous, and the acceleration may jump, which is not necessarily continuous. Moreover, the discontinuous acceleration will make the joints of the robotic arm shake and impact, which will influence the working efficiency and service life of the robotic arm [8]. Compared with the 3rd-degree polynomial, the 5th-degree polynomial interpolation can ensure continuous and smooth curves of joint displacement, velocity, and acceleration, even though the calculation amount

is increased [9]. Generally, the higher the order of the polynomial function, the more constraints it can satisfy. Therefore, a 5th-degree polynomial is used to fit each joint position sequence node, and a continuous and smooth trajectory of displacement, velocity, and acceleration can thus be obtained [10, 11].

Assuming that the robot's starting point, ending point, and two intermediate path points are known, each joint angle of each joint at the four interpolation points can be obtained by solving the inverse kinematics equation. The interpolation angle of joint i is denoted by θ_{ji}, where i represents the number of joints ($i = 1, 2, 3, 4, 5, 6$), and j denotes the serial number of the interpolation point ($j = 1, 2, 3, 4$). The general formula of the 3-5-3 spline polynomial for the i joint is:

$$\begin{cases} l_{j1}(t) = a_{j13}t^3 + a_{j12}t^2 + a_{j11}t + a_{j10} \\ l_{j2}(t) = a_{j25}t^5 + a_{j24}t^4 + a_{j23}t^3 + a_{j22}t^2 + a_{j21}t + a_{j20} \\ l_{j3}(t) = a_{j33}t^3 + a_{j32}t^2 + a_{j31}t + a_{j30} \end{cases} \tag{2.1}$$

where $l_{j1}(t), l_{j2}(t)$ and $l_{j3}(t)$ represent the trajectory of the 3-5-3 spline polynomial, respectively. According to the constraints of the multi-degree polynomial, the polynomial is solved. The unknown coefficient a can be obtained. The initial point of each segment of the joint, the displacement of the two path points and the termination point, the continuous displacement, velocity, and acceleration between the path points, as well as the velocity (usually 0) and acceleration of the initial point and the termination point, are known.

2.2. Solving polynomial interpolated trajectories

The matrix $\boldsymbol{A}$ can be listed according to the constraints, including Eq (2.2). It can be seen from the matrix expression that the constraints and constraints are only related to the time t.The value of the coefficient a can be obtained from Relational Eq (2.5). t_1, t_2, t_3 in Eq (2.2) respectively represent the time of the 3-segment polynomial interpolation of the i joint, respectively.

$$\boldsymbol{A} = \begin{pmatrix} t_1^3 & t_1^2 & t_1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 3t_1^2 & 2t_1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 6t_1 & 2 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & t_2^5 & t_2^4 & t_2^3 & t_2^2 & t_2 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 5t_2^4 & 4t_2^3 & 3t_2^2 & 2t_2 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 20t_2^3 & 12t_2^2 & 6t_2 & 2 & 0 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_3^3 & t_3^2 & t_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3t_3^2 & 2t_3 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6t_3 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.2}$$

$$\boldsymbol{\theta} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & x_{j3} & 0 & 0 & x_{j0} & 0 & 0 & x_{j2} & x_{j1} \end{bmatrix}^{\mathrm{T}} \tag{2.3}$$

$$\boldsymbol{a} = \boldsymbol{A}^{-1}\boldsymbol{\theta} = \begin{bmatrix} a_{j13} & a_{j12} & a_{j11} & a_{j10} & a_{j25} & a_{j24} & a_{j23} & a_{j22} & a_{j21} & a_{j21} & a_{j33} & a_{j32} & a_{j31} & a_{j30} \end{bmatrix}^{\mathrm{T}} \tag{2.4}$$

2.3. Establishment of objective function and kinematic constraints

Under the premise of considering various constraints in kinematics to improve the working efficiency of the manipulator, reduce energy consumption, reduce the impact, and achieve the multi-objective comprehensive optimization of time, energy consumption, and impact, the optimization objective function is defined as:

$$\begin{cases} S_1 = T = \sum_{j=0}^{n-1} \left(t_{j+1} - t_j\right) \\ S_2 = \sum_{i=1}^{N} \sqrt{\frac{1}{T}\int_0^T a_i^2 \, \mathrm{d}t} \\ S_3 = \sum_{i=1}^{N} \sqrt{\frac{1}{T}\int_0^T J_i^2 \, \mathrm{d}t} \end{cases} \tag{2.5}$$

In the formula, S_1 is the total running time of the manipulator, the sum of the time intervals between the position sequence nodes, and the work efficiency index of the manipulator. S_2 is the average acceleration of the joint, which is the energy consumption index. S_3 is the average jerk in the joint motion and the pulsating impact index. T is the total running time of the manipulator. a_i is the acceleration of the joint, and J_i is the jerk of the joint. The manipulator must be subject to constraints such as position, velocity, acceleration, and jerk during operation. t_j is the time of the jth point, and N is the total number of joints of the manipulator. Besides, the constraints are defined as

$$\begin{cases} g_1(t) = |q_i(t)| - Q_{\mathrm{jmax}} \\ g_2(t) = |v_i(t)| - V_{\mathrm{jmax}} \\ g_3(t) = |a_i(t)| - A_{\mathrm{jmax}} \\ g_4(t) = |j_i(t)| - J_{j\,\mathrm{max}} \end{cases} \tag{2.6}$$

wherein, $g_1(t)$ is the joint displacement constraint; $g_2(t)$ represents the joint velocity constraint; $g_3(t)$ indicates joint acceleration constraints; $g_4(t)$ suggests joint acceleration constraints; Q_{jmax} is the maximum displacement of the joint, $q_i(t)$ is the angle of the ith joint; V_{jmax} is the maximum speed of the joint, $v_i(t)$ is the speed of the ith joint; A_{jmax} is the maximum acceleration of the joint, $a_i(t)$ is the acceleration of the ith joint; and J_{jmax} indicates the maximum jerk of the joint, $j_i(t)$ is the jerk of the ith joint.

2.4. Robotic arm trajectory optimization based on multiverse algorithm

2.4.1. Multiverse algorithm

The main idea of the multiverse optimization algorithm (MVO) is to build a model based on the three main concepts of the multiverse theory, namely, white holes, black holes, and wormholes. A white hole is a celestial body that only emits but does not absorb. A black hole absorbs everything in

the universe. A wormhole is like a space-time tunnel connecting a white hole and a black hole, sending individuals to any corner of the universe, even from one universe to another. The multiverse reaches a stable state through the three. The candidate solution is defined as the universe, and the fitness of the candidate solution is the universe's expansion rate. Additionally, the MVO algorithm can be divided into the following steps:

1) Set up a group consisting of N universes to search in the D-dimensional target space, and initialize it.

$$\boldsymbol{X} = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^d \\ x_2^1 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \ddots & \vdots \\ x_n^1 & x_n^2 & \cdots & x_n^d \end{bmatrix} \tag{2.7}$$

2) To establish a mathematical model between white and black holes and exchange the objects of the universe, the roulette mechanism is adopted. In each iteration, the universe is sorted according to the expansion rate (fitness) of the universe. Besides, a roulette wheel is used to select one White hole.

$$x_i^j = \begin{cases} x_k^j, & r_1 < \mathrm{NI}(X_i) \\ x_i^j, & r_1 \geq \mathrm{NI}(X_i) \end{cases} \tag{2.8}$$

where x_i^j is the j argument for the i universe, $\mathrm{NI}(X_i)$ shows the normalized expansion rate of the i universe, r_1 is a random number between [0, 1], and x_k^j is the j parameter of the k universe selected by the roulette mechanism.

3) The wormhole existence probability WEP increases linearly, the travel distance rate TDR decreases continuously in the iterative process. Thus, a more accurate local search can be performed within the obtained global optimum range.The adaptive formulas of two coefficients are Eqs (2.9) and (2.10):

$$WEP = WEP_{\min} + l \times \left(\frac{WEP_{\max} - WEP_{\min}}{L} \right) \tag{2.9}$$

$$TDR = 1 - \frac{l^{1/p}}{L^{1/p}} \tag{2.10}$$

where $WEP_{\min}$ is the minimum value of WEP, $WEP_{\max}$ is the largest value of WEP, l is the current number of iterations, L is the maximum number of iterations, and p defines the development precision during iteration. It indicates the higher the value of p, the faster the local search.

4) Update the cosmic position and find the optimal individual.

When $r_2 < WEP$,

$$x_i^j = \begin{cases} x_j + TDR \times \left(\left(\mathrm{ub}_j - \mathrm{lb}_j \right) \times r_4 + \mathrm{lb}_j \right), & r_3 < 0.5 \\ x_j - TDR \times \left(\left(\mathrm{ub}_j - \mathrm{lb}_j \right) \times r_4 + \mathrm{lb}_j \right), & r_3 \geq 0.5 \end{cases} \tag{2.11}$$

When $r_2 \geq WEP$,

$$x_i^j = x_i^j \tag{2.12}$$

where x_j represents the j parameter of the best universe currently formed, ub_j and lb_j represents the upper and lower bounds of the j variable, and r_2, r_3 and r_4 suggest random numbers between [0, 1].

2.4.2. Multiverse algorithm improvements

Wormhole existence probability WEP increases linearly in the iterative process. In order to speed up the convergence speed of the algorithm, WEP is currently improved on the basis of traditional MVO, and its expression is:

$$\alpha_t = \alpha_0 \exp(-\beta \times t) \tag{2.13}$$

where α_0 is the initialization learning rate, the learning rate at the t iteration is α_t, and β is the attenuation rate.

$$WEP = WEP_{\min} + (1 - \alpha_t)(WEP_{max} - WEP_{\min}) \tag{2.14}$$

MVO has great advantages in local search, but it is not difficult to fall into local optimum, resulting in insufficient accuracy, whereas the genetic algorithm has strong global search ability. In order to overcome the above shortcomings, the population mutation strategy is currently used to enhance the diversity of the population and avoid falling into local optimum. Improvement strategies are described below:

$$x_i = [x_i^1, x_i^2, \cdots, x_i^k, \cdots, x_i^d], r_4 < \text{Mut}(X_i) \tag{2.15}$$

$$x_i^k = \text{lb}_k + rand \times (\text{ub}_k - \text{lb}_k) \tag{2.16}$$

where $\text{Mut}(X_i)$ is the rate of variability of the i universe, r_4 is a random number between $[0, 1]$, and k refers to a random integer between $[1, d]$.

In this paper, an adaptive adjustment strategy for parameter p is proposed. The specific steps of the strategy are as follows:

1) Calculate the average value of the normalized expansion rate f_m.

2) Take out the universes whose expansion rate is greater than the average expansion rate, and calculate their average expansion rate f_{m1}. Assign a universe with an expansion rate greater than f_{m1} to a maximum p.

3) Take out the universes whose expansion rate is less than the average expansion rate, and calculate their average expansion rate f_{m2}. Particles with an expansion rate less than f_{m2} are assigned the smallest p.

4) According to the adaptive adjustment strategy, the particles whose fitness value is between f_{m1} and f_{m2} are given values that vary linearly between the maximum and minimum p of f_{m1} and f_{m2}.

When there are multiple objectives, due to the conflict and incomparability between objectives, a solution may be the best on one objective and the worst on other objectives. These solutions, which, while improving any objective function, necessarily weaken at least one other objective function, are called non-dominated solutions or Pareto solutions.

The comparison in the single objective only compares the value of the single objective function. In the multi-objective, if all the objective function values of the A solution are better than the B solution, then A is then said to dominate B. If A solution cannot satisfy all target values that are better than B, and no target values of B are better than A, then A and B belong to the same level.

1) First, generate the population, obtain the first dominant solution, and use the roulette wheel to select the optimal universe in the first dominant solution.

2) Update the universe according to the original MVO algorithm, the expansion rate belonging to the first dominant solution takes the smaller value, and the expansion rate which does not belong to the feasible solution takes 1.

3) The newly obtained universe is merged with the parent population, and the first dominant solution is updated. Additionally, redundant and inferior solutions are deleted.

2.4.3. Test function and validation

After testing the ZDT1 function and ZDT2 function, the effect is shown in Figures 1 and 2.

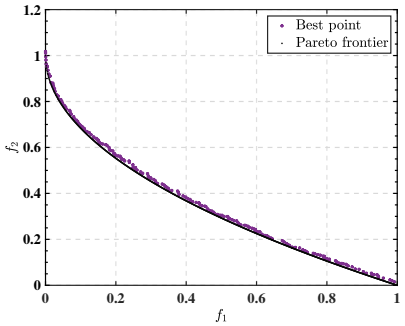


Figure 1. ZDT1 function effect.

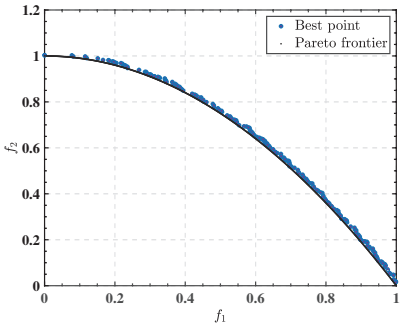


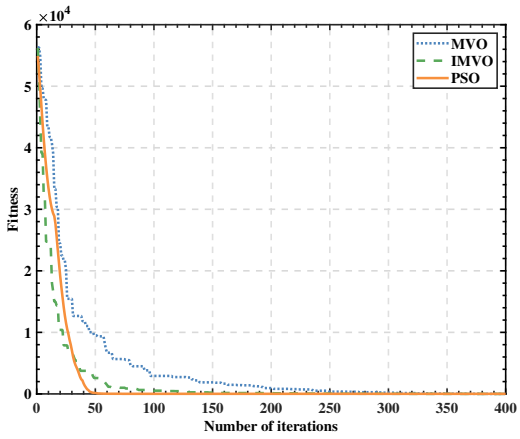
Figure 2. ZDT2 function effect.

Table 1. Benchmark functions used in the study.

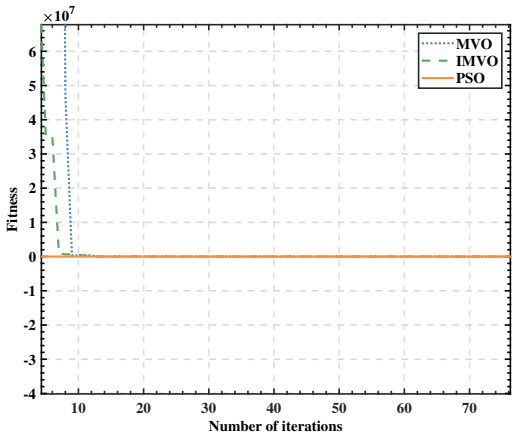
Benchmark function	Range	$f_{\min}$
$f_1(x) = \sum_{i=1}^{n} x_i^2$	[-100,100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10,10]	0
$f_3(x) = \max_{i=1}^{n} \{\lvert x_i \rvert\}$	[-100,100]	0
$f_4(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	[-100,100]	0
$f_5(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	[-5.12,5.12]	0
$f_6(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\frac{x_i}{\sqrt{i}} + 1$	[-600,600]	0
$f_7(x) = 1 - \cos\left(2\pi\sqrt{\sum_{i=1}^{n} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2}$	[-32,32]	0
$f_8(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	[-100,100]	0
$f_9(x) = \sum_{i=1}^{n} i x_i^4 + \text{rand}$	[-1.28,1.28]	0
$f_{10}(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	[-30,30]	0

Ten classical benchmark functions, as shown in Table 1, are used to evaluate the performance of MVO, PSO and IMVO are the algorithms for comparison. In the ten functions, $f_1 \sim f_4$ are unimodal test functions that can test the optimization accuracy of the algorithms, $f_5 \sim f_8$ are multimodal test functions that can test the global optimization ability and convergence speed of the algorithms, $f_9 \sim f_{10}$ is ill-conditioned test functions that can test the exploration and exploitation capabilities of the algorithms.

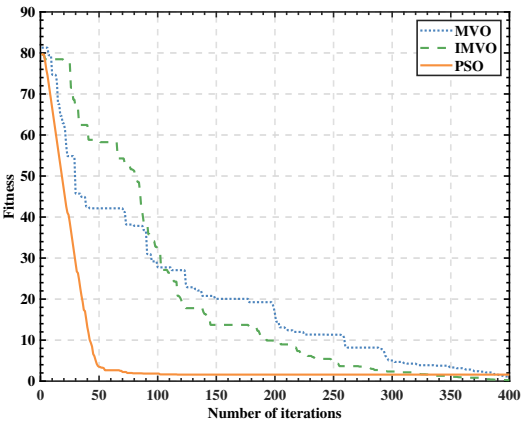
In Figure 4, it can be seen that the optimization speed and accuracy of IMVO added with the parameter adaptive adjustment and wormhole probability curve are significantly improved compared with MVO, which shows that improving the population diversity and the ability to jump out of the local solutions is incredibly beneficial. $f_1 \sim f_4$ prove that the optimization accuracy of IMVO is not weaker than that of PSO, and the convergence speed is faster than that of MVO. $f_5 \sim f_8$ prove that the global optimization ability and convergence speed of IMVO is higher than that of MVO and PSO. $f_9 \sim f_{10}$ prove that the exploration capabilities of IMVO can be maintained at a high level.



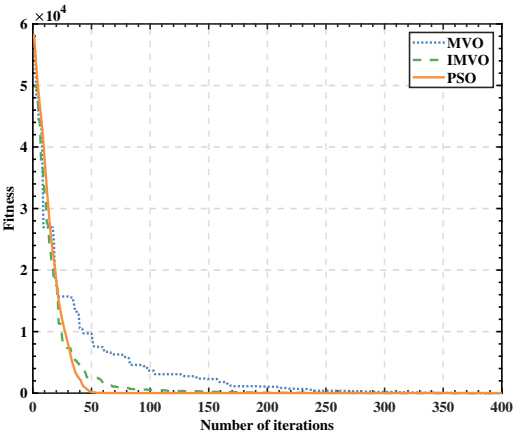
(a) Performance comparison on the f_1 function



(b) Performance comparison on the f_2 function



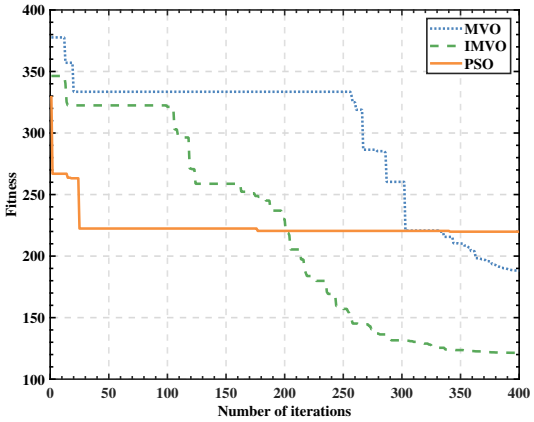
(c) Performance comparison on the f_3 function



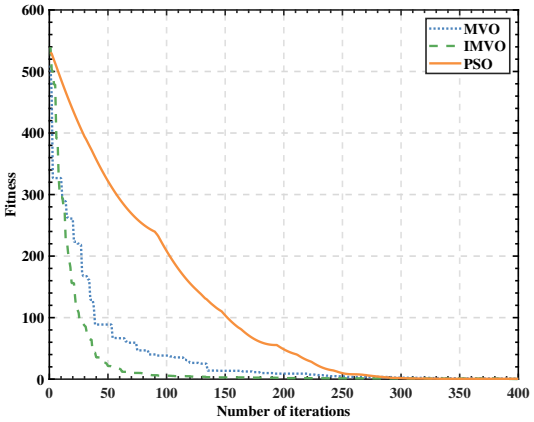
(d) Performance comparison on the f_4 function

Figure 3. Average convergence curve of the standard test functions $f_1 \sim f_4$.

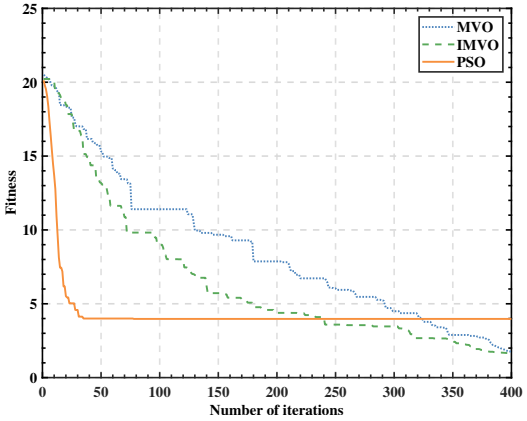
All of the above show that the IMVO algorithm is effective in dealing with unimodal, multimodal, and ill-conditioned test functions, and it has better optimization accuracy and speed, which is very helpful in solving the problems to be optimized in engineering practice.



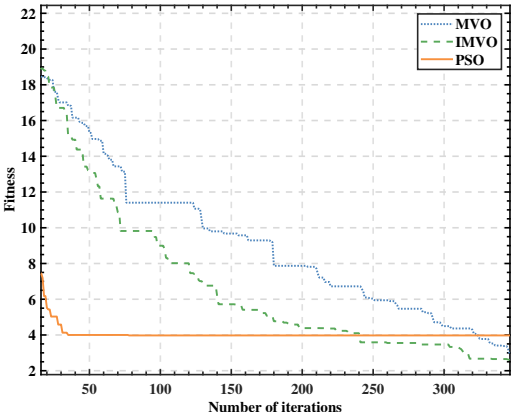
(a) Performance comparison on the f_5 function



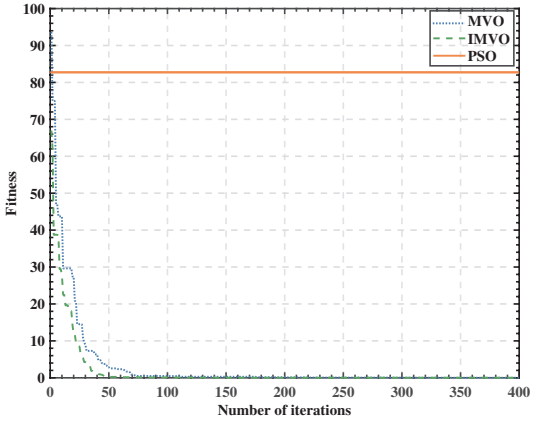
(b) Performance comparison on the f_6 function



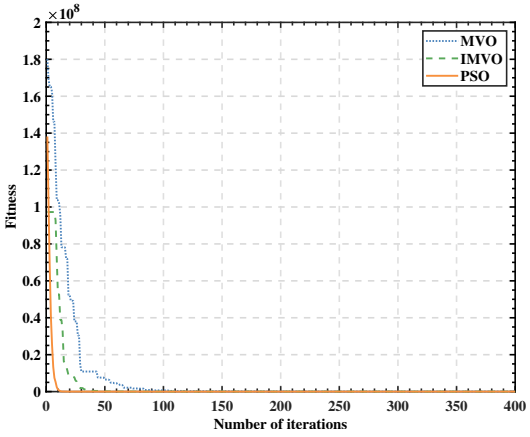
(c) Performance comparison on the f_7 function



(d) Performance comparison on the f_8 function



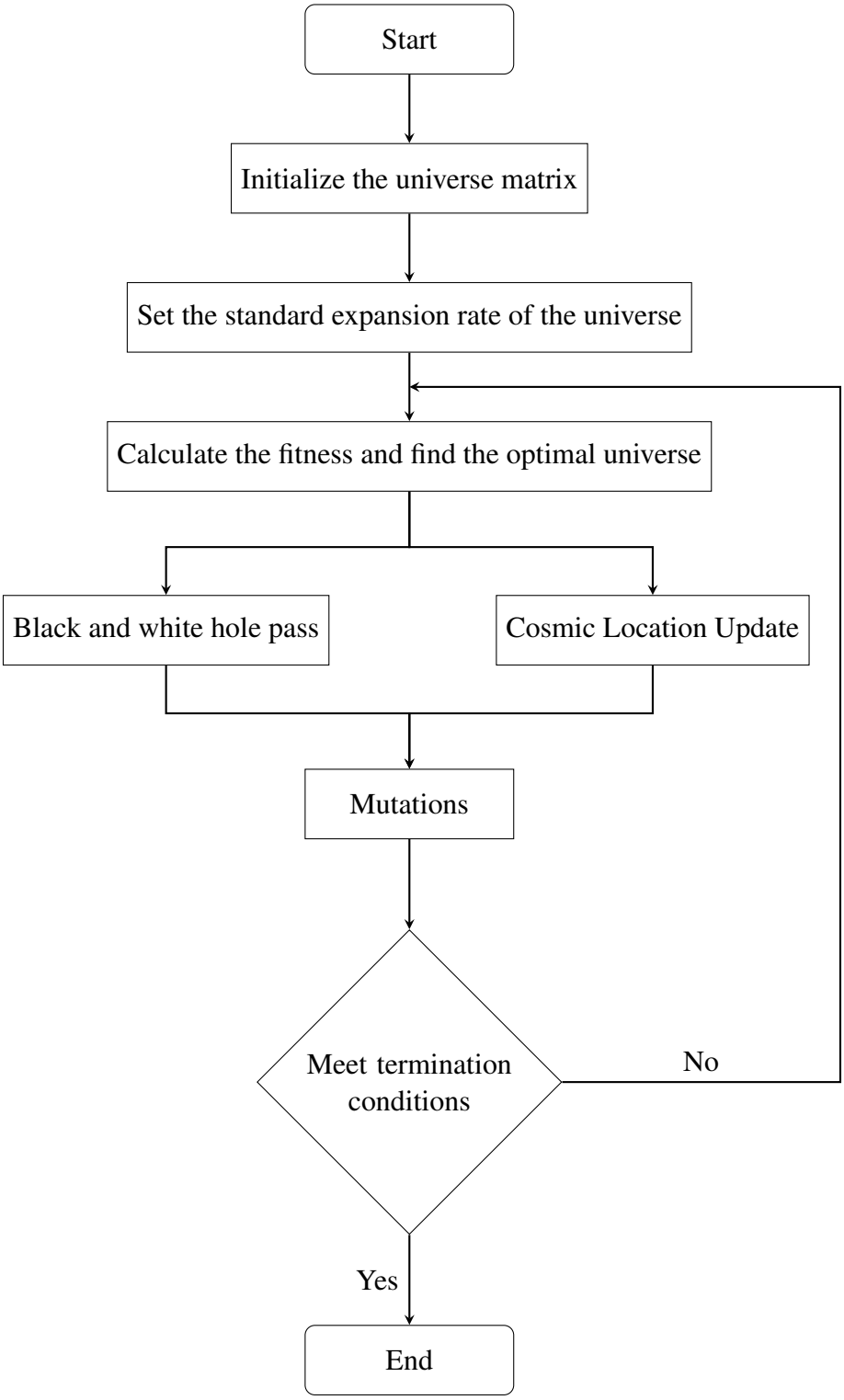
(e) Performance comparison on the f_9 function



(f) Performance comparison on the f_{10} function

Figure 4. Average convergence curve of the standard test functions $f_5 \sim f_{10}$.

3. Algorithm flow



Details as follows:

1) Initialization parameters $\boldsymbol{X}$.

2) Set Normalized Expansion Ratio NI(X) and initialize α_t, WEP and p.

3) Calculate and normalize various fitness of population individuals to find the optimal fitness and normalize it.

4) Black and white hole transfer and update position, α_t and p.

5) Individual variation in populations.

6) If the termination condition is not met, go to Step 3. otherwise, end.

4. Simulation

4.1. Testing environment

Test environment: the hard disk running environment is AMD Ryzen 7 5800X 8-Core Processor, the memory is 16GB, the software running environment is Windows11 system, and the running software is MATLAB 2021a.

4.2. Model building

The D-H parameters of the PUMA560 robot arm are shown in Table 2.

Table 2. D-H parameter table.

number	θ_i/ (°)	d_i/m	a_i/m	α_i
1	0	0	0	$-\pi/2$
2	0	0	0.432	0
3	0	0.149	0.02	$-\pi/2$
4	0	0.433	0	$\pi/2$
5	0	0	0	$-\pi/2$
6	0	0	0	0

Table 3. Each joint position series.

	First Pose	Via Pose 1	Via Pose 2	End Pose
θ_1/rad	0.2	0.8	-0.2	-0.8
θ_2/rad	1.1	1.5	0.8	-0.3
θ_3/rad	-0.2	-0.8	-0.4	-0.1
θ_4/rad	1	0.8	-0.3	-0.7
θ_5/rad	1.4	0.6	1.7	2.6
θ_6/rad	0	-0.7	-1.1	-1.5

As shown in Table 2, θ_i is the joint angle around the Z axis; d_i is the joint distance; a_i is the connecting rod length, and α_i is the torsion around the X axis.

Using the D-H parameters in Table 2, create the model of the PUMA560 manipulator in the four postures in Table 3 in Matlab simulation software. The models are shown in Figure 5.

The joint constraints are shown in Table 4. In order to objectively compare the optimization performance of each algorithm, all algorithms select the same initial population, and the average fitness value, optimal fitness value, standard deviation of fitness value, and average running time of each algorithm running independently for ten times are counted, and the algorithms are comprehensively evaluated. Specific parameter settings are shown in Table 5.

Table 4. Constraints of each joint.

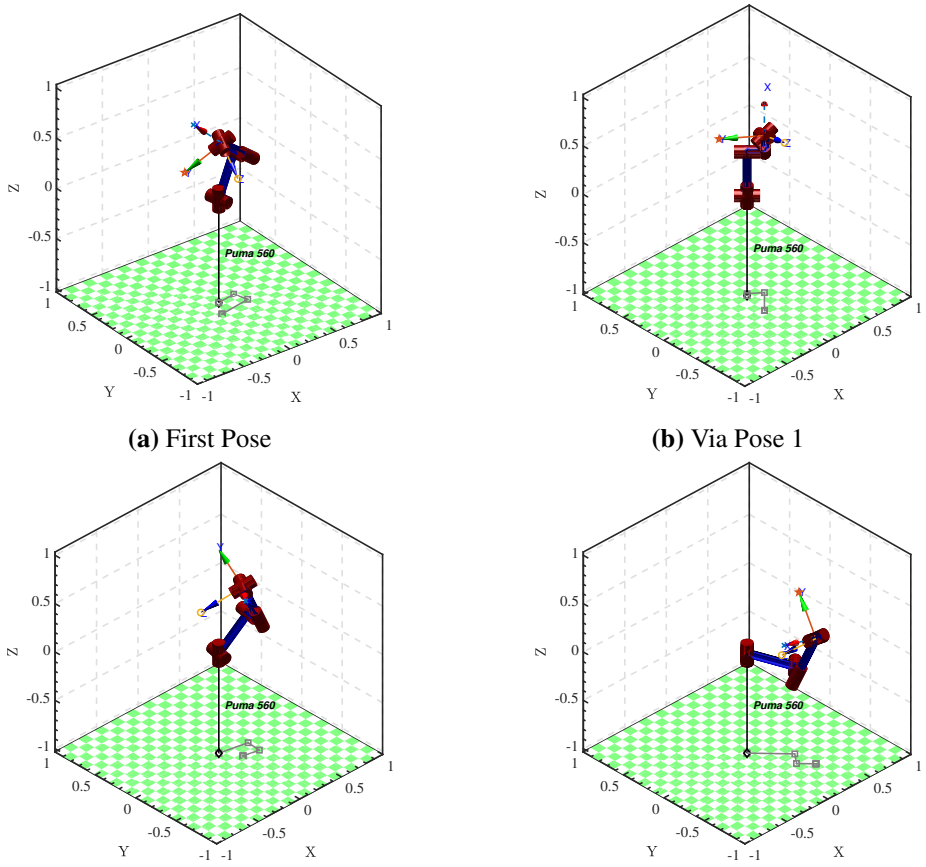
	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Q_{max}/rad	3.14	3.14	3.14	3.14	3.14	3.14
V_{max}/(rad/s)	1.65	2.36	2.33	2.57	2.44	1.85
A_{max}/(rad/s^2)	1.08	1.49	1.88	1.42	1.57	1.44
J_{max}/(rad/s^3)	2.63	2.72	3.14	2.83	2.63	3.14

Table 5. Parameter setting.

	IMVO	MVO	PSO	ASPSO
pop	100	100	100	100
ub	100	100	100	100
lb	0.1	0.1	0.1	0.1
MaxIter	100	100	100	100
v_{max}	-	-	2	2
v_{min}	-	-	-2	-2
WEP_{min}	0.2	0.2	-	-
WEP_{max}	1	1	-	-
Mut	0.001	-	-	-
β	1.00E-03	1.00E-03	-	-
a_0	1	1	-	-
p_{min}	2	-	-	-
p_{max}	20	-	-	-
c_1	-	-	2	2
c_2	-	-	2	2
w_{min}	-	-	-	1
w_{max}	-	-	-	3

4.3. Simulation results and analysis

Trajectory optimization simulation is carried out by MOPSO and MOMVO, and Pareto optimal solution is set with time, Energy, and impact as optimization objectives are obtained. As shown in Figure 6, Energy is positively correlated with impact, and both of them are negatively correlated with time, which jointly restricts the optimal performance of time. Because it is necessary to increase the speed to pursue the shortest time, the energy consumption and impact will increase. In practice, it is necessary to select an appropriate solution in the Pareto optimal solution set according to the specific operating conditions of the manipulator.



(a) First Pose **(b)** Via Pose 1

(c) Via Pose 2 **(d)** End Pose

Figure 5. Simulation model diagram of PUMA560 manipulator.

Since a single objective cannot compare the pros and cons of the solutions in the Pareto optimal solution set, it is essential to define a weighted objective function in order to compare the iterative results. The normalized weighted objective function is defined as

$$f = \frac{S_1}{N_1} + \frac{S_2}{N_2} + \frac{S_3}{N_3} \tag{4.1}$$

N_1, N_2, and N_3 take different values, respectively, and establish different objective functions for comparison. The comparison results of each algorithm are shown in Tables 6 and 7.

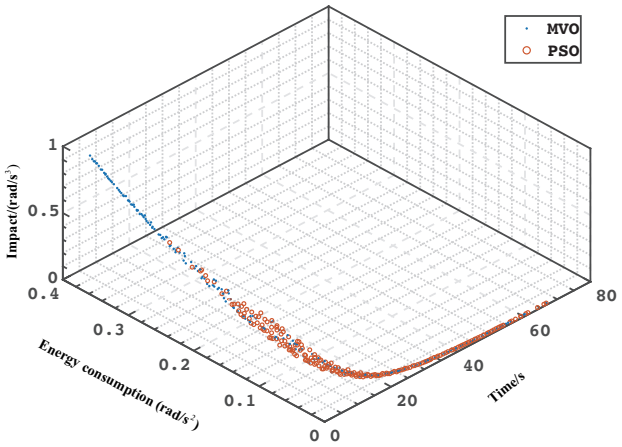
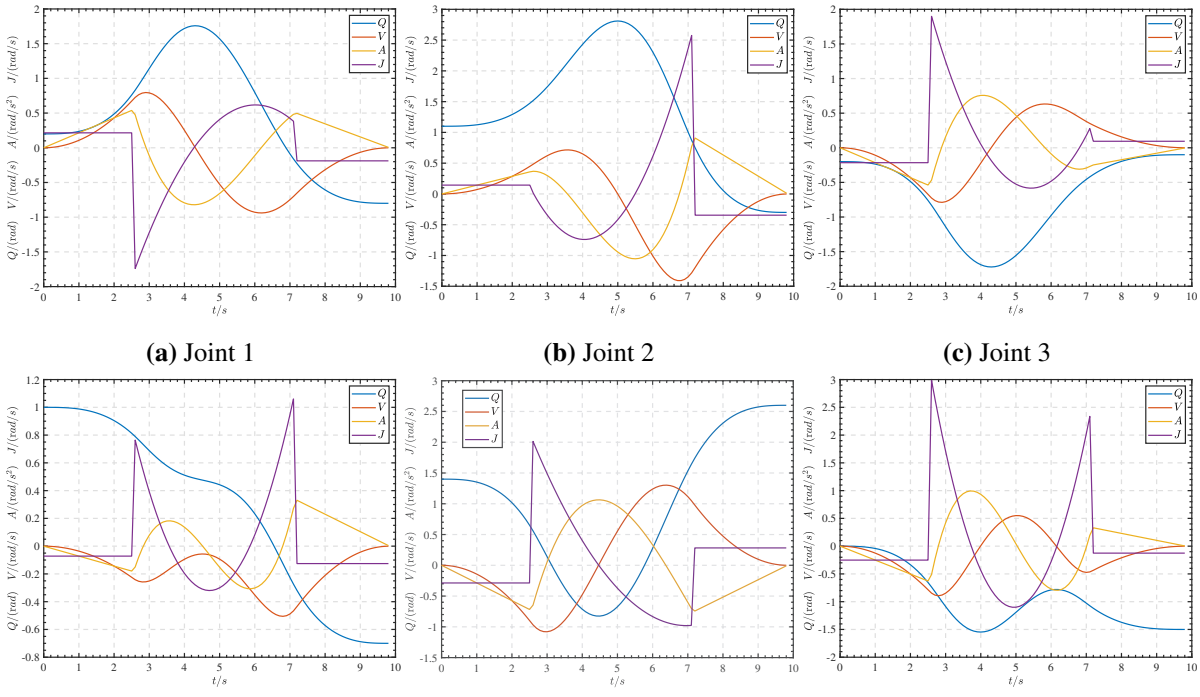


Figure 6. Pareto optimal solution set comparison.



(a) Joint 1 **(b)** Joint 2 **(c)** Joint 3

(d) Joint 4 **(e)** Joint 5 **(f)** Joint 6

Figure 7. Displacement, velocity, acceleration and jerk curves of each joint.

As seen from Figure 9, the improved MVO algorithm quickly obtained iterative convergence in the early stage, and the convergence was stable later.

In addition, as shown in Tables 6 and 7, in order to minimize the error, in a statistical sense, IMVO is better than MVO, PSO, and SAPSO in terms of mean and optimal values of four sets of models constructed with different values of N_1, N_2 and N_3 respectively. The standard deviation of the four data sets is also the smallest IMVO, meaning that the results are more concentrated and the algorithm is more stable. Only the running time of MVO is optimal in the four sets of data because IMVO is an improvement on the MVO, the long running time is normal, and the running time of IMVO is shorter

than PSO and SAPSO. It is only about 1s more than the MVO, which is allowed in engineering. Figure 7 shows that the parameters optimized by IMVO can meet the constraints in Table 4.

The simulation results show that when the constraint conditions are satisfied, the convergence speed and accuracy of the improved MVO are obviously higher than those of MVO, PSO and SAPSO.

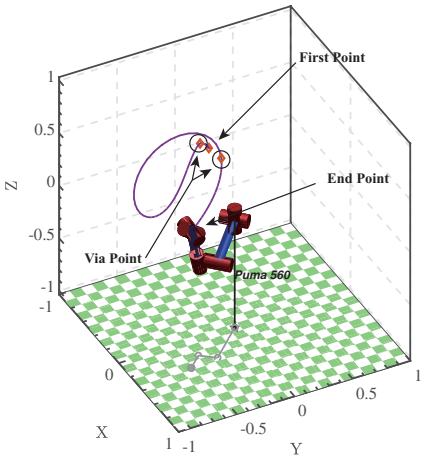


Figure 8. Simulation trajectory diagram.

Table 6. Test result 1.

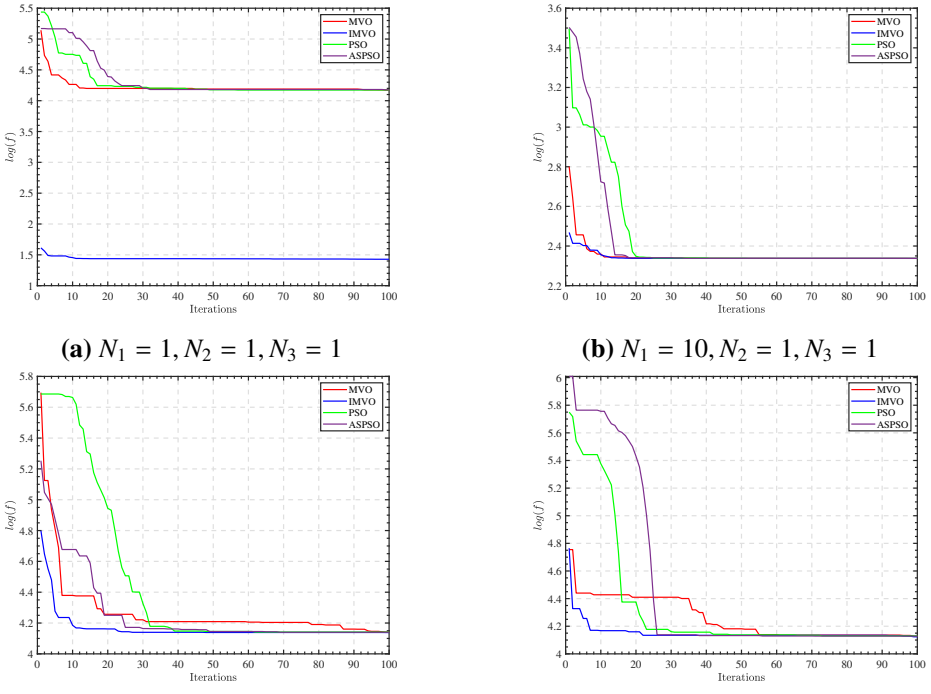
	$N_1 = 1, N_2 = 1, N_3 = 1$				$N_1 = 10, N_2 = 1, N_3 = 1$			
	Mean	Best	Std	Time/s	Mean	Best	Std	Time/s
IMVO	64.7978*	64.7409*	0.0520*	6.08	10.3552*	10.3552*	0.0000*	8.52
MVO	64.9327	64.8866	0.0640	5.09*	10.3554	10.3553	0.0001	6.92*
PSO	65.5215	65.3328	0.1783	6.08	10.3565	10.3553	0.0012	8.82
SAPSO	65.2849	65.1291	0.1969	7.04	10.3560	10.3555	0.0005	8.83

Table 7. Test result 2.

	$N_1 = 1, N_2 = 10, N_3 = 1$				$N_1 = 1, N_2 = 1, N_3 = 10$			
	Mean	Best	Std	Time/s	Mean	Best	Std	Time/s
IMVO	62.6674*	62.6204*	0.0475*	6.18	61.8478*	61.8304*	0.0230*	6.23
MVO	62.8381	62.7740	0.0948	5.19*	61.8927	61.8629	0.0381	4.97*
PSO	63.2617	62.8332	0.3766	6.54	62.6859	62.5218	0.2414	6.45
SAPSO	62.9724	62.8403	0.1389	6.74	62.1954	62.0787	0.1790	6.27

5. Conclusions and future works

The kinematics analysis of the joint space of the 6-DOF manipulator is performed, and the trajectory model of the manipulator in the joint space is constructed by 3-5-3 polynomial interpolation.



(a) $N_1 = 1, N_2 = 1, N_3 = 1$ **(b)** $N_1 = 10, N_2 = 1, N_3 = 1$

(c) $N_1 = 1, N_2 = 10, N_3 = 1$ **(d)** $N_1 = 1, N_2 = 1, N_3 = 10$

Figure 9. Iterative graph under different parameters.

Taking the running time, energy consumption, and impact of each joint of the manipulator as the optimization goal, the optimization is conducted under the specified constraints, and the Pareto optimal solution set with good diversity and convergence is obtained. Customizing the normalized weighted objective function and selecting the optimal solution meeting the conditions from the Pareto optimal solution set is essential.

Compared with MVO, PSO, and SAPSO, the improved MVO algorithm optimizes the iteration result, improves the optimization structure, and improves the algorithm efficiency.

This paper only discusses 3-5-3 polynomial interpolation. As future work, IMVO can optimize other interpolation functions such as cubic polynomial and quintic polynomial. IMVO can also be applied to optimization in other fields, such as parameter structure optimization of machine learning, fuzzy control optimization, inverse kinematics solution of the manipulator, etc.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. P. Ngatchou, A. Zarei, A. El-Sharkawi, Pareto multi objective optimization, in *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, (2005), 84–91. https://doi.org/10.1109/ISAP.2005.1599245

2. R. Benotsmane, L. Dudás, G. Kovács, Trajectory optimization of industrial robot arms using a newly elaborated "whip-lashing" method, *Appl. Sci.*, **10** (2020). https://doi.org/10.3390/app10238666

3. S. Han, X. Shan, J. Fu, W. Xu, H. Mi, Industrial robot trajectory planning based on improved pso algorithm, *J. Phys.: Conf. Ser.*, **1820** (2021), 012185. https://doi.org/10.1088/1742-6596/1820/1/012185

4. X. Peng, G. Chen, Y. Tang, C. Miao, Y. Li, Trajectory optimization of an electro-hydraulic robot, *J. Mech. Sci. Technol.*, **34** (2020), 4281–4294. https://doi.org/10.1007/s12206-020-0919-4

5. K. Ota, D. K. Jha, T. Oiki, M. Miura, T. Nammoto, D. Nikovski, et al., Trajectory optimization for unknown constrained systems using reinforcement learning, in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (2019), 3487–3494. https://doi.org/10.1109/IROS40897.2019.8968010

6. X. Shi, H. Fang, G. Pi, X. Xu, H. Chen, Time-energy-jerk dynamic optimal trajectory planning for manipulators based on quintic nurbs, in *2018 3rd International Conference on Robotics and Automation Engineering (ICRAE)*, (2018), 44–49. https://doi.org/10.1109/ICRAE.2018.8586763

7. G. I. Sayed, A. Darwish, A. E. Hassanien, Quantum multiverse optimization algorithm for optimization problems, *Neural Comput. Appl.*, **31** (2019), 2763–2780. https://doi.org/10.1007/s00521-017-3228-9

8. W. P. Bailón, E. B. Cardiel, I. J. Campos, A. R. Paz, Mechanical energy optimization in trajectory planning for six dof robot manipulators based on eighth-degree polynomial functions and a genetic algorithm, in *2010 7th International Conference on Electrical Engineering Computing Science and Automatic Control*, (2010), 446–451. https://doi.org/10.1109/ICEEE.2010.5608583

9. S. Lu, Y. Li, Minimum-jerk trajectory planning of a 3-DOF translational parallel manipulator, in *39th Mechanisms and Robotics Conference of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2015. https://doi.org/10.1115/DETC2015-46866

10. H. I. Lin, Y. C. Liu, Minimum-jerk robot joint trajectory using particle swarm optimization, in *2011 First International Conference on Robot, Vision and Signal Processing*, (2011), 118–121. https://doi.org/10.1109/RVSP.2011.70

11. P. Boscariol, A. Gasparetto, R. Vidoni, Planning continuous-jerk trajectories for industrial manipulators, in *ASME 2012 11th Biennial Conference on Engineering Systems Design and Analysis*, 2012. https://doi.org/10.1115/ESDA2012-82103

AIMS Press